Sem vložte zadání Vaší práce.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

MIDI stage piano

Bc. Petr Svoboda

Katedra softwarového inženýrství
Vedoucí práce: Ing. Josef Pavlíček, Ph.D.

7. ledna 2020

Poděkování

Děkuji vedoucímu této práce Ing. Josefovi Pavlíčkovi, Ph.D. a hudebníkům, kteří se zůčastnili uživatelského testování výsledné aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. ledna 2020 .....

Odkaz na tuto práci

Svoboda, Petr. *MIDI stage piano*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem této práce bylo vytvořit prostředí pro živou hudební produkci, které by využívalo virtuálních hudebních nástrojů a zároveň přinášelo výhody čistě hardwarových hudebních nástrojů. Byla provedena rešerše existujících řešení a následně navržen systém s využitím počítače a připojené MIDI klaviatury s prezentačními prvky. Součástí práce je i implementace navrženého řešení, která byla otestována na úrovni uživatelských testů.

Klíčová slova VST, MIDI, OSC

Abstract

The main objective of this thesis is to develop a platform suitable for live music performance combining usage of virtual music instruments and advantages of standalone electronic instruments. After research of possible solutions was made, custom system using computer and external MIDI controller with presentation elements was designed. The system was implemented for real-world usage and evaluated within user testing process.

Keywords VST, MIDI, OSC

Obsah

Seznam obrázků

Úvod

V hudební branži se můžeme setkat s rozličnými hudebními nástroji. V posledních desetiletích zaujaly významné místo na trhu elektronické hudební nástroje. Zároveň se s příchodem dostupných a výkonných výpočetních jednotek začaly objevovat virtuální hudební nástroje[1]. Jedná se o software, který dokáže generovat zvukovou vlnu v reálném čase na základě kofigurovatelných parametrů a vstupů, jako jsou MIDI data, či zvukový signál. Lze tak dosáhnout věrné simulace syntezátorů, varhan a dalších mnoha hudebních nástrojů, včetně efektů. Takové nástroje velmi často implementují některé z dostupných API tak, aby je bylo možné použít v různých prostředích. Dříve byly hlavní doménou virtuálních nástrojů prostředí pro studiovou produkci hudby[2]. V dnešní době, kdy mobilní zařízení dokáží poskytnout dostatečný výkon, stoupá zájem o využití softwarových hudebních nástrojů při živých vystoupeních. Cílem práce je navrhnout takové řešení, které by umělci poskytovalo dostatečné možnosti a komfort pro efektivní využití virtuálních hudebních nástrojů na pódiu.

Kapitola 1

Analýza

1.1 Požadavky

Z praktických zkušeností lze formulovat následující požadavky na řešení, které bude vyhovovat účelu profesionálního živého vystoupení. Jelikož lze požadavky formulovat pouze na obecné úrovni, následuje jejich bodové vyjádření.

Funkční požadavky:

- Ovládací prvky musí být snadno dostupné.
- Ovládací prvky musí být precizně ovladatelné.
- Uživatel musí být schopen určit aktuální stav systému vizuálním vjemem.
- Uživatel může definovat svojí sadu zvuků.
- Uživatel může přepínat zvuky z přednastavené sady.
- Uživatel může ovládat parametry zvuku v reálném čase.
- Uživatel může uložit aktuální zvuk.
- Uživatel může ovládat běžné parametry systému, jako např. hlasitost, transpozice.

Nefunkční požadavky:

- Systém nesmí přestat reagovat.
- Systém nesmí generovat nežádoucí zvuky.
- Systém musí správně ukládat konfiguraci poskytnutou uživatelem.
- Systém nesmí umělce omezovat dlouhou dobou prodlevy.

1.2 Průzkum existujících řešení

1.2.1 Hardwarové nástroje

Mezi hardwarovými hudebními nástroji se lze setkat s různou mírou optimalizace pro živá vystoupení. Zde budeme uvažovat pouze skupinu nástrojů, která je tomuto účelu nejblíže, tzv. stage piana.

Tyto nástroje často poskytují obsáhlou banku zvuků, propracované uživatelské rozhraní a velmi vysokou stabilitu. Pokročilejší modely disponují možností přidat nové zvuky v proprietárním formátu, které může uživatel vytvořit za pomocí speciálního nástroje nebo je získat z online databáze. Některé modely navíc umožňují připojit další klaviaturu přes MIDI rozhraní, to ovšem přináší jistá omezení, například absenci možnosti použít zvuk stejného typu na obou klaviaturách současně, jelikož zařízení disponuje pouze jedním zvukovým modulem pro daný typ zvuku. Zároveň nástroje poskytují bohaté možnosti nastavení běžných parametrů, jako jsou např. hlasitost, transpozice, rozdělení klaviatury do zón, citlivost klaviatury apod.

Typickým zástupcem jsou modely značky Nord (obr. 1.1), které jsou velmi oblíbené pro svoji stabilitu a mnohostrannost, jsou nicméně poměrně nákladnou záležitostí.

Obrázek 1.1: Nord Stage 3 HP

1.2.2 Prostředí typu DAW

Prostředí typu DAW jsou určena převážně pro studiovou práci, tedy nahrávání, střih, míchání a postprodukci zvukového materiálu. K jejich využití je potřeba počítač nebo tablet. Klaviatury jsou připojeny přes MIDI rozhraní. Ke generování zvuku používají virtuální hudební nástroje. V prostředí lze používat více stop, kde každá stopa může používat různý virtuální nástroj a sadu efektů. Poskytuje také bohaté možnosti routování zvukového signálu. Nechybí ani integrace s klaviaturami a kontrolery, což usnadňuje práci a přispívá ke komfortu užívání.

Ikdyž je prostředí typu DAW primárně určeno pro studiovou práci, může být vhodné i pro použití při živých vystoupeních. Jedná se však převážně

o takový typ vystoupení, kdy má umělec předem pečlivě připravené celé představení. Flexibilita prostředí při samotném vystoupení je pak nízká. Lze sice docílit i flexibilního chování, nezbytná je však interakce uživatele s počítačem nebo tabletem. Při živém vystoupení je však taková interakce značně nevhodná, jelikož neposkytuje dostatečnou rychlost a preciznost ovládání. Navíc lze použít pouze omezené množství přednastavených zvuků, jelikož pro každý zvuk musí být zavedena zvláštní instance virtuálního nástroje, která čerpá systémové prostředky.

Obrázek 1.2: Ukázka prostředí Bitwig Studio

1.2.3 Prostředí pro živá vystoupení

S příchodem výkonných mobilních zařízení se začaly objevovat prostředí pro živá vystoupení. Jelikož jde o novou oblast a uživatelská základna není příliš početná, každé řešení přichází s unikátním přístupem. Jasnou výhodou tohoto řešení je svoboda ve volbě zásuvných mdulů (pluginů) představujících virtuální nástroje a efekty. Další velkou výhodou je možnost připojit k systému více jednoduchých MIDI klaviatur. MIDI klaviatury vynikají oproti hardwarovým nástrojům nízkými pořizovacími náklady a nízkou hmotností. K systému využívající prostředí pro živá vystoupení lze připojit jakoukoliv MIDI klaviaturu. Umělec tak může v optimálním případě na zkoušky a vystoupení cestovat pouze s notebookem nebo tabletem, kde má uloženy veškeré zvuky a nastavení.

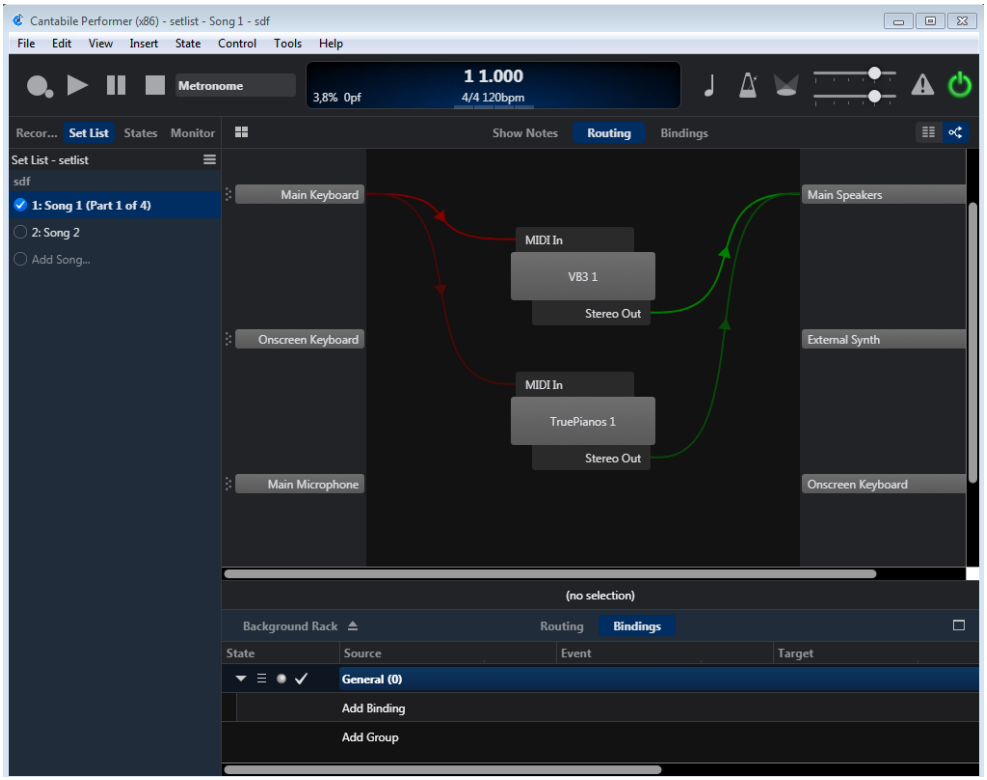
Následuje výčet vlastností, které jsou pro většinu takových řešení společné.

