Alternativa k systémovému zobrazení hodin

Technická zpráva k projektu do předmětu ITU
FIT VUT v Brně, 2020

Název týmu
xhubik03

Autoři
Kateřina Mušková, xmusko00
David Holas, xholas11
Antonín Hubík, xhubik03

1. Zadání a organizace týmu

1.1 Cíl

Cílem projektu je vytvořit alternativu k systémovým hodinám, snadno viditelnou v rámci dané platformy. Hodiny budou podporovat přepínání mezi digitálním a analogovým vzhledem, nastavení barev, velikosti a průhlednosti či výplně pozadí. Pro podporu těchto možností bude vyrobeno vhodné ovládání, snažící se o vysokou míru jednoduchosti a intuitivity.

Projekt bude realizován jako aplikace pro Ubuntu GNOME, kde se bude narozdíl od klasických desktopových widgetů zobrazovat nad všemi okny, a jako aplikace na iOS 14 a vyšší, podporující zobrazení ve formě widgetu a umožňující výrazné zobrazení sekund, které je ve zde ve výchozí aplikaci Hodiny potlačené.

Aplikace je zamýšlena jako přizpůsobitelný doplněk desktopového a mobilního pracovního prostředí, komplementární k informačním a organizačním nástrojům jako je zobrazování počasí či kalendář.

1.2 Tým

Společná část

- Průzkum existujících řešení a uživatelských potřeb
- Testování aplikací
- Návrh backendu

iOS aplikace

- xholas11

Desktop aplikace

- Vzhledy a varianty zobrazovaných informací - xhubik03
- Ovládání uživatelských nastavení - xmusko00
- Finální design - xhubik03
- MVC - xmusko00

1.3 Roadmapa

První část - zadání, průzkumy, návrh aplikace

- do 4. 10. 2020 - volba a formulace zadání, předběžné rozdělení práce (proběhlo)
- **5. 10.** - obhajoba zadání, příjem zpětné vazby od vyučujícího (proběhlo)
- 7. 10. - zpracování připomínek a zpětné vazby od “investorů” (proběhlo)
- 15.10. - definice rizik a opatření, dodatečné rozdělení práce (proběhlo)

- 15. 10. - 25.10. - průzkum existujících řešení na daných platformách, průzkum uživatelských potřeb (proběhlo)
- 15. 10. - 27. 10. - návrh architektury řešení, návrh GUI pro obě platformy, vytvoření wireframů pro pilotní testy (proběhlo)
- 27.10. - 29. 10. - pilotní testy (proběhlo)
- 29. 10. - shrnutí výsledků pilotních testů
- 29. 10. - 31. 10. - revize návrhu na základě výsledků pilotních testů
- 30. 10. - 1. 11. - sepsání první části technické zprávy
- **1. 11. 2020** - odevzdání první části technické zprávy

Druhá část - implementace GUI, závěrečné testy, zhodnocení výsledků

- 9. 11. - 30. 11. - implementace GUI pro hodiny a jejich nastavení na iOS
- 9. 11. - 23. 11. - implementace GUI pro zobrazování hodin na desktopu
- 16. 11. - 30. 11. - implementace GUI pro nastavení hodin na desktopu
- **1. 12.** - jsou k dispozici plné testovací verze obou aplikací
- 1. 12. - 4. 12. - finální testování
- 4. 12. - shrnutí výsledků finálního testování
- 5. 12. - 7. 12 - případné revize závažných nedostatků
- 8. 12. - 10. 12. - dokončení technické zprávy
- **10. 12**. - prefinální verze obou implementací
- **10. 12.** - prefinální verze technické zprávy
- 11. 12. - 13. 12. - rezerva
- **13. 12. 2020** - termín odevzdání celého řešení

1.4 Rizika a opatření

Možné problémy a řešení:

- Nefunkční SDK: použití virtuálního stroje
- Neshody v týmu: hlasování (tým má lichý počet členů, rozhodnutí je tedy zaručeno)
- Nedokončení aplikace: rozvrhnutí si časového plánu, který se musí dodržovat

Zajištění bezproblémového vývoje

- Verzování a sdílení kódu: použití verzovacího nástroje Git (ve službě Bitbucket)
- Zajištění kvality kódu: code review po každé větší změně
- Komunikace v týmu: plánování pravidelných společných online setkání a textová komunikace přes skupinový chat

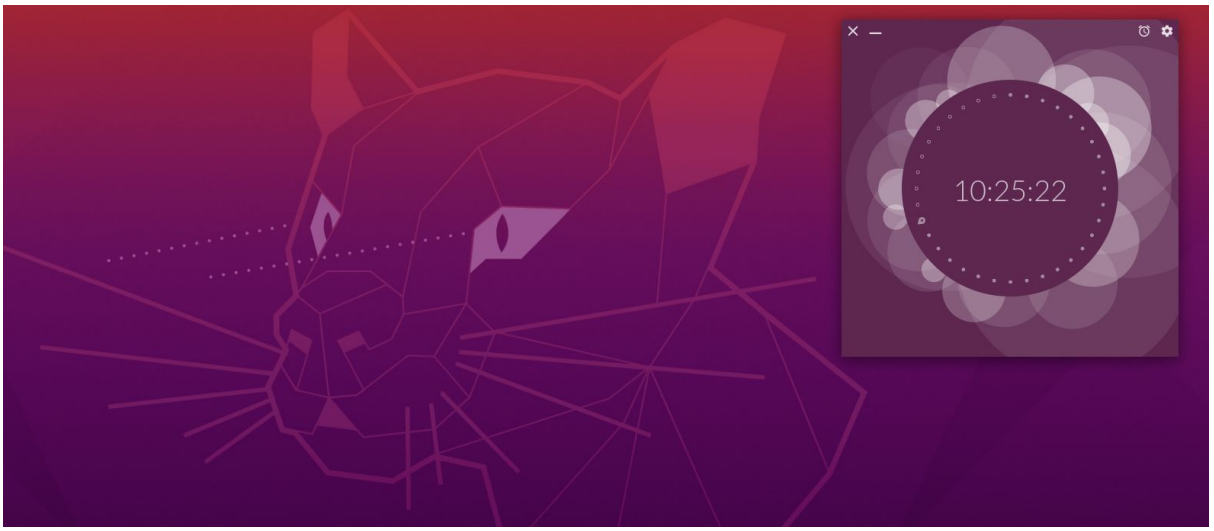
2. Průzkum a zkušenosti

2.1 Existující řešení

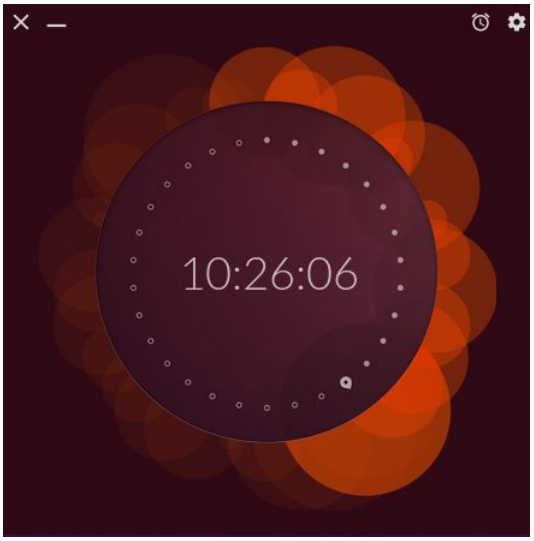
Desktop Aplikace (Kateřina Mušková)

UP Clock[1]

Jednoduchá hodinová aplikace s nastavením formátu času a barvou.
Chová se jako aplikace s běžným oknem. Nezůstává ani na ploše, ani není plovoucí.
I přes možnost nastavení barev může aplikace aplikace působit rušivým dojmem kvůli tvrdým okrajům.



UP Clock s nastavením fialové barvy z palety Ubuntu



UP Clock ve výchozím nastavení

[1] http://upclock.tk

GNOME Clock[2]

Naprosto minimalistická aplikace s alarmem a stopkami. Styl je velmi jednoduchý bez jakékoliv personalizace. Schází alespoň základní nastavení.

GNOME Clock

Desktop Aplikace (Antonín Hubík)

Conky Lua[3]

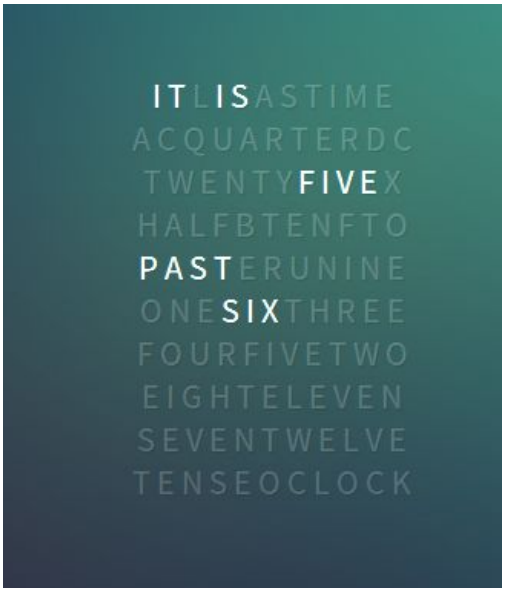
Jedná se o základní widget Conky.
Design je barevně uzpůsoben každé distribuci zvlášť, vizuálně tedy dobře sedí k výchozí ploše.
Tento widget zůstává na ploše permanentně, není otevřen v klasickém okně.
Kromě času také přehledně zobrazuje doprovodné informace jako využití RAM nebo CPU, čímž se z hodin stává spíše light tool s diagnostikou počítače. Jako dva jeho možné nedostatky pro některé uživatele by bylo možné vnímat prostorovou náročnost widgetu a upřednostnění designu před snadným čtením analogového času.

[2] https://wiki.gnome.org/Apps/Clocks
[3] https://www.gnome-look.org/p/1115225

Conky Lua v prostředí Mint, Ubuntu a Fedora

qlocktwo conky[4]

Opět se jedná o widget, který zůstává na ploše. Tentokrát s minimalistickým a originálním designem, ale opět za cenu ztížení čitelnosti času. Silně upřednostňuje estetiku.



Hodiny (David Holas)

Nejrozšířenějším příkladem existujícího řešení je vestavěná aplikace Hodiny v systému iOS a její widget. Aplikace neumožňuje žádnou možnost konfigurace. Není možné změnit barvy

[4] https://www.deviantart.com/mowgli-writes/art/qlocktwo-conky-470067388

ani změnit zobrazení na digitální hodiny. Sekundy jsou sice na analogových hodinách zobrazovány pomocí třetí ručičky, ale v systému není možné zobrazit digitální reprezentaci sekund.

Flip Clock (David Holas)

Design se snaží vypadat jako překlápěcí hodiny. Neumožňuje analogové zobrazení, ale zobrazuje datum. Vzhled se nedá příliš nastavit.

Clock Widget (David Holas)

Umožňuje výběr pouze ze čtyř přednastavených vzhledů, mezi kterými jsou ale jak digitální tak analogové hodiny.

2.2 Uživatelské potřeby

Rozprava s potenciálními uživateli proběhla neformálně bez dotazníků, v některých případech posloužila jako odrazový můstek zpětná vazba k již existujícím řešením, zejména co v těchto řešeních uživatelům chybělo.

Desktop aplikace (Kateřina Mušková, Antonín Hubík)

Náš průzkum proběhl na dvou uživatelích Ubuntu.

V prvním případě se jednalo o začátečnického uživatele, který počítač využívá především k psaní dokumentů a přístupu k internetu.
V jeho případě by byly ideální klasické hodiny, které by nebylo třeba nijak upravovat, ale po jednoduché instalaci by hned fungovaly. Daná osoba neovládá angličtinu, proto musí být nastavení v češtině. Tento uživatel je navíc starší člověk a preferuje analogové zobrazení hodin před digitálními a výchozí zobrazení času na Ubuntu pro něho navíc může být trochu malé a tudíž nepříjemné na čtení.

Ve druhém případě se jedná o pokročilého uživatele, který se věnuje designu a rád experimentuje.
Hodiny by si naopak rád přizpůsobil svému zvolenému tématu a tapetě. Pokud by měla být aplikace na popředí, neměla by podle něj chybět funkce na nějakou formu minimalizace v případě nutnosti. Celý systém má v angličtině a anglickou mutaci pokládá také za samozřejmou.

Aplikace pro iOS (David Holas)

Aplikace je určena pro uživatele, kteří by rádi měli na své domovské stránce velkým zobrazeny digitální hodiny (i včetně zobrazených sekund), protože na systémové digitální hodiny špatně vidí (kvůli věku, atp). Současně další skupinou uživatelů jsou ti, kterým se vestavěný vzhled nelíbí nebo jim nevyhovuje a chtěli by jej upravit tak, aby víc seděl do jeho konkrétního stylu. Většina dotázaných uživatelů by ocenila možnost průhledného nebo

poloprůhledného pozadí. Uživatelé by často chtěli na vzhledu vestavěných hodin pouze upravit/přidat/odebrat pár maličkostí tak aby pro ně zobrazení bylo více přehledné nebo jim více vyhovovalo. Někteří uživatelé chtěli v analogovém módu mít možnost zobrazit na ciferníku buď čárky nebo čísla.

2.3 Shrnutí

Desktop aplikace (Kateřina Mušková, Antonín Hubík)

Z dosavadních aplikací lze nalézt některé s příjemným designem, vyšší mírou nastavitelnosti, nebo s dobrou přehledností, ale žádná z nich tyto vlastnosti nekombinuje ve vyváženém poměru.

Aplikace by měla být snadno použitelná bez předchozího nastavení, ale s možností toto nastavení ovlivnit.

Zatím existují buď aplikace formou widgetu, které ale zůstávají na pozadí, nebo aplikace mající běžné okno. Při srovnávání jsme ale nenašli žádné plovoucí řešení, umístěné na podobně jako překryv nade všemi okny. Tento styl můžeme vidět například u mobilní verze facebooku.

Chtěli bychom poskytnout základní možnosti přizpůsobení, obsahující nevyhrocené varianty vhodné pro širší okruh uživatelů, například do aplikace začleníme čtyřiadvacetihodinový i dvanáctihodinový formát času, stejně jako jazykové mutace - v základu alespoň CZ a EN.

Aplikace pro iOS (David Holas)

Naše aplikace bude umožňovat zobrazit hodiny ve stylu inspirovaném vestavěného widgetu hodin. Oproti vestavěné aplikaci však bude podporovat i digitální zobrazení a možnost podrobného nastavení vzhledu, tak aby si aplikaci mohl každý uživatel přizpůsobit k obrazu svému.

3. Architektura řešení

3.1 Architektura aplikací

Obě aplikace jsou rozděleny do dvou částí - samotné hodiny a jejich nastavení - majících společný datový model.

Desktop aplikace (Kateřina Mušková, Antonín Hubík)

Aplikace bude v souladu se zadáním implementována s využitím návrhového vzoru MVC.[5]

[5] https://doc.qt.io/archives/qq/qq10-mvc.html

Model

Model bude vzhledem k povaze aplikace (není zapotřebí žádných větších výpočtů nad daty apod.) pouze načítat a ukládat (přepisovat) uživatelská nastavení a při spuštění načítat systémový čas.

View

View bude implementovat vizualizaci hodin a uživatelských nastavení. V nastavení se bude odehrávat drtivá většina možných interakcí ze strany uživatele, s výjimkou změny velikosti okna, minimalizace okna a ukončení aplikace. Úspěšné provedení interakce bude viditelně indikováno uživateli.

Controller

Controller bude na základě akcí uživatele obstarávat aktualizaci dat ve View a v Modelu a zajišťovat jejich vzájemnou konzistenci. V našem případě se zde nachází pravděpodobně především tato “spojovací” logika.

Aplikace pro iOS (David Holas)

Widget

Jádrem části aplikace, která obsluhuje widget bude takzvaný *Provider*, který se stará o odesílání dat z *modelu* přímo do widgetu v daných časových intervalech. Samotný widget pak bude pouze uživatelské rozhraní, které přijatá data pouze zobrazuje.

Nastavení

Část aplikace s uživatelskými nastaveními bude mít více konvenční architekturu. Základem je hlavní obrazovka (*view*) na které se zobrazují všechny možné nastavení. Její *controller* se bude starat o zobrazování uživatelských nastavení z modelu na obrazovce. Změna každého nastavení bude prováděna na nové obrazovce a její *controller* se bude starat o předání změny v *modelu*. Aplikace by měla být v kódu snadno rozšiřitelná, tak aby se daly jednoduše přidávat další možnosti nastavení.

Model

Data budou ukládána pomocí jednoduchého uložiště klíč-hodnota. *Model* bude toto úložiště konvertovat na objekt se kterým mohou *controllery* snadno pracovat. *Model* je společný pro obě výše zmíněné části aplikace.