- Snaha o optimalizaci využití systémových prostředků.
- Možnost vkládat vlastní skladby a jejich části.
- Možnost uložit konfiguraci zvuku pro skladbu a její část.
- Přítomnost filtrů, transformací a směrování MIDI zpráv.
- Podpora ovládání prostředí hardwarovým kontrolerem nebo tabletem.
- Multiplatformní podpora
- Podpora VST/VSTi API

V následující podkapitole bude zkoumán přední představitel prostředí pro živá vystoupení, software Cantabile od společnosti Topten Software.

1.2.3.1 Prostředí Cantabile

Software Cantabile od společnosti Topten Software představuje robustní řešení. Implementuje pokročilé optimalizace využití systémových prostředků a disponuje širokými možnostmi konfigurace, ovládání však není příliš intuitivní (viz obr. 1.3). Prostředí umožňuje použití více klaviatur, jejich rozdělení na zóny a poskytuje bohaté možnosti pro směrování a transformaci MIDI zpráv, což zajišťuje vysokou flexibilitu. Způsob ukládání nastavení VST pluginů poskytuje více strategií. Cantabile se opírá o model uložení konfigurace pro uživatelem zadanou skladbu nebo její část. Integrace s hardwarovými kontrolery je z pohledu vstupu na vysoké úrovni, ovládacím prvkům lze přiřadit jakoukoliv funkci programu. Z hlediska výstupu je však integrace omezená na obecnou úroveň, pro efektivní použití je třeba používat zároveň tablet nebo počítač.



Obrázek 1.3: Ukázka prostředí Cantabile

1.3 Závěr

Z provedeného průzkumu je patrné, že existují různé možnoti řešení. Každé z nich řeší problém různým způsobem a má své výhody a nevýhody. Nebylo však nalezeno řešení, které by přinášelo možnost využití virtuálních hudebních nástrojů a integraci s hardwarovým kontrolerem tak, aby bylo dosaženo chování hardwarových hudebních nástrojů. Proto byla zvolena cesta návrhu vlastního řešení, které kombinuje přístup prostředí určených pro živá vystoupení a chování hardwarových nástrojů.

Kapitola 2

Návrh vlastního řešení

2.1 Architektura

Jelikož by bylo zbytečné realizovat celé řešení, bylo použito vzájemné propojení více technologií tak, aby vlastní implementace řešila pouze problémy přímo související s dosažením kýženého chování systému. V následujících podkapitolách budou rozebrány jednotlivé části architektury.

2.1.1 VST host

Každý virtuální hudební nástroj podporující VST API, tzv. VST plugin, potřebuje ke svému běhu hostitelské prostředí, tzv. VST host aplikaci, která API implementuje[3].

VST API umožňuje pluginům generovat zvukovou vlnu na základě konfigurovatelných parametrů a vstupních dat, jako jsou MIDI události, či zvukový signál. Konfigurace parametrů virtuálního nástroje probíhá přes vlastní uživatelské rozhraní pluginu nebo přes hostitelskou aplikaci. Každý VST plugin poskytuje hostitelské aplikaci výčet dostupných parametrů a jejich vlastnosti. Zároveň je připravena i podpora tzv. presetů, které reflektují stav pluginu. Presety lze ovládat též prostřednictvím hostitelské aplikace.

Hostitelská aplikace zároveň zajišťuje přenos zvukového signálu z pluginu, např. do zvukového rozhraní zařízení, a přenos MIDI zpráv do pluginu, např. z připojené klaviatury či kontroleru. Hostitelská aplikace musí vykazovat vysokou stabilitu, aby nedocházelo k výpadkům ve streamu zvukového signálu či k zahazování MIDI zpráv. Oblíbenou technologií používanou k implementaci ovladačů zvukových zařízení při zachování nízké latence je technologie ASIO[4].

Jelikož by byla implementace funkcí hostitelského prostředí náročnou záležitostí, byl zvolen přístup využití již existující aplikace, která bude plnit úlohu VST hostitelského prostředí a bude ovládána externě naší vlastní aplikací.

2.1.2 OSC

OSC představuje obecný a jednoduchý protokol určený pro přenos zpráv mezi multimediálními zařízeními a aplikacemi[5]. Obsah zprávy je složen z adresy ve tvaru řetězce a argumentů několika typů. Zprávy jsou přenášeny typicky přes UDP protokol. Multimediální aplikace často poskytují možnost oboustranné integrace se službami třetích stran právě přes OSC protokol.

2.1.3 HW kontroler

Hardwarový kontroler slouží k ovládání multimediálních aplikací nebo zařízení. V naprosté většině případů k přenosu používá MIDI protokol a bývá připojen přes rozhraní USB nebo MIDI. Kontroler navíc může být vybaven prezentačními prvky, jako jsou LED indikátory, displej apod.

2.1.4 Automap

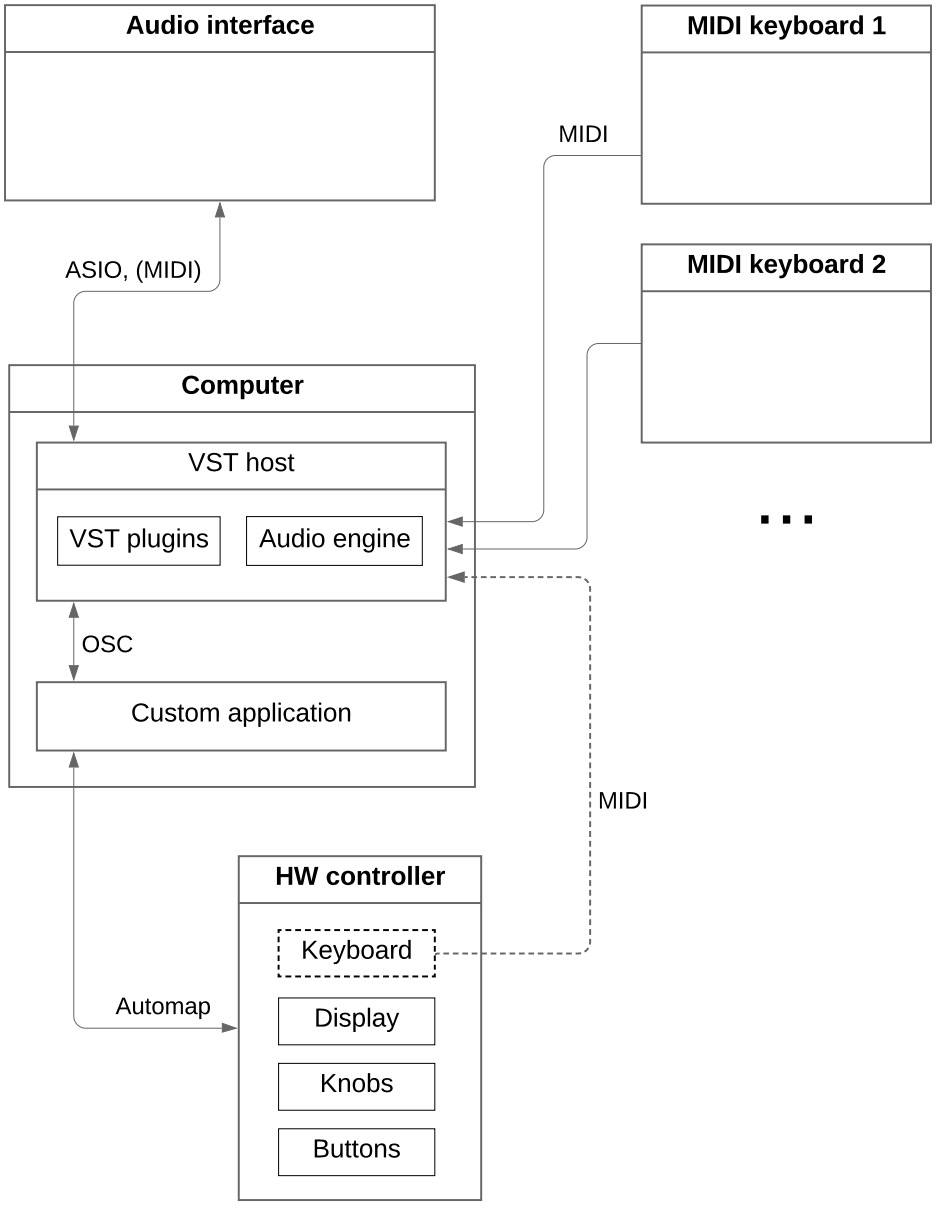
Protokol Automap od společnosti Novation je postavený nad protokolem MIDI, využívá kombinaci zpráv typu CC a SysEx [6][7]. Slouží k integraci multimediálních aplikací se skupinou kontrolerů od společnosti Novation. Existují implementace pro přední DAW prostředí.

2.1.5 Vlastní aplikace

Cílem vlastní aplikace je propojení všech komponent tak, aby bylo dosaženo požadované funkcionality. Aplikace bude ovládat VST hostitelské prostředí a v něm obsažené nástroje a efekty s využitím OSC protokolu. S hardwarovým kontrolerem bude aplikace obousměrně integrována přes protokol Automap. K dispozici bude též desktopové uživatelské rozhraní sloužící ke konfiguraci zvuků, presetů a dalších nastavení.

2.1.6 Závěr

Na diagramu 2.1 je zobrazena architektura systému.



Obrázek 2.1: Architektura řešení

2.2 Analýza kompatibility

Na počátku byla provedena analýza kompatibility jednotlivých komponent.

2.2.1 HW kontroler

Při analýze kompatibility kontroleru za použití modelu Novation SLMkII 49 bylo zjištěno, že kontroler připojený přes USB sice exportuje rozhraní pro komunikaci přes protokol Automap, ale toto rozhraní nehlásí systému příslušnou třídu USB zařízení. Podle zachycené sériové komunikace se ukázalo, že formát přenášených dat přesto odpovídá třídě USB pro MIDI zařízení[8].

2.2.2 VST host

Jako hostitelská aplikace pro VST pluginy byl testován program Bidule od společnosti Plogue. Z anlýzy vyplynulo, že poskytuje bohaté možnosti ovládání přes kombinaci protokolů OSC a UDP. Program přes OSC umožňuje spouštět důležité akce, lze navíc přijímat a odesílat hodnoty všech parametrů VST pluginů a interních modulů. Data jsou uložena ve formátu XML, ze kterého lze snadno získat veškeré informace o uloženém projektu.

2.3 Zvolené technologie

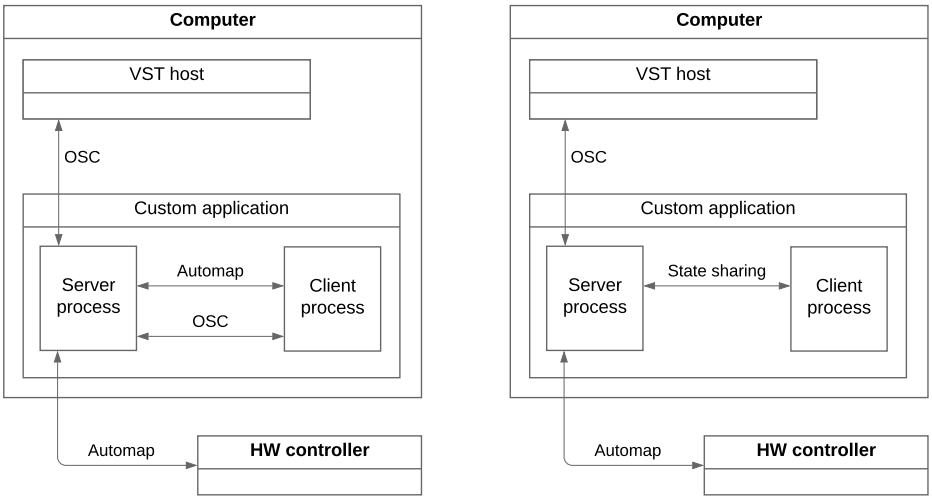
Na základě analýzy kompatibility byl za hardwarový kontroler zvolen model Novation SLMkII 49, jako VST hostitelskou aplikací byl zvolen program Bidule od společnosti Plogue. Pro zajištění nízkoúrovňové komunikace s kontrolerem byl zvolen univerzální USB ovladač WinUSB pro platformu Windows.

Při výběru technologií pro implementaci vlastní aplikace byly zohledněny následující požadavky:

- Možnost low-level komunikace přes USB
- Podpora komunikace přes UDP
- Podpora asynchronního kódu na vysoké úrovni
- Nenáročná tvorba UI
- Stabilita a rychlost odezvy

Pro implementaci vlastní aplikace byl zvolen programovací jazyk TypeScript, jehož kód je kompilován do jazyka JavaScript.

Dále byl zvolen framework Electron, který poskytuje možnost tvorby desktopových aplikací za použití jazyka JavaScript. Při spuštění aplikace vytvořené ve frameworku Electron je spuštěn na pozadí tzv. hlavní proces. Hlavní proces vytváří další vedlejší nezávislé procesy, které slouží k zobrazení oken s



(a) Původní návrh (b) Optimalizovaný návrh

Obrázek 2.2: Srovnání původního a konečného návrhu architektury aplikace.

webovým prohlížečem, kde je načtena klientská část aplikace. Ke komunikaci mezi hlavním a vedlejšími procesy framework poskytuje jednoduché rozhraní.

Komunikaci mezi aplikací a hardwarovým kontrolerem lze implementovat pouze v hlavním procesu. Původním záměrem bylo tedy využití architektonického vzoru tlustý klient. Hlavní proces by fungoval pouze jako proxy pro protokoly OSC a Automap mezi příslušnými komponentami a klientskou částí aplikace (diagram 2.2a). Při testování bylo zjištěno, že rozhraní pro komunikaci mezi hlavním a vedlejšími vlákny není pro tento účel dostatečně rychlé a nemá dostatečně rychlou odezvu. Jako řešení tohoto problému bylo zvoleno rozdělení aplikace na serverovou a klientskou část, kde mezi oběmi stranami nebude docházet k objemné komunikaci citlivé na dobu odezvy. Jelikož lze předpokládat, že velká část stavu aplikace bude sdílená serverovou i klientskou částí, byl vybrán pro obě části framework pro vývoj webových aplikací Angular a knihovna NGXS představující framework pro řízení stavu podle patternu CQRS[9], který vychází ze vzoru CQS[10]. Výhodou je pak možnost automatické synchronizace stavu mezi serverovou a klientskou částí (diagram 2.2b).

Pro klientskou část byla použita knihovna `Angular Material UI`, která poskytuje základní prvky pro tvorbu uživatelského rozhraní.

Pro servrovou část byla využita knihovna `tessel/node-usb`, jejíž úlohou je zpřístupnění funkcí knihovny `libusb`, která poskytuje možnost low-level komunikace s USB zařízeními[11]. V serverové části též byla využita knihovna `node-osc`, která umožňuje komunikaci přes OSC a UDP protokol.

2.4 Funkční požadavky

Při návrhu řešení byly určeny následující funkční požadavky.

V aplikaci bude možné používat více projektů programu Bidule. Bude k dispozici rozhraní pro správu projektů, kde bude možné zadat cestu k uloženému projektu a název projektu. Je možné načíst a zavřít projekt. Veškerá nastavení kromě globálních se budou ukládat pro každý projekt zvlášť. Po načtení projektu dojde k automatickému importu konfigurace projektu Bidule.

Vrstvou se rozumí skupina VST pluginů definovaná v projektu programu Bidule. Manuálem se rozumí skupina vrstev a představuje připojenou klaviaturu.

V rámci aplikace bude možné zobrazit seznam dostupných nástrojů a efektů. Seznam nástrojů a efektů bude importován z konfigurace projektu Bidule. U nástrojů a efektů bude možné vytvořit snímek všech parametrů pluginu, který bude považován za výchozí a bude načten při načtení projektu nebo po přepnutí zvuku. V rámci obou seznamů bude též možné otevřít uživatelské rozhraní VST pluginu, či zobrazit jeho dostupnost v jednotlivých vrstvách. Pro nástroje bude možnost definovat více pojmenovaných stránek mapování, pro efekt bude možné zadat právě jednu stránku mapování s volbou hlavního mapování.