3.2 Datový model

Desktop aplikace (Kateřina Mušková, Antonín Hubík)

Jedinými potřebnými daty jsou uživatelské nastavení hodin, trvale uložené v konfiguračním souboru na lokálním stroji, a aktuální čas, který bude přebrán z času systémového. Aplikační rozhraní datového modelu bude dáno sadou metod pro získávání a nastavování jednotlivých položek nastavení.

Aplikace pro iOS (David Holas)

Datový model je velmi jednoduchý, sestává z jedné entity obsahující všechna uživatelská nastavení.

3.3 Vybrané technologie a implementace

Desktop aplikace (Kateřina Mušková, Antonín Hubík)

Aplikace bude implementována ve frameworku Qt[6] v prostředí Qt Creator. Hlavním programovacím jazykem zde bude C++ (pro většinu Modelu a přinejmenším část Controlleru), doplněný o QML[7] pro frontend a především deklarativní popis komponent. Zde je pravděpodobné využití kombinace přímého QML a UI navrženého v Qt Designeru.

Aplikace pro iOS (David Holas)

Celá aplikace je implementována v jazyce Swift. Možnost přidávat widgety je na iOS nová a proto musí být implementována pomocí nové knihovny SwiftUI[8], která specifikuje uživatelské rozhraní deklarativně. Pro část aplikace s nastavením jsem zvolil starší knihovnu UIKit[9], která specifikuje uživatelské rozhraní imperativně. Tuto knihovnu jsem zvolil hlavně proto, že je v ní víc vidět architektura MVC.

4. Návrh GUI - nastavení desktop aplikace [xmusko00]

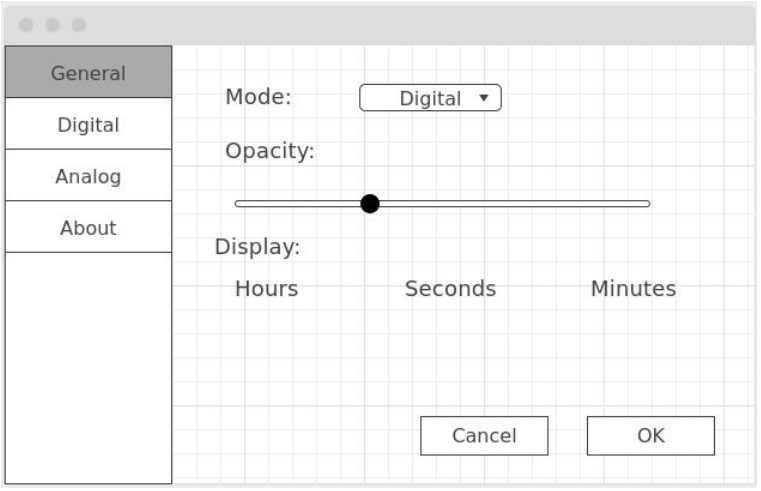
4.1 Požadavky na GUI

Chtěla bych se zaměřit hlavně na přehlednost, intuitivnost a snadnou nastavitelnost. Aby zobrazení sedělo k platformě, bude se nastavení po jeho vyvolání zobrazovat uprostřed plochy jako klasické okno.

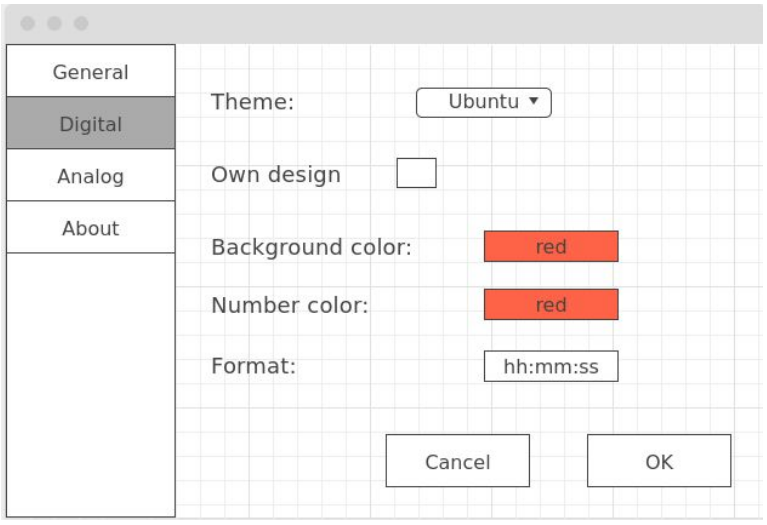
Nastavení bude rozděleno na obecnou část, společnou pro všechny varianty hodin, a poté na dvě další části pro detailnější nastavení zvlášť analogových a zvlášť digitálních. Odsud se budou provádět všechny změny v perzistentních datech. Při zobrazení bude vždy načtena a v rozhraní zobrazena dosavadní konfigurace, při jejíž změně se budou data propisovat přes kontrolér jednak do modelu a jednak do samotných hodin (viz návrh hodin od xhubik03).

4.2 Makety

[6] https://doc.qt.io/
[7] https://doc.qt.io/qt-5/qtqml-index.html
[8] https://developer.apple.com/xcode/swiftui/
[9] https://developer.apple.com/documentation/uikit



Wireframe obecného nastavení hodin



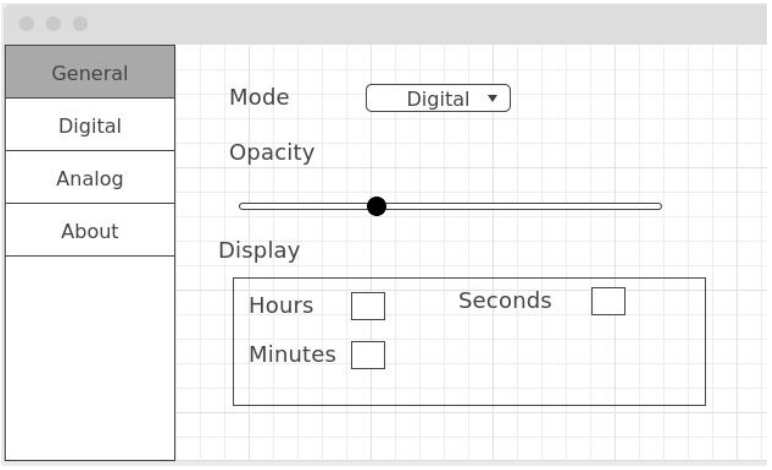
Wireframe specifického nastavení digitálních hodin

4.3 Pilotní test

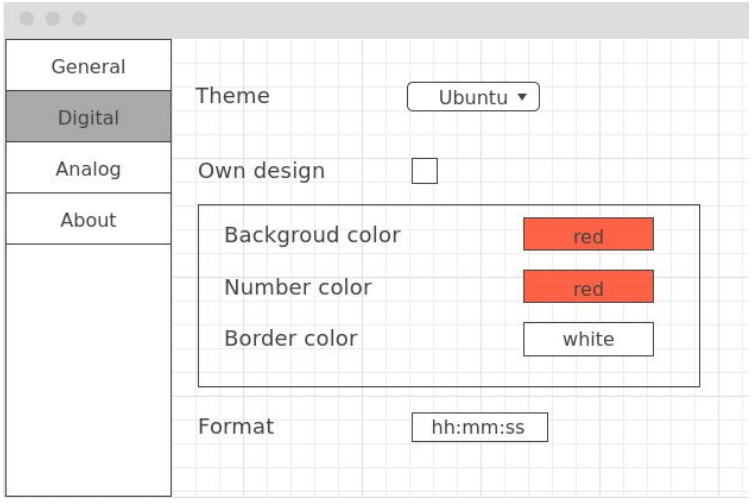
Pilotní test byl realizován na pokročilém uživateli Ubuntu. Vzhledem mu návrh připomínal nastavení Ubuntu. Stejně jako v Ubuntu by podle něj bylo na místě odstranit tlačítka cancel a ok.
Jinak bylo nastavení označeno za dostatečně přehledné a intuitivní.

4.4 Vyhodnocení testu a revize návrhu

Odstranění tlačítek ok a cancel dává smysl. Změny v nastavení se rovnou promítnou do zobrazení hodin. Nebude tudíž třeba nastavení zavírat, aby člověk mohl zjistit výsledek.