Editor stránek mapování bude poskytovat možnost konfigurace mapování parametrů VST pluginu na enkodéry kontroleru. V rámci jedné stánky mapování bude možné ke každému enkodéru přiřadit právě jeden parametr VST pluginu a zadat tak až 8 různých mapování. Při volbě parametru pluginu bude možné použít tzv. funkce learn, kdy lze ovládaný parametr vybrat přímo v uživatelském rozhraní pluginu. Bude možné zadat název mapování. Pro každé mapování budou k dispozici následující strategie:

- **Lineární** - Tato strategie bude umožňovat určit lineární vztah mezi hodnotou enkodéru na hardwarovém kontroleru a hodnotou parametru příslušného pluginu. Bude možné zadat rozsah hodnot zobrazených na displeji, rozsah hodnot parametru VST pluginu, invertovat průběh a zadat suffix pro zobrazení na displeji. Rozsah hodnot parametrů pluginu bude možné nastavovat i přes tzv. learn metodu, která umožňuje hodnoty zvolit přímo v prostředí VST pluginu. Na LED prstencích rotačních enkodérů kontroleru bude hodnota parametru indikována lineárně na celém rozsahu enkodéru.

- **Lineární s položkovým zobrazením** - Jedná se o rozšíření lineární strategie s možností zadání položek, které budou zobrazeny na displeji v závislosti na hodnotě parametru pluginu. Každá položka bude určena názvem a hraniční hodnotou parametru VST pluginu. Položky bude možné přidávat i za pomoci tzv. learn metody, kdy je možné zadat hraniční hodnotu přímo v prostředí VST pluginu.

- **Seznam** - Za použití této strategie bude možné definovat seznam položek obsahujících název a hodnotu parametru VST pluginu. Oproti lineární strategii s položkovým zobrazením bude probíhat změna hodnoty parametru VST pluginu skokově. Při reflektování změn hodnoty parametru bude zobrazena položka, která je aktuální hodnotě parametru nejblíže.

V editoru presetů bude možné zadat až 8 kategorií zvuků, které bude možné řadit a přejmenovat. Pro každou kategorii bude možné přidat nový preset, či duplikovat stávající. Preset určuje použitý VST plugin, strategii inicializace pluginu vč. její konfigurace, rozdílovou sadu hodnot parametrů pluginu a rozdílovou sadu hodnot parametrů efektů vrstvy. Rozdílová sada bude pro danou vrstvu vytvářena při změně v parametrech pluginu v době, kdy je preset aktivní. Zaznamenanou sadu bude možné pro aktivní preset uložit. V rámci inicializace VST pluginu budou k dispozici následující strategie:

- **Snímek všech parametrů** - V této strategii bude možné pořídit snímek hodnot všech parametrů VST pluginu. Snímek bude načten při načtení presetu.

- **Zpráva typu Program Change** - Tato strategie zajistí odeslání MIDI zprávy typu Program Change do patřičného VST pluginu.

- **Načtení interního VST presetu** - Při použití této strategie dojde k načtení interního presetu pluginu definovaného VST protokolem. Interní presety pluginu je možné editovat v hostitelské aplikaci či přímo v pluginu.

Pro každou vrstvu může být aktivní právě jeden preset. Po načtení presetu bude pro danou vrstvu aplikovaná inicizalizační strategie na virtuální nástroj a budou načteny rozdílové sady hodnot parametrů nástroje a efektů. Vlastnosti presetu bude možné upravit, či preset odstranit.

Za použití kontroleru bude možný výběr kategorie a stránkovaný výběr presetu v rámci dané kategorie. Zároveň bude k dispozici historie posledních použitých presetů vč. jejich stavů pro každou vrstvu. Hodnoty parametrů pluginů bude možné ovládat rotačními enkodéry kontroleru. Hodnoty budou indikovány LED prstencem enkodéru a grafickým výstupem na displeji podle konfigurace strategie mapování. V rámci ovládání parametrů nástrojů bude k dispozici zobrazení stránky mapování daného presetu. Pro ovládání parametrů efektů bude k dispozici agregované zobrazení, kde bude zobrazena hlavní položka stránky mapování pro každý efekt, který je v dané vrstvě dostupný. Bude možné ovládat i všechny parametry efektu. Kontroler uživateli umožní přepínat a indikovat právě ovládanou vrstvu. Za pomocí posuvných prvků kontroleru bude možné ovládat hlasitosti jednotlivých vrstev. K dispozici bude též ovládání hlavní hlasitosti a tlačítko pro ukončení všech znějících tónů.

2.5 Doménový model

Z analýzy požadavků byl navržen doménový model (viz obr. 2.3), který zobrazuje entity problémové domény a jejich vzájemné vztahy.

2.6 Struktura projektu Bidule

Aby mohly být projekty uložené v programu Bidule načteny ve vlastní aplikaci, je třeba definovat strukturu projektů. Projekt vytvořený v programu Bidule se skládá z modulů, které jsou uspořádány do stromové struktury a jsou vzájemně propojeny. Modulem může být např. VST plugin, vstupně-výstupní rozhraní pro audio nebo MIDI. Program Bidule též poskytuje množství interních modulů. Moduly lze agregovat do samostatných celků a vytvářet tak vlastní znovupoužitelné moduly. V následujících podkapitolách budou rozebrány jednotlivé části architekury projektu programu Bidule. Příklad možné architektury lze vidět na diagramu 2.4.

2.6.1 Modul manuálu

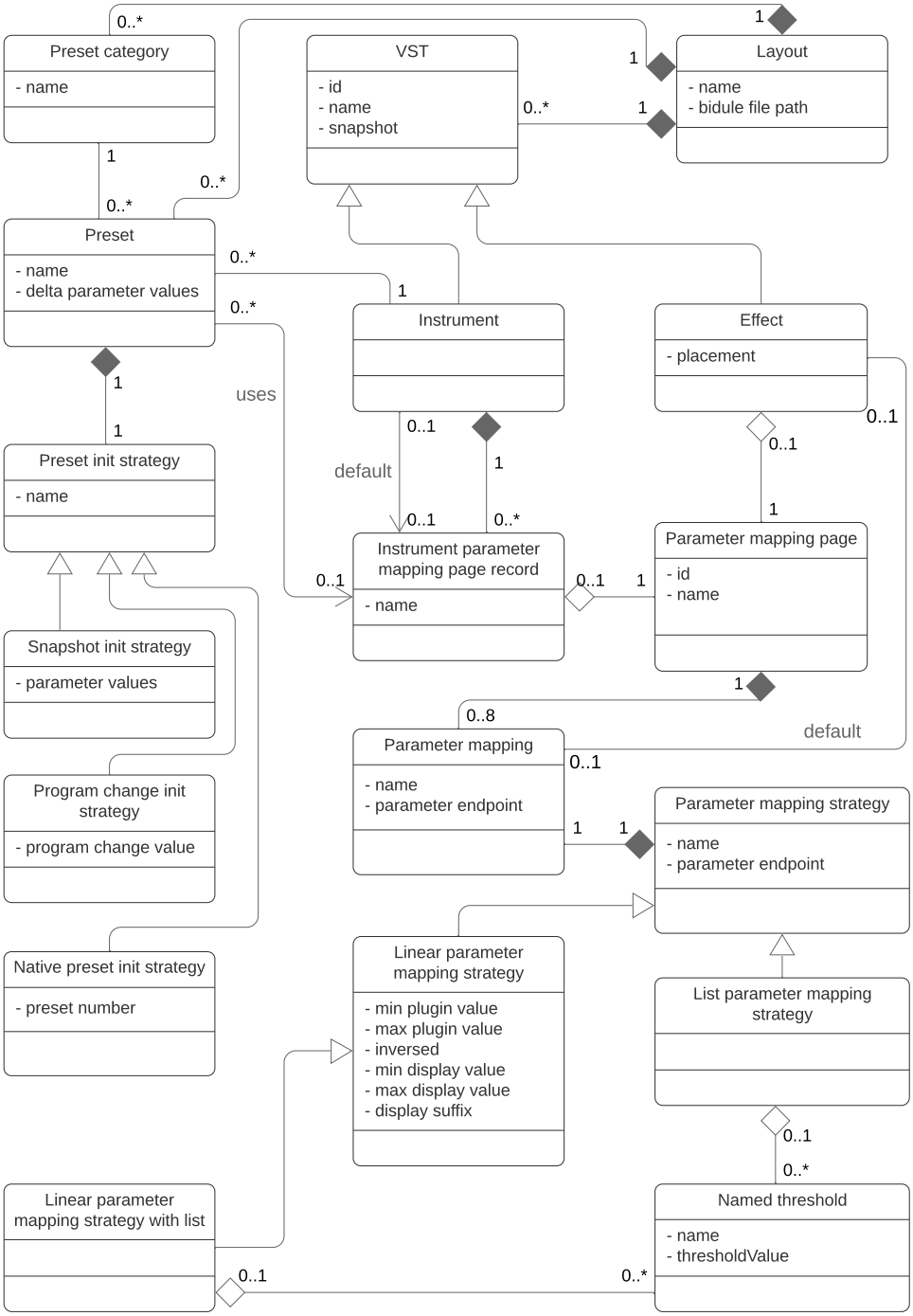
Modul manuálu se skládá ze skupiny vrstev (viz kapitolu 2.6.2) a modulu pro zajištění smíchání zvukového signálu jednotlivých vrstev. Představuje konfiguraci dostupnou pro připojenou klaviaturu. Každá klaviatura může mít různý počet vrstev. Doporučeno je však použití stejné konfigurace pro všechny manuály kvůli snazší orientaci uživatele. V projektu může být zadán jeden až čtyři manuály.