Upravený wireframe obecného nastavení hodin



Upravený wireframe specifického nastavení digitálních hodin

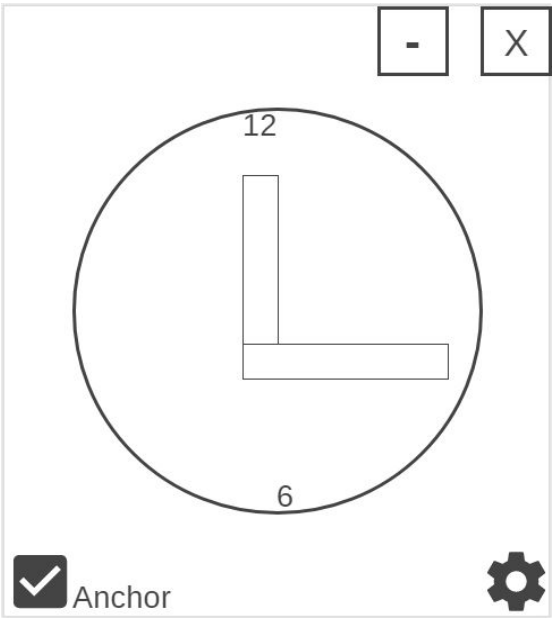
4. Návrh GUI - hodiny a základní ovládání desktop aplikace [xhubik03]

4.1 Požadavky na GUI

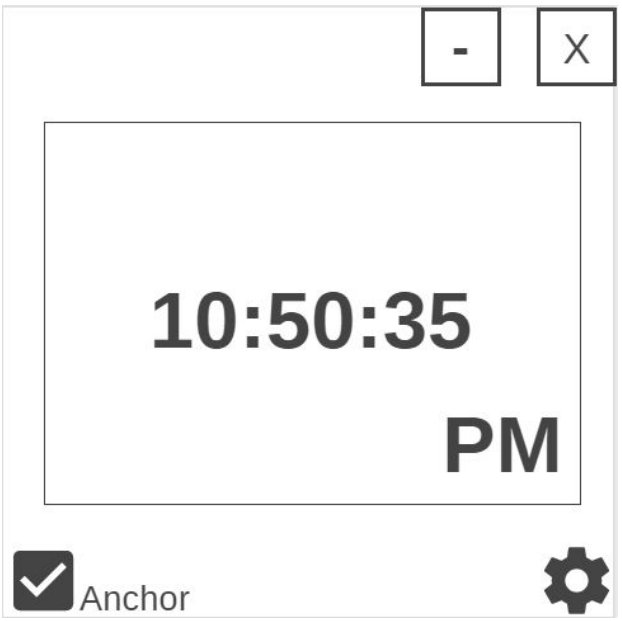
Hodinový ciferník, ať už v analogové nebo digitální variantě, bude především realizovat sice pasivní, avšak pravděpodobně nejběžnější uživatelský požadavek - viditelně informovat o aktuálním čase. V obou hlavních zobrazovacích režimech by tedy měl ve výchozím nastavení mít dostatečně výrazné nejdůležitější prvky a zcela průhledné pozadí, aby nezakrýval více plochy, než je nutné. Kromě samotného zobrazování zde musí být tlačítko pro minimalizaci, případně zavření okna a tlačítko umožňující přístup do nastavení. Navíc je v návrhu následující podkapitole ještě přidána možnost ukotvit hodiny, vyňatá pro rychlejší dostupnost z hlavního nastavení.

Právě zde se projevuje většina změn v datech aplikace: varianta zobrazení času, formát, římské/arabské číslice, barvy jednotlivých prvků, průhlednost pozadí, atd.

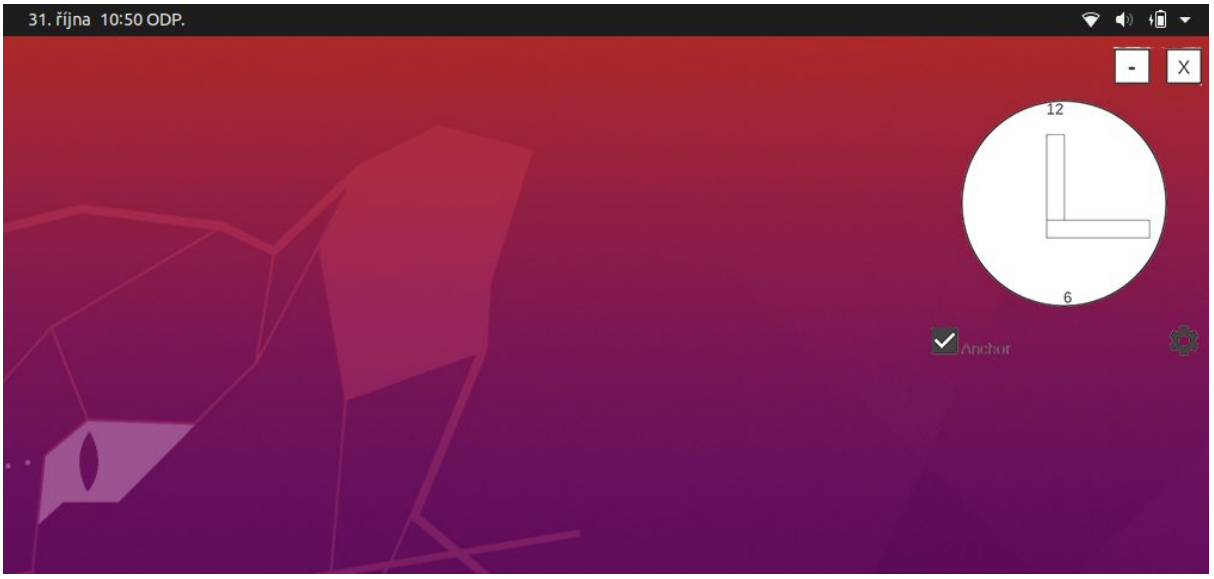
4.2 Wireframy



Analogový hodinový ciferník



Digitální hodinový ciferník, dvanáctihodinový



Ilustrační ukázka možného umístění na pracovní ploše v Ubuntu 20.04

4.3 Pilotní test

Pilotní test byl proveden na běžném středně pokročilém uživateli, který byl nejprve požádán, aby popsal, co podle jeho názoru jednotlivé ovládací prvky znamenají. Velmi rychle a bez problému vesměs na první pokus identifikoval všechny potřebné akce, pouze se zarazil u prozatímního checkboxu pro ukotvení, protože neznal anglické slovo “anchor”. Také si postěžoval, že není na pozadí příliš dobře vidět. Proto by bylo vhodné v implementaci toto vyřešit vhodnou ikonou. Dále by ocenil, kdyby se při přetažení hodin na levou stranu obrazovky tlačítko pro zavření a přetažení překlopilo podle osy na druhou stranu, aby hodiny lépe seděly i do opačného rohu. Poté byl požádán, aby provedl přetažení a změnu velikosti, což udělal zamýšleným způsobem.

4.4 Vyhodnocení testu a revize návrhu

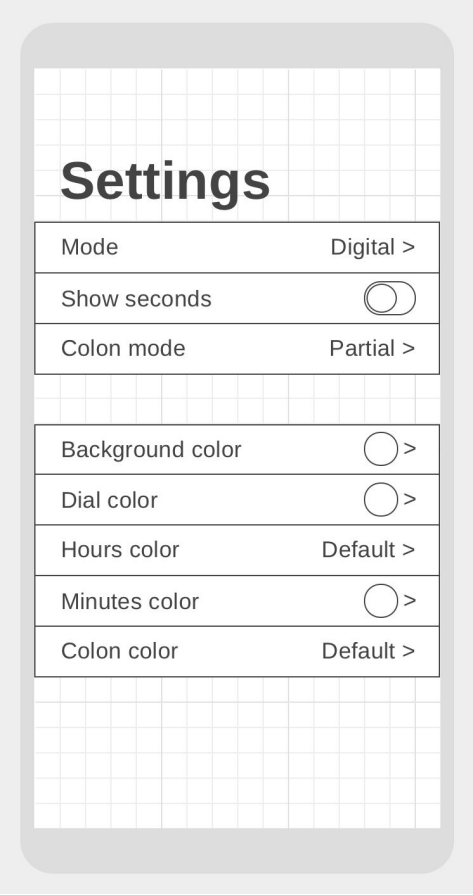
Uživatelská zpětná vazba byla víceméně pozitivní, navržené úpravy se týkají spíše dynamické reakce rozhraní na události, takže nevyžadují úpravy wireframu.
Lze ovšem ještě vyvodit, že česká jazyková mutace celé aplikace je obecně dost na místě.