2.6.2 Modul vrstvy

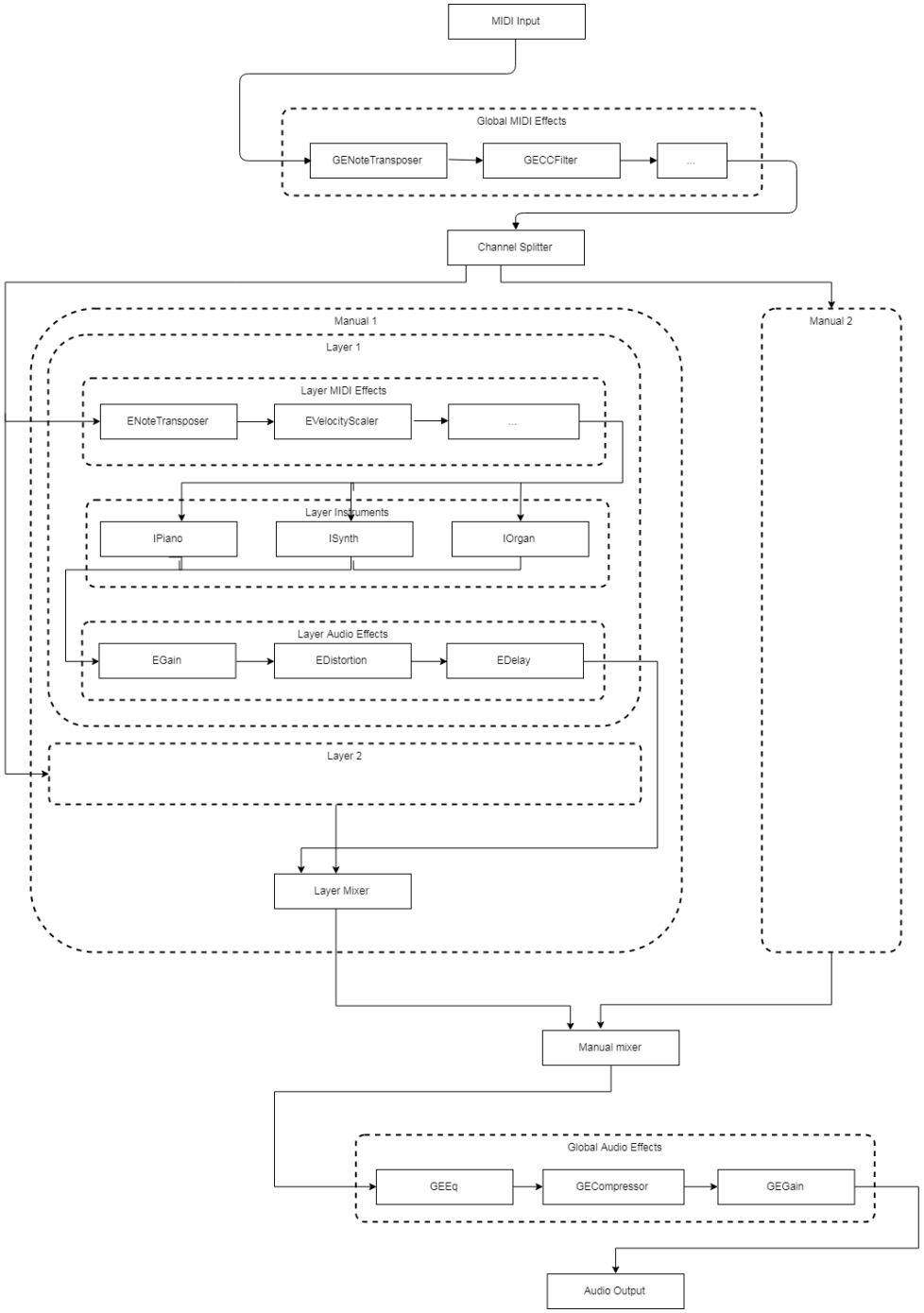
Modul vrstvy představuje sadu vstupních efektů, virtuálních nástrojů a výstupních efektů. Skupina vrstev je součástí modulu manuálu (viz kapitolu 2.6.1). Manuál může obsahovat jednu až čtyři vrstvy. Každá vrstva může obsahovat různé moduly. Pro jednodušší orientaci uživatele je však doporučeno uchovat všehcny dostupné vrstvy ve stejné konfiguraci, a to i mezi jednotlivými manuály.

2.6.3 Globální efekty

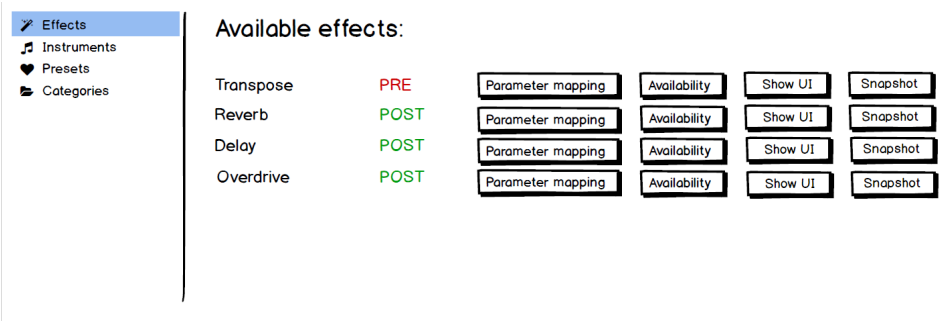
V projektu je možné definovat vstupní a výstupní efekty, které budou k dispozici nezávisle na vrstvě a budou aplikovány na veškerá vstupní a výstupní multimediální data. Příkladem takového efektu může být transpozice tónů nebo ekvalizace zvukového výstupu. Vstupní a výstupní efekty jsou rozděleny do dvou skupin. Projekt může obsahovat až 8 vstupních a 8 výstupních efektů.



Obrázek 2.3: Doménový model



Obrázek 2.4: Ukázka architektury projektu Bidule



Obrázek 2.5: Drátěný model obrazovky `D6` (seznam efektů)

2.6.4 Vstupně-výstupní rozhraní

V projektu musí být k dispozici alespoň jedno MIDI vstupní zařízení a jedno zvukové vstupně-výstupní zařízení. Vstup z MIDI zařízení je směrován do skupiny komponent představujících globálních MIDI efekty. Výstup ze skupiny komponent představující globální zvukové efekty je pak směrován do výstupního zvukového zařízení. V případě potřeby lze také směrovat vstup ze zvukového zařízení do nástrojů a efektů, které podporují zpracování zvukového signálu.

2.6.5 Propojení modulů

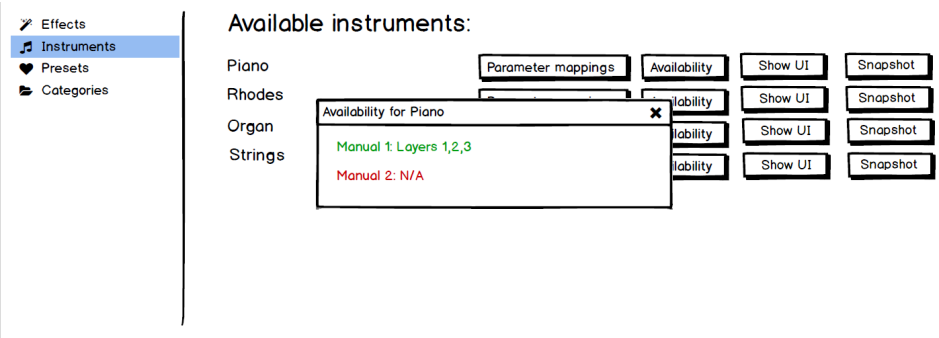
Jelikož musí být zachována variabilita směrování MIDI zpráv, celý systém pracuje pouze s jedním zdrojem MIDI zpráv. Každá MIDI zpráva má určené číslo kanálu, které může nabývat 16 různých hodnot. U zdrojových MIDI zprávy lze v programu Bidule libovolně přemapovat kanál pomocí interních modulů, např. na základě rozsahu klaviatury, připojeného manuálu apod. Podle kanálu zprávy je pak zpráva směrována do příslušného modulu manuálu. Smíchání zvukového signálu ze všech modulů manuálu je zajištěno interním modulem programu Bidule.