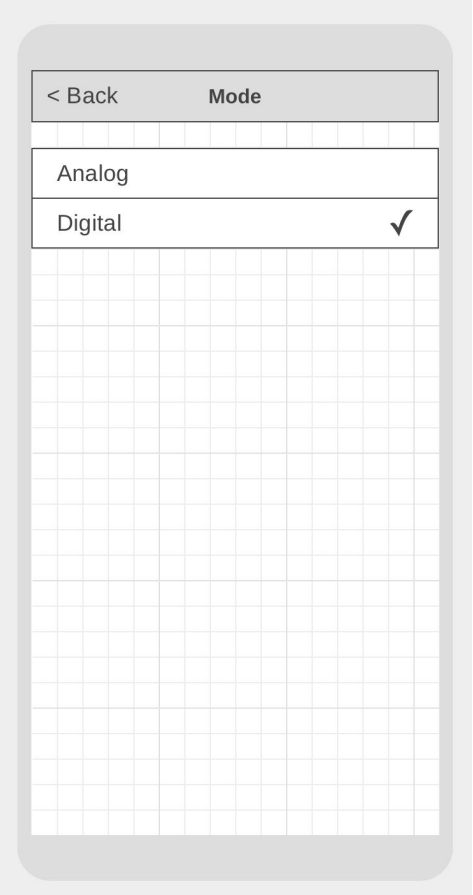
4. Návrh GUI - aplikace pro iOS [xholas11]

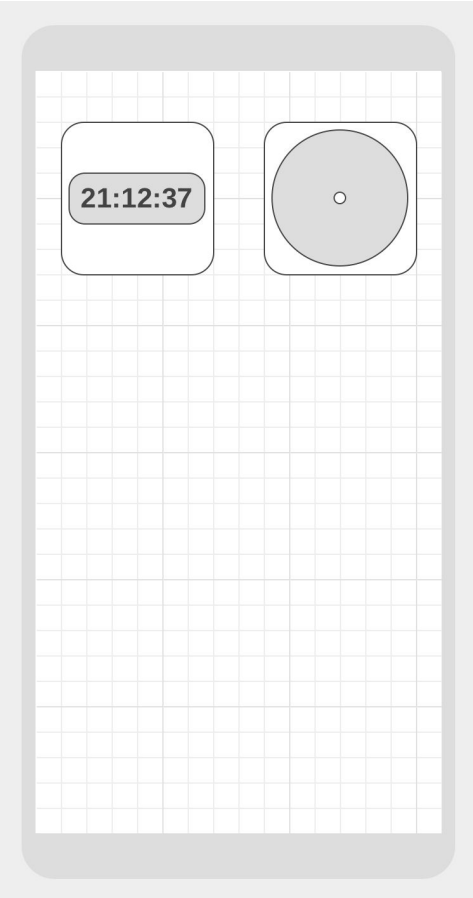
4.1 Požadavky na GUI

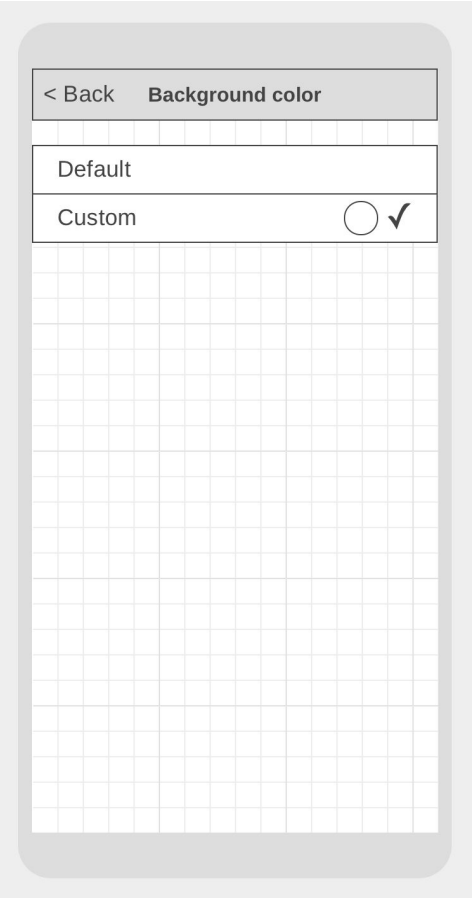
Uživatel by měl mít možnost intuitivně nastavit vzhled všech komponent widgetu hodin a to pomocí prostředí ve kterém se umí orientovat. Aplikace by měla obsahovat výchozí styl, který bude závislý na vzhledu systému (světlý/tmavý). Výchozí styl bude využívat systémové barvy, tak aby co nejvíce zapadl do celkového designu systému iOS. Toto by mělo uživatelům umožnit upravit pouze pár drobností a zbytek aplikace ponechat ve výchozím stylu.

4.2 Makety









4.3 Pilotní test

Pilotní test pomocí náčrtků byl realizován na 2 uživatelích. Bylo sledováno zda uživatel dokáže snadno najít příslušné nastavení a změnit jej na požadovanou hodnotu.
Úkoly byly:

- Změnit barvu dvojteček v digitálních hodinách.
- Změnit mód z digitálního na analogový.
- Zapnout zobrazování sekund.

První pokorčilý uživatel neměl s ovládánim aplikace problémy.
Druhý méně pokročilý uživatel, také dokázal ovládat aplikaci, ale problém mu kvůli anglické lokalizaci činila některá slovíčka použitá v nastavení (např. „colon“).

4.4 Vyhodnocení testu a revize návrhu

Uživatelé snadno pochopili jak hodnoty v nastavení upravovat. Největším problémem tedy byla jazyková bariéra, která se však díky jednoduchosti aplikace dá překonat tím, že uživatel nastavení změní a podívá se na výsledek přímo na widgetu. Případně přidáním české lokalizace, což však tento problém neřeší pro uživatele nemluvící česky ani anglicky.

5. Implementace GUI - nastavení desktop hodin [xmusko00]

5.1 Implementace

Implementace staví na architektuře Model View Controller.

Datový model

Data, která jsme potřebovali zahrnout do naší aplikace, je konkrétní nastavení hodin, které se promítá do zobrazeného vzhledu hodin. Toto nastavení se dá rozdělit do 3 základních částí, a to :

- Obecné nastavení
- Nastavení digitálních hodin
- Nastavení analogových hodin

Tyto tři části jsou implementovány jako tři základní modely *(GeneralModel*, *AnalogModel* a *DigitalModel*).
Každý z těchto modelů musí v systému existovat pouze v jedné instanci, proto jsem využila návrhového vzoru Jedináček, který toto zajišťuje. Všechny tři modely také sdílí část kódu, který je obsažen v *SettingModel*, ze kterého všechny tři dědí.

Jelikož nastavení musí zůstat permanentní i po ukončení aplikace, je třeba jej uchovávat. Qt nabízí velice šikovný způsob vytváření konfiguračního souboru QSettings, jehož formát i lokace je závislá na použitém operační systému.
V případě Ubuntu je nastavení hodin uloženo v ~/.config/StackUndeflow/ITU_Clock.conf

Kontrolér

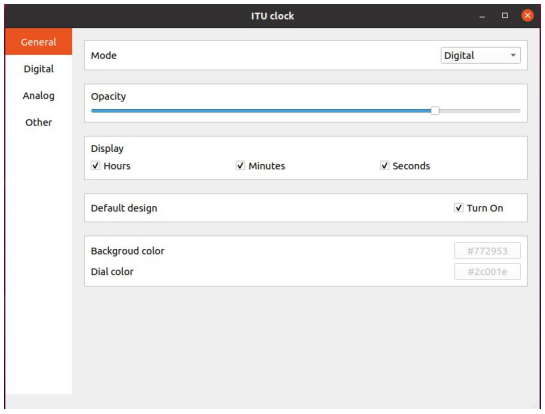
Jako kontroler v mém případě posloužila třída *SettingWindow*, která je přímo napojena view. Má na starosti zprostředkovávat interakci mezi modely a zobrazením nastavení. Při inicializaci nastavuje správné parametry komponent v okně a při změně komponenty promítá změnu do modelu.

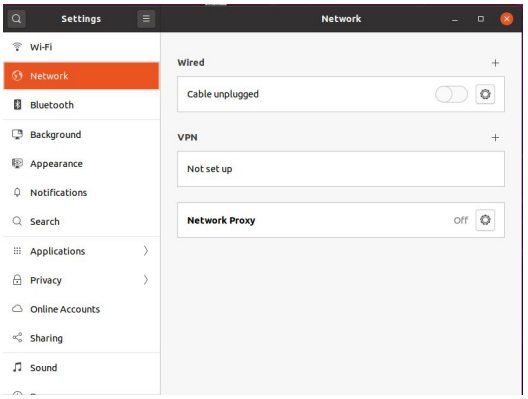
Při mapování Modelu do View jsem se rozhodla nevyužít vestavěnou komponentu *QWidgetMapper*, ale mapování jsem jsem si implementovala vlastní, protože jsem potřebovala větší míru přizpůsobení.

Mapperů jsem implementovala 5, pro každý druh použité komponenty ve view jeden (*QComboBox, QPushButton, QCheckBox, QLineEdit, QSlider*). Jedná se vždy o třídu QMap (slovník implementovaný jako red-black tree), která obsahuje jako klíč ukazatel na komponentu daného druhu a jako hodnotu ukazatel na atribut modelu daného datového typu.

View

K předešlému kontroleru náleží jedno View - *SettingWindow.ui.* To bylo vytvořeno pomocí Designu v QTCreatoru z běžných komponent.
Cílem byla podobnost s nativním nastavením Ubuntu. V levé části je tedy navigační lišta, které přepíná zobrazenou stránku. V ní jsou pak na šedém podkladu konkrétní nastavení. Barvy byly přebrány z oficiálních stránek Ubuntu.





Qt dokáže využít nativní pop-up dialogy nativní k danému OS. Toho jsem využila při dialogu vybírající barvy.

Translator

Jelikož jsme zamýšleli udělat aplikace internacionální, museli jsme přidat jazykové mutace. Ty jsou řešeny pomocí *QTranslator* obsažením ve třídě *Translator*.

Pokud se při prvním spuštění je jazyk operačního systému jeden z námi podporovaných (AJ, CZ, DE), je aplikace přeložena do tohoto jazyka. V opačném případě zůstává v angličtině. Výběr jazyka se dá dále změnit v nastavení.

Překlady byly vytvořeny pomocí nástroje *Qt Linquist*.

Signály a Sloty

Propojení pomocí signálů a slotů je vytvořeno mezi každou komponentou v nastavení a kontrolerem při změně. Aby mohly hodiny reagovat, je zároveň vyslán signál o změně pro hodiny.

Při změně jazyka je také vyslán signál pro *Translator*.

5.2 Použité nástroje a knihovny

Projekt byl implementován v jazyce C++ s využitím knihovny Qt. Qt má docela rozsáhlou dokumentaci, ke které bohužel není moc tutoriálů, nebo jsou minimální. Lehce se dalo ale inspirovat PyQt, který funguje na stejném principu a má více tutoriálů a to i mimo oficiální zdroje.

Z knihovny Qt mě mile překvapily *QTranslator* a *QSettings*, jak jsem zmínila výše. Naopak problém jsem měla s *QDocuWidgetMapper*, jehož obdobu jsem si nakonec implementovala sama.

Jelikož byl projekt implementován v C++ s využitím knihovny Qt, dávalo smysl použít i Qt Creator IDE. Na menší projekty je dostačující, ale přece jen se nevyrovná vývojovým prostředím jako např. sada od JetBrains nebo VisualStudiu. S Qt Linquist se naproti tomu pracovalo dobře.

5.3 Finální testování

Testování opět proběhlo na pokročilém a nepokročilém uživateli Ubuntu.
Oba měli za úkol nastavit hodiny do podobného stavu, jaký byl v přiloženém souboru.
Dále si pak pokročilý uživatel měl nastavit hodiny tak, aby ladily s přiloženými tapetami.

5.4 Vyhodnocení testu

První část zadání splnily úspěšně oba.
Lehký problém byl u zorientování se při popisu a módu ciferníku. Jsou totiž povoleny jen některé kombinace, a to uživatele mátlo. Rozporuplná byla pak funkce přejití do systémového nastavení hodin. Pokročilý uživatel ji ocenil, nepokročilý předpokládal, že bude možné si zde nastavit čas.
Celkově ale oběma přišlo nastavení srozumitelné.

U druhé části neměl pokročilý uživatel problém si nastavit hodiny dle tapety. Doba nastavení byla do pár minut.

Do budoucna by se hodilo lépe formulovat a rozvrhnout popis a mód ciferníku.

Závěr

Nastavení funguje spolehlivě a je použitelné. Drobnosti zmíněné v předchozí kapitole se dají do budoucna ještě vylepšit.

Myslím, že velkou předností jsou jazykové mutace a 2 formáty zobrazení digitálních hodin, díky kterým se rozšiřuje počet potenciálních uživatelů.

Náš tým si docela sedl a spolupráce byla dobrá. Větší oddělení aplikací nakonec mělo jistou výhodu v tom, že jsme si navzájem mohli zkonzultovat problémy.
Na rozdíl od minulého projektu jsme si hned na začátku dobře rozložili úkoly, což dost zjednodušilo postup a implementaci.
V týmu jsem měla na starosti nastavení aplikace, větší část backendu a jazykové mutace, což mi vyhovovalo.

Kdybychom podobnou aplikaci dělali znovu, tak bych se přiklonila k použití C#, nebo alespoň PyQt.

Stručně popište dosažený výsledek a hlavní zjištění z testování. Důležité uživatelské zkušenosti nepopisujte obecně (“uživatelům se to líbilo”), ale co nejvíce konkrétně (co a jak uživatelé použili, zvládli, pochopili, ocenili).
Zakončete krátkým shrnutím Vaší zkušenosti s prací celého týmu a Vaší roli. Co pro Vás bylo při práci v týmu přínosné (2-3 věci) a co byste příště udělali jinak (pouze 1 věc).
Kapitola bude mít číslo pravděpodobně vyšší, podle počtu kapitol s Návrhem GUI a s Implementací (tj. podle počtu členů týmu).

6. Implementace GUI - zobrazování hodin a základní ovládací prvky desktopové aplikace [xhubik03]

6.1 Implementace

Zpracování dat

Princip implementovaného MVC je blíže popsán v předchozí kapitole. Uživatelské rozhraní hodin využívá, zpracovává a zobrazuje data získávaná z dostupných modelů. Aktualizace uživatelských rozhraní je (v Qt obecně i v této implementaci) zajišťována sloty navázanými na příslušné signály o změnách v datech (většinou změnách provedených v uživatelském nastavení).

Hlavní okno

Hlavní okno aplikace je implementováno ve třídě *ClockWindow* (*clockwindow.h, clockwindow.cpp)*, která je potomkem knihovnové třídy *QMainWindow*. Využívá možnosti přizpůsobení vzhledu okna pomocí příznaků, včetně příznaku specifického pro linuxové platformy, který umožňuje vytvořit okno bez klasického rámečku a bez vstupu na liště operačního systému. Protože se jedná o nástroj ve styly desktopového widgetu, jsou tyto vlastnosti žádoucí. Dále jsou zde implementována vlastní tlačítka na zvětšení/zmenšení a zavření okna a události, které padly za vlast kvůlii výše zmíněným protisystémových nastavením. Prvky jsou uspořádány pomocí rozložení podle potomků třídy *QLayout.* Okno obsahuje odkaz na widget analogových a/nebo digitálních hodin.

Analogové hodiny

Jsou implementovány třídou *AnalogClock* (*analogclock.h, analogclock.cpp).* Zobrazují hodiny, zpracovávají a zobrazují změny v nastaveních jako vzhled ciferníku, barvy, apod. Starají se o aktualizaci hodinového widgetu po každé sekundě. Hodiny jsou složeny z designových prvků, které je možné najít společně s ikonami hlavního okna v podadresáří *assets* odevzdaného projektu.

Změny barev probíhají pomocí získání masky měněného prvku podle jeho současné barvy a následné jeho překreslení s pomocí masky na barvu novou. Třída pro tyto účely drží stavové proměnné s hodnotami těchto nastavení před novou změnou.

Digitální hodiny

Implementovány třídou *DigitalClock* (*digitalclock.h, digitalclock.cpp*). Kromě obecných nastavení hodin umožňují také zobrazit čas ve dvanáctihodinovém, či čtyřiadvacetihodinovém formátu. Čas je zobrazován ve stylu LCD displeje s plochými čísly bez vnějších obrysů.

Pozn.: Pro oba typy hodin jsou v příloze ukázané screenshoty, demonstrující i některá rozdílná nastavení.

6.2 Použité nástroje a knihovny

Desktopové hodiny byly kompletně implementovány v C++ s knihovnami *Qt,* za použití IDE *Qt Creator*. Implementace probíhala přímo na zamýšlené cílové platformě.