Celé propojení jednotlivých modulů lze vidět na obr. 2.4.

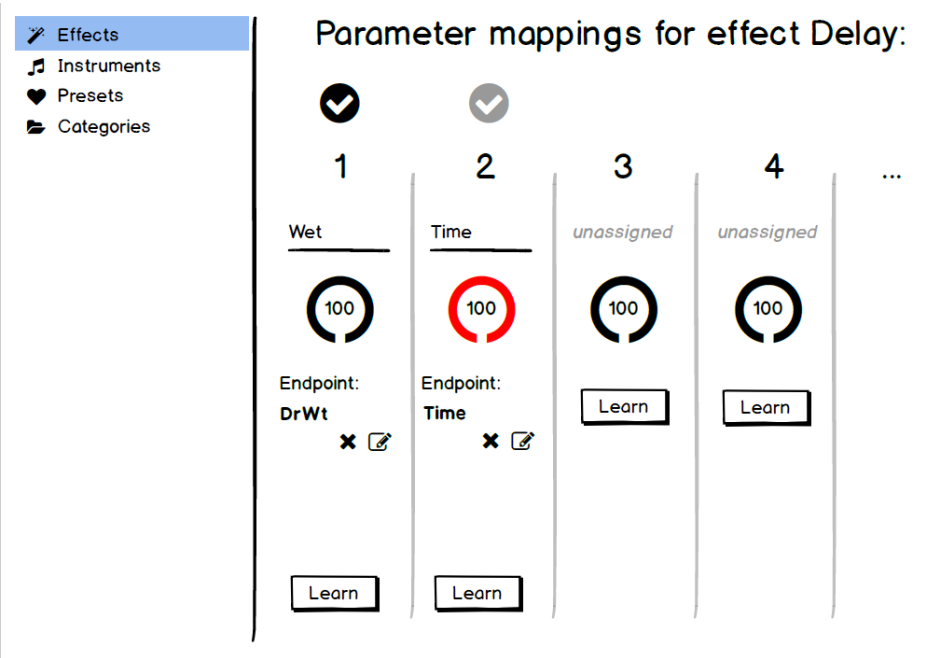
2.7 Desktopové uživatelské rozhraní

Při návrhu uživatelského rozhraní desktopové aplikace byl sestrojen následující seznam obrazovek.

Pro vybrané obrazovky byly sestrojeny drátěné modely (obr. 2.5, 2.6, 2.7, 2.8).



Obrázek 2.6: Drátěný model obrazovky `D4` (zobrazení dostupnosti nástroje)



Obrázek 2.7: Drátěný model obrazovky `D8` (správce mapování parametrů pluginu pro efekty)

D1. Seznam projektů

D2. Editace projektu

D3. Seznam nástrojů

D4. Zobrazení dostupnosti nástroje (obr. 2.6)

D5. Správce mapování parametrů pluginu pro nástroje (obr. 2.8)

D6. Seznam efektů (obr. 2.5)

D7. Zobrazení dostupnosti efektu

D8. Správce mapování parametrů pluginu pro efekty (obr. 2.7)

D9. Správce presetů

D10. Editace presetu

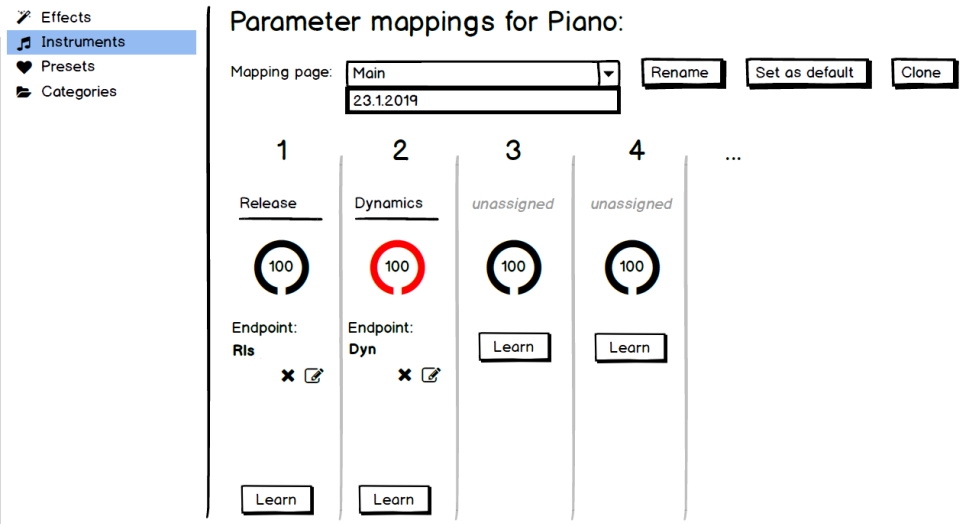
Seznam 2.1: Seznam obrazovek desktopového uživatelského rozhraní aplikace

2.8 Uživatelské rozhraní kontroleru

Jelikož byl v řešení vybrán hardwarový kontroler podporující protokol Automap, bylo navrženo uživatelské rozhraní pro celou skupinu kontrolerů společnosti Novation podporující tento protokol. Všechna taková zařízení disponují téměř shodným rozvržením prvků (obr. 2.9).

Obrázek 2.9: Ovládací panel zařízení od společnosti Notvation podporující protokol Automap.

V rámci návrhu uživatelského rozhraní pro hardwarový kontroler byl kladen důraz na oblast ovládacích prvků umístěných kolem displeje. Seznam 2.2 shrnuje klíčové obrazovky hardwarového kontroleru. Jednotlivé obrazovky budou diskutovány v dalších podkapitolách.



Obrázek 2.8: Drátěný model obrazovky `D5` (správce mapování parametrů pluginu pro nástroje)

C1. Procházení presetů (obr. 2.10)

C2. Ovládání parametrů nástroje (obr. 2.11)

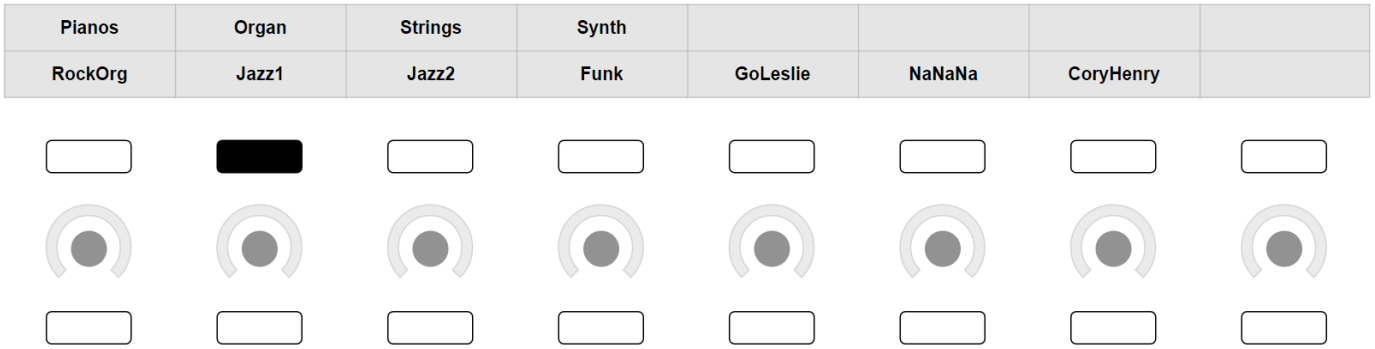
C3. Souhrnné ovládání efektů (obr. 2.12)

C4. Ovládání parametrů efektu (obr. 2.13)

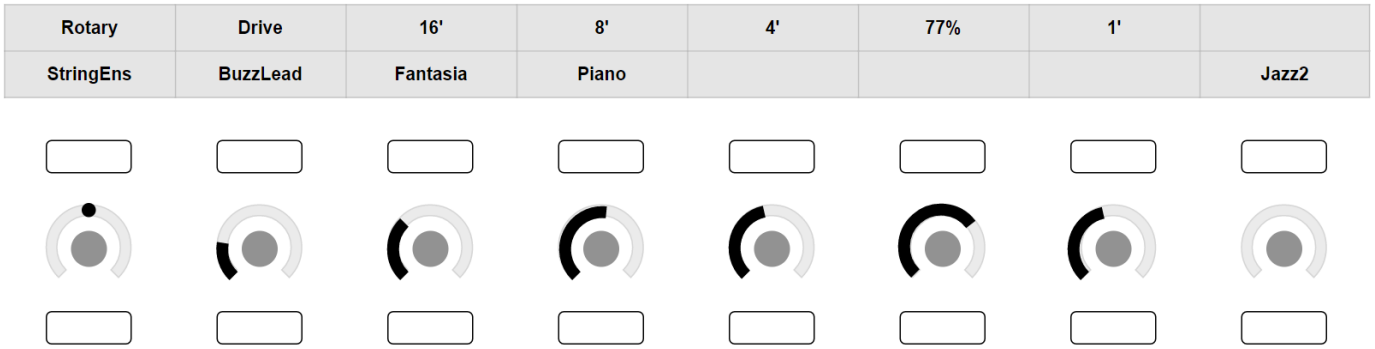
Seznam 2.2: Seznam obrazovek desktopového uživatelského rozhraní aplikace

2.8.1 Hlavní navigace

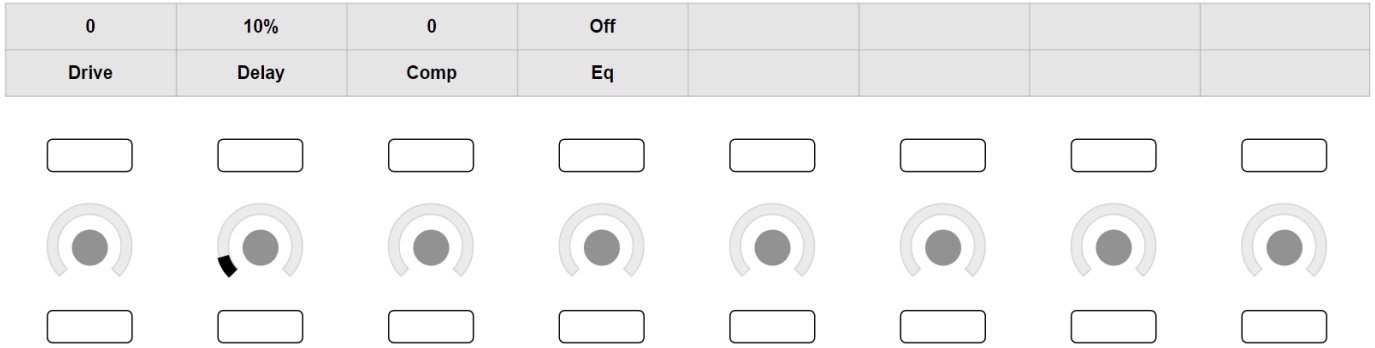
Pro hlavní navigaci mezi obrazovkami kontroleru byla zvolena dvojice tlačítek umístěných nalevo od displeje. Jednoduchým stisknutím spodního tlačítka dojde k načtení obrazovky ovládání parametrů nástroje. Pokud uživatel stiskne vrchní tlačítko, je zobrazena obrazovka procházení presetů. Při dvojitém stisknutí jednoho z tlačítek dojde k načtení obrazovky souhrnného ovládání efektů. Jestliže bylo dvakrát stisknuto horní z tlačítek, budou zobrazeny vstupní efekty vrstvy, v případě spodního tlačítka bude zobrazeno ovládání výstupních efektů pro danou vrstvu. Při trojitém stisknutí se dvojice tlačítek chová podobně jako v případě dvojitého stisknutí, k ovládání efektů ale nedochází na aktivní vrstvě, ale na globální úrovni. Volba aktivní obrazovky je indikována pomocí LED podsvícení příslušného tlačítka. V případě obrazovek, které lze aktivovat vícenásobným stisknutím tlačítka je stav indikován rychlostí blikání LED prvku.



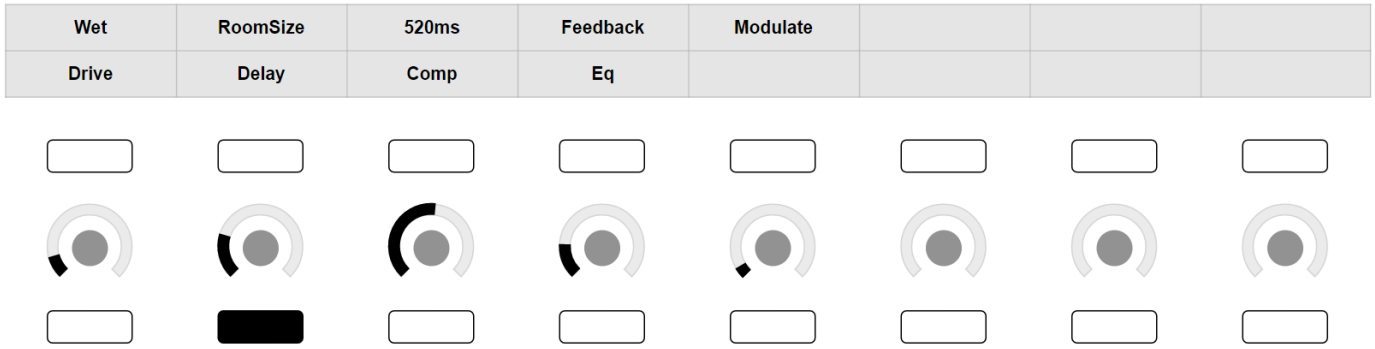
Obrázek 2.10: Drátěný model obrazovky `C1` (procházení presetů)



Obrázek 2.11: Drátěný model obrazovky `C2` (ovládání parametrů nástroje)



Obrázek 2.12: Drátěný model obrazovky `C3` (souhrnné ovládání efektů)



Obrázek 2.13: Drátěný model obrazovky `C4` (ovládání parametrů efektu)

Pro výběr aktuálního manuálu a vrstvy byla zvolena vertikální řada tlačítek pod dvojicí tlačítek sloužicích k hlavní navigaci. Tlačítka představují jednotlivé manuály. Stisknutím tlačítka dojde k přepnutí na vrstvu manuálu určenou podle počtu stisknutí. Aktuální manuál je indikován LED prvkem příslušného tlačítka. Rychlost blikání indikuje pořadí zvolené vrstvy, přičemž konstantní rozsvícení LED značí volbu první vrstvy.

2.8.2 Procházení presetů (obrazovka C1)

Jak lze vidět z drátěného modelu 2.10, na obrazovce procházení presetů jsou na prvním řádku displeje zobrazeny kategorie presetů, na druhém řádku je zobrazen seznam presetů v dané kategorii. Jelikož množství presetů v rámci jedné kategorie může být značné, je uplatněno stránkování presetů. Pro volbu kategorie a její stránky slouží první řada tlačítek umístěných pod displejem. Při stisku tlačítka dojde k volbě příslušné kategorie. Počet stisknutí pak vyjadřuje pořadí stránky presetů, která má být zobrazena.

Na druhém řádku displeje jsou zobrazeny presety pro danou stránku kategorie. Při kliknutí na tlačítko z druhé řady tlačítek pod displejem dojde k výběru aktivního presetu.

Aktuální kategorie a preset jsou indikovány rozsvícenou LED u příslušného tlačítka.

Po aktivování obrazovky je automaticky indikován aktivní preset pro aktuální vrstvu a jeho zařazení v kategorii.

2.8.3 Ovládání parametrů nástroje (obrazovka C2)

Na této obrazovce lze ovládat parametry nástroje a vybrat aktivní preset z historie nedávno aktivovaných presetů (viz diagram 2.11). V prvním řádku displeje je možné vidět název mapování parametru nástroje nebo jeho aktuální hodnotu. V druhém řádku je zobrazena historie nedávno použitých presetů. V posledním poli druhého řádku displeje je pak zobrazen název aktuálního presetu.

Aktuální stav hodnoty parametru nástroje je indikován nepřetržitě LED prstencem umístěným okolo otočného enkodéru na základě zvolené strategie mapování. Při změně parametru nástroje je hodnota zobrazena i graficky v příslušném poli na displeji, jinak je v poli zobrazen název mapování parametru. Formát zobrazení hodnoty parametru na displeji je opět určen strategií mapování parametru.

Při stisku tlačítka z první řady pod displejem dojde k nastavení hodnoty parametru nástroje na výchozí hodnotu určenou aktuálně zvoleným presetem. Pokud uživatel stiskne tlačítko z druhé řady, dojde k načtení presetu příslušného záznamu historie, včetně neuložených rozdílových sad hodnot parametrů nástroje a efektů.

2.8.4 Souhrnné ovládání efektů (obrazovka C3)

Jelikož efekty většinou disponují jedním hlavním parametrem, který určuje míru použití efektu, na obrazovce souhrnného ovládání efektů (viz diagram 2.12) lze ovládat pro každý efekt právě tento zástupný parametr.

První řádek displeje a související řada tlačítek se chová podobně jako v případě obrazovky ovládání parametrů nástroje s tím rozdílem, že jsou použity hlavní mapování parametrů každého efektu namísto mapování parametrů nástroje. Navíc je místo zobrazení názvu mapování parametru na displeji vždy zobrazena hodnota parametru, jelikož je zřejmé, že je se jedná o hlavní parametr pro daný efekt.