Přinejmenším část hodin (analogový ciferník) měla být původně implementována deklarativně v jazyce QML a integrována s výše popsaným nastavením implementovaným s využitím *QWidgets*. To mělo být dosaženo možností integrace *Qt Quick* a *QWidget* prvků. Opakovaně se zde však projevovala obskurní chyba SDK a vzhledem k tomu, že k ní nebyla dostupná žádná relevantní dokumentace, bylo nakonec přistoupena k implementaci všech prvků pomocí klasických widgetů.

Pro implementaci digitálních hodin byl jako základ použit widget *QLCDNumber*. Implementace analogových hodin je postavena na prvcích *QGraphicsScene* a *QGraphicsView.*

Pro načítání a zpracování designových elementů (ručičky, ciferník) jsou použity knihovny *QImage* a *QPixmap.* Změny jejich vzhledu vyžadují metody a datové struktury knihoven *QPaint* a *QColor*.

Kromě toho jsou použitý také knihovny *QTime, QTimer, QCursor*, *QRegion* a další, které zajišťují důležité funkcionality či rozvíjejí vzhled aplikace.

Použité nástroje mi v roce 2020 nepřipadaly příliš vhodné, zejména ne pro tvorbu analogových hodin. Domnívám se, že kdyby šlo pouze o zobrazení a nastavování např. hodin digitálních, byly by tyto nástroje zcela bezproblémové, ale v tomto případě snadno docházelo např. ke snížené čitelnosti kódu oproti deklarativním řešením. Nicméně nelituji, že jsem si tyto nástroje vyzkoušel, kromě menší pohodlnosti a rozšiřitelnosti zápisu jsou stále silné.

6.3 Finální testování

Prefinální verze aplikace byla předložena uživateli, který je v užívání stolního počítače středně pokročilý, ale s Ubuntu žádné zkušenosti nemá, a dále staršímu uživateli, který je začátečníkem. Cílem testu bylo vyhodnotit, jestli zobrazení dostupných nastavení odpovídají očekávání uživatelů a jestli si uživatelé dokáží vybrat nastavení, které pro ně bude dobře viditelné a současně ne rušivé. Kromě toho bylo cílem získat obecnou zpětnou vazbu a ověřit intuitivitu základních ovládacích prvků.

6.4 Vyhodnocení testu

Oba uživatelé bez problému pracovali s oknem hodin (přesunutí, minimalizace, zavření) a dokázali otevřít nastavení, ve kterém se rychle zorientovali. Nenarazili na žádné nesmyslné chování, kterému by nerozuměli. V obou případech pracovalo zobrazování podle očekávání.

První uživatel ocenil, že po zmenšení okna se místo hlavních hodin zobrazí decentní štítek s aktuálním časem.

Druhý uživatel měl problém se sledováním času ve výchozím nastavení vzhledu, protože čárky na ciferníku pro něho byly příliš nevýrazné. Nicméně dokázal najít římské a arabské číslice, které už dostačovaly.

Ani jednoho z uživatelů nelákala možnost ukotvit hodiny na místě a oba se shodli, že v kontextu průhledného okna působila rušivým dojmem. Tato funkce byla v důsledku zcela odebrána.
Aby hodiny co nejlépe zapadaly do pracovní plochy, bylo by do budoucna vhodné zbytná tlačítka kolem ciferníku skrývat, pokud na okno uživatel nenajede myší. V takové situaci by ale naopak mohlo v orientaci pomoci dočasné zvýraznění okna (např. pouze poloviční průhlednost). Na hodinách by bylo zejména vhodné zbavit se drobných nedostatků při vykreslování analogového ciferníku.

Závěr

Dokončená aplikace v sobě nezahrnuje všechny původně zamýšlené vlastnosti, některé z nich (např. možnost ukotvení okna) vypadly během uživatelských testování. Zobrazení analogových hodin by mohlo dosahovat mírně vyšší kvality, zejména v případě použití jiné sady nástrojů. Motivace projektu byla nicméně alespoň v podstatných bodech naplněna a výsledkem je jednoduché, přehledné, přizpůsobitelné zobrazení času pro desktop.

Práce na projektu mne osobně bavila, bavila by mne však podstatně více, kdybych ho mohl v kontextu spolupráce s rozhraním uživatelského nastavení implementovat pomocí QML. Práce v týmu podle mého probíhala za rozumných okolností, komunikace atd. byly dostatečné. Osobně jsem nejvíce ocenil přítomnost poskytovat a přijímat zpětnou vazbu.

7. Implementace GUI - aplikace pro iOS [xholas11]

7.1 Implementace

Implementace by se dala rozdělit na tři části: implementaci modelu, widgetu a samotné aplikace.

Model

Z celé aplikace jde přistupovat ke třídě *ClockSettings*, každá její položka odpovídá jedné položce klíč-hodnota z datového modelu. Takto je zajištěn snadný přístup k datům z kteréhokoliv místa aplikace.

Aplikace

Hlavní obrazovku tvoří tabulka s jednotlivými zastaveními. Aplikace je implementována tak, aby se dala případně snadno rozšířit o další možnosti nastavení. Proto je v hlavním *controlleru* popsána struktura tabulky s nastavením, tak aby bylo možné generovat pro každé nastavení příslušnou obrazovku. Detailní vzhled a chování buněk tabulky je doladěn v jim přiřazených souborech s kódem. Pro výběr barev je použit vestavěný *UIColorPickerViewController* z knihovny UIKit.

Widget

Jádrem části aplikace, která obsluhuje widget je takzvaný *Provider*, který se stará o odesílání dat z *modelu* přímo do widgetu každou sekundu, tak aby mohlo dojít k překreslení času. Widget je potom deklarativně implementován, tak aby data zobrazoval jako hodiny. Všechny prvky rozhraní jsou vytvořeny z primitiv a tím pádem jsou počítány a vytvářeny za běhu aplikace. Výpočetní síla widgetu je však systémem omezena a v sekundovém intervalu mezi překreslením widgetu zvládne vykreslit pouze omezený počet objektů. Z toho důvodu jsou značeny na ciferníku analogových hodin implementoány pomoci obrázku s průhledným pozadím, které jsou pomocí masky vykreslovány požadovanou barvou.

7.2 Použité nástroje a knihovny

Pro implementaci byly použity pouze základní nástroje dodávané v programu XCode. Widget využívá knihovnu WidgetKit společně s knihovnou SwiftUI, zbytek aplikace je implementován pomocí knihovny UIKit. Zatímco SwiftUI je nutné napsat ručně, pro UIKit jsem využil grafické prostředí pro tvorbu GUI vestavěné v prostředí XCode. Jedinou další použitou knihovnou je Foundation, která zajišťuje základní funkcionalitu, jedná se tedy o jakousi standartní knihovnu jazyka Swift.

7.3 Finální testování

Finální testování probíhalo na aplikaci běžící na mobilu.
Úkoly se od pilotního testování nezměnily, tedy:

- Změnit barvu dvojteček v digitálních hodinách.
- Změnit mód z digitálního na analogový.
- Zapnout zobrazování sekund.

Současně si několik uživatelů nechalo aplikaci nainstalovanou a používali ji po delší čas. Uživatelé obecně neměli s ovládáním aplikace problémy.

7.4 Vyhodnocení testu

Předpoklad z vyhodnocení pilotních testů, že uživatelé budou aplikaci ovládat snáz pokud se budou moci rovnou podívat na změny ve widgetu, se ukázal jako validní. Ovládání aplikace bylo pro uživatele jednodušší než u pilotních testů. I přesto by bylo ale bylo dobré v budoucnu dodat další lokalizace. Po delším používání se ukázalo, že barvy přidané mezi oblíbené nejsou sdíleny v celé aplikaci, v budoucnu by tedy bylo dobré tuto chybu opravit. Zdaleka nejčastější výtkou byla chybějící možnost nastavit si průhledné/poloprůhledné pozadí widgetu. To ale bohužel není technicky možné, takže nezbývá než doufat, že Apple tuto možnost vývojářům v budoucnu dá. Na základě požadavků uživatelů byla aplikace rozšířena o zobrazování data a časového pásma.
Uživatelé pochopili koncept ovládání aplikace a nečinilo jim potíže používat aplikaci po delší časové období. To že si uživatelé začali přidávat barvy mezi oblíbené mě překvapilo, proto mě ani nenapadlo tuto možnost při vývoji testovat. Celkově ale aplikace splnila očekávání uživatelů.

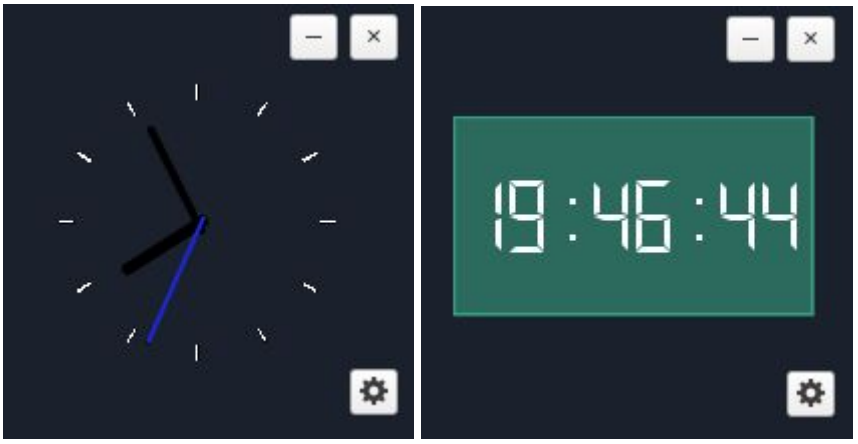
Závěr

V týmu jsem tentokrát pracoval spíše samostatně, s případnými konzultacemi s kolegy, protože naše aplikace kvůli rozdílným platformám neměly moc společných částí. Tato role mi vyhovovala, protože na mě často “padne” role vedoucího týmu, což je při 3 a více souběžných projektech dost vyčerpávající. Celkově náš tým pracoval v rámci časových možností bez problémů. Nicméně do následujícího hodnocení zahrnu zkušenosti i z ostatních projektů.
Pro mě přínosné věci při práci v týmu:

- Možnost konzultovat důležitá rozhodnutí
- Každý člen v týmu může pracovat na tom v čem je dobrý.

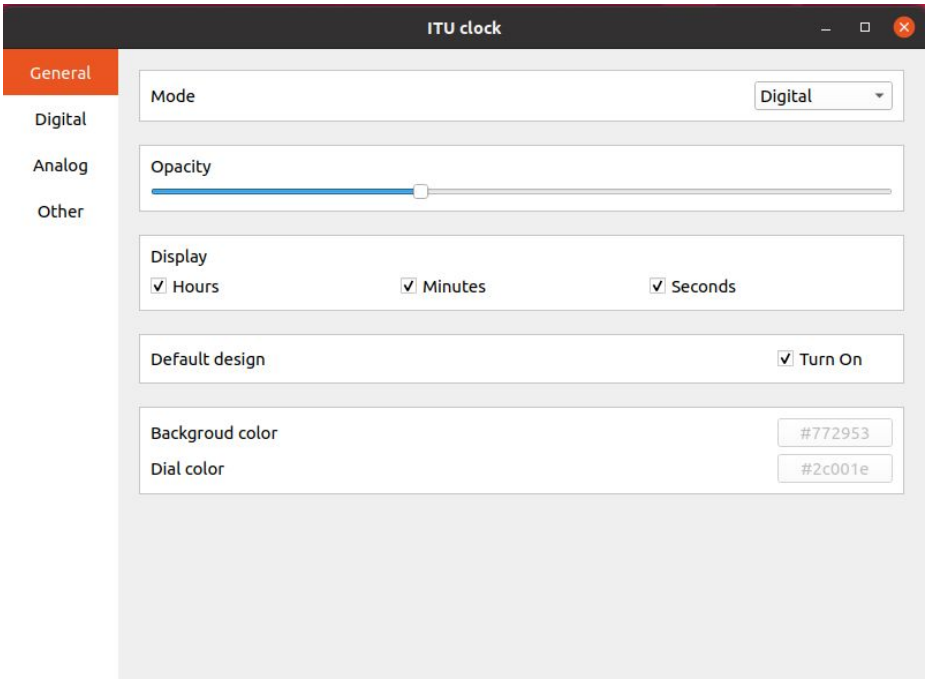
Příště bych se více soustředil na tvorbu uživatelského rozhraní bez jakékoliv funkcionality, tak aby GUI mohlo být komplexnější za cenu toho, že aplikace nebude při odevzdání plně funkční.

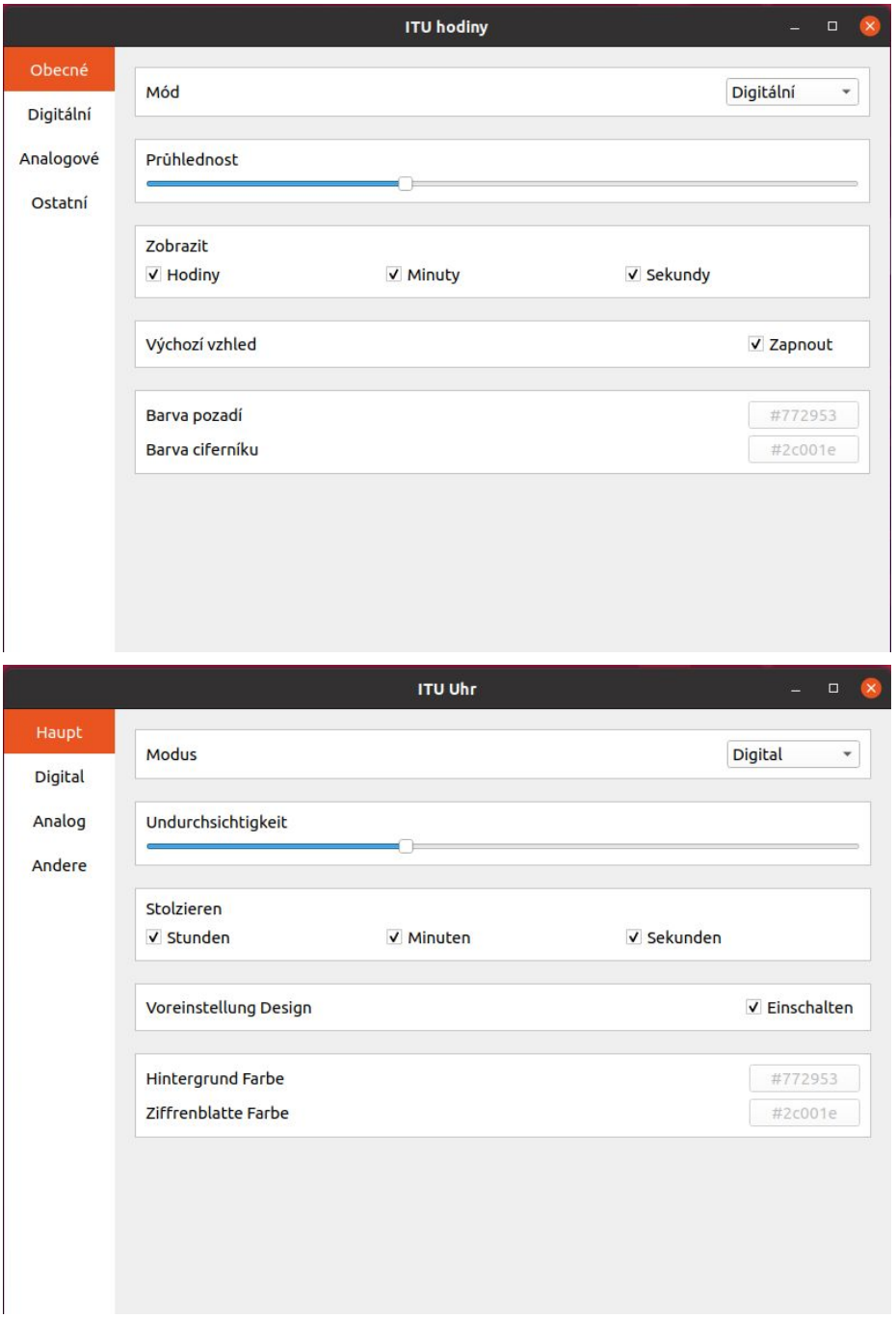
9. Obrázky



analogové a digitální hodiny na tmavém pozadí

analogové a digitální hodiny na Ubuntu pozadi





Anglická, česká a německá mutace nastavení

Reference

https://doc.qt.io/
https://doc.qt.io/qt-5/qtqml-index.html
https://doc.qt.io/qt-5/qdatawidgetmapper.html
https://design.ubuntu.com/brand/colour-palette/
https://doc.qt.io/qt-5/qtlinguist-index.html

https://doc.qt.io/qt-5/qlcdnumber.html
https://doc.qt.io/qt-5/qtime.html
https://doc.qt.io/qt-5/qgraphicsscene.html
https://doc.qt.io/qt-5/qpaintevent.html
https://doc.qt.io/qt-5/qquickwidget.html