Na druhém řádku displeje je zobrazen seznam dostupných efektů pro danou vrstvu a umístění. Při stisknutí tlačítka z druhé řady je načtena obrazovka ovládání všech parametrů příslušného efektu (viz kapitolu 2.8.5).

2.8.5 Ovládání parametrů efektu (obrazovka C4)

Na této obrazovce je k dispozici ovládání parametrů efektu (viz diagram 2.13). Chování je podobné jako u obrazovky ovládání parametrů nástroje (viz sekci 2.8.3), druhá řada displeje a tlačítek slouží k přepínání ovládaného efektu podobně jako u obrazovky souhrnného ovládání efektů (viz kapitolu 2.8.4).

Kapitola 3

Implementace vlastního řešení

3.1 Reaktivní programování

Pro velkou část aplikace bylo zvoleno reaktivní programovací paradigma, které je orientováno na datové toky a šíření změn. Lze tak jednoduše sledovat změny hodnot, provádět nad nimi různé transformace, filtrování, agregaci a další operace. Podporu reaktivního programování přináší do aplikace použitý framework RxJS[12]. Velkou výhodou je možnost udržování počtu naslouchání s využitím např. funkce `switchMap`, zrušení nerelevantních naslouchání pak probíhá automaticky a není předmětem práce programátora. Framework RxJS je využitý též knihovnou NGXS zajišťující správu stavu aplikace (viz kapitolu 3.3).

3.2 Komunikace přes USB

Pro komunikaci s hardwarovým kontrolerem nebylo možné použít dostupné API pro použití MIDI protokolu, jelikož se zařízení systému nejeví jako standardní MIDI zařízení. Formát komunikace však příslušné třídě USB rozhraní odpovídá. Proto bylo nezbytné implementovat vrstvu, která umožní přijímat a odesílat MIDI zprávy typu CC a SysEx podle protokolu USB třídy pro MIDI zařízení[8].

Zprávy typu CC jsou opatřeny jednoduchou hlavičkou a celá zpráva má konstantní délku 4B.

SysEx zprávy představují zprávy variabilní délky. Jelikož lze v jednom packetu přenést pouze 4B, je třeba zprávy tohoto typu rozdělit na zarovnané části a ty odesílat, resp. zpracovávat samostatně.

Třída `MidiAdapter` tak poskytuje částečnou implementaci tohoto protokolu. Obsahuje metodu pro odeslání zprávy a možnost naslouchat příchozím zprávám za použití třídy `Observable` z knihovny RxJS. Třída využívá funkcí třídy `USBDriver`, jejíž instance je v konstruktoru předána s pomocí techniky dependency injection[13].

Třída `USBDriver` poskytuje jednoduché rozhraní pro komunikaci s připojeným USB zařízením. Umožňuje odesílat data a poskytuje možnost naslouchání údalostem typu příjem dat, připojení a odpojení zařízení podobně jako třída `MidiAdapter`.

3.3 Stavový model aplikace a knihovna NGXS

Aplikace je rozdělena na klientskou a serverovou část. Jelikož hlavní logika ovládání hardwarového kontroleru a VST host aplikace musí vykazovat co nejnižší odezvu, je třeba, aby byl dostupný aktuální stav aplikace v serverové části. Zároveň však musí být stav přístupný i části klientské, aby mohly být reflektovány změny stavu iniciované přes hardwarový kontroler nebo VST host aplikaci.

Knihovna NGXS představuje framework pro správu stavu aplikace podle návrhového vzoru Flux[14], který vychází ze vzoru CQRS[15]. Výhodou je možnost použití vzoru Observer u celého stavu aplikace a docílit tím asynchronního přístupu ke stavu nebo jeho částem[16]. Stav je aktualizován tzv. akcemi, které lze kdykoliv vyvolat, mají určený jednoznačně identifikovaný typ a mohou nést libovolná data.

Pro synchronizaci stavu mezi klientskou a serverovou částí aplikace byla implementována vrstva, která funguje jako proxy pro akce vyvolané v obou částech aplikace. Přenos akcí probíhá přes systém zasílání zpráv poskytovaný frameworkem Electron. Veškeré akce vyvolané v jedné z částí aplikace jsou serializovány a odesílány ve formě zprávy do opačné části aplikace, kde probíhá deserializace a předání akce ke zpracování frameworkem NGXS. Při vyvolání akce v jedné části aplikace tak dojde ke změně stavu v serverové i klientské části zároveň. U každé zprávy lze nastavit příznak určující, že akce nemá být odeslána do opačné strany aplikace. Zároveň je možné stanovit tzv. blacklist, tedy seznam akcí, které nebudou předávány druhé straně. Lze tak filtrovat např. interní akce frameworku NGXS a jeho přídavných modulů k zamezení vzniku jejich duplicity nebo zacyklení. Jelikož obě části aplikace používají pouze část celého stavu, jsou předané akce měnící stav irelevantní pro danou část aplikace jednoduše ignorovány, jelikož v dané části aplikace není zaregistrována komponenta, která by na zprávy reagovala změnou lokálního stavu. Možná je budoucí optimalizace komunikace zamezením přenosu irelevantních akcí.

Při návrhu jednotlivých částí stavu bylo třeba dbát na požadavek, aby nedocházelo k přenosu zpráv (a tedy i vyvolávání automaticky předávaných akcí) s vysokou frekvencí nebo velkým množstvím dat. Ve sdílené části stavu aplikace tak nebude zahrnuta např. informace o aktuálních hodnotách parametrů VST pluginů, jelikož může docházet k jejich intenzivním změnám ze strany VST hosta nebo hardwarového kontroleru.

Struktura modelu stavu aplikace je normalizovaná z důvodu zamezení du-

plicity dat a zajištění datové konzistence[17]. Pro CRUD operace nad entitami bylo vytvořeno vlastní jednoduché API.

V následujících podkapitolách následuje výčet jednotlivých částí modelu stavu aplikace.

3.3.1 Synchronizace přes IPC

Pro klientskou i serverovou část byly implementovány třídy `IpcActionReceiver` a `IpcActionTransmitterPlugin`. Třída `IpcActionTransmitterPlugin` implementuje NGXS API pro tvorbu pluginů a zajišťuje zasílání kopií akcí vyvolaných přes NGXS framework druhé straně aplikace. K příjmu zprávy slouží třída `IpcActionReceiver`, která akci předá NGXS frameworku tak, že simuluje lokální vyvolání dané akce. Zprávy jsou serializovány do formátu JSON a k jejich přenosu dochází za použití IPC API frameworku Electron.

3.3.2 API pro CRUD operace nad entitami

Jelikož skrz celou aplikací bude potřeba provádět nad entitami operace typu přidání, čtení, aktualizaci a odstranění, bylo nejprve implementováno API pro tyto operace, aby nebylo nutné využívat duplicitní kód pro každou entitu zvlášť. Implementovány byly operace aktualizace, částečná aktualizace, přidání, přidání s případnou aktualizací a odstranění. Operace pro čtení je triviální a každá třída reprezentující stav pro danou entitu jí implementuje zvlášť.

3.3.3 LayoutState

V tomto stavu budou obsažené informace o uložených projektech VST host programu Bidule. Model používá API pro CRUD operace. Stav je globální, jeho obsah tedy není ukládán v závislosti na právě otevřeném projektu. Kromě samotných entit obsahuje informace o právě načteném projektu a příznak určující zda právě probíhá načítání layoutu. V modelu je obsažena logika pro řazení projektů podle času posledního použití.

3.3.4 FrontendState

Tento stav je jediný, který není sdílený mezi serverovou a klientskou částí. Obsahuje informace typické pro klientskou část, např. zda již byla aplikace inicializována. Původním záměrem bylo zahrnout i informace o navigaci, ale k tomuto účelu byla později vybrána knihovna `ngxs/router-plugin`, která tuto informaci obsahuje v rámci svého vlastního stavu a poskytuje též možnost provést změnu navigace odesláním tzv. akce za použití frameworku NGXS.

3.3.5 VSTState

V tomto stavu jsou obsaženy informace o dostupných VST pluginech. Stav je uložen pro každý projekt zvlášť (viz kapitolu 3.8) a je sdílený klientskou i serverovou částí aplikace. Model používá z API pro CRUD operace pouze funkce pro přidání a aktualizaci. Pro získávání entit představujících virtuální nástroje a efekty jsou přítomné příslušné metody. Zároveň je zde implementována obsluha akcí pro správu stránek mapování parametrů VST pluginů pro virtuální nástroje i efekty.

3.3.6 ManualState

Tento stav obsahuje informace o manuálech a vrstvách dostupných v otevřeném projektu a je sdílený klientskou i serverovou částí aplikace. Není ukládán do perzistentního úložiště. Model používá z API pro CRUD operace pouze funkce pro přidání, jelikož k načtení stavu dochází pouze při otevření projektu. Obsahuje metody pro získání aktuálně zvolené vrstvy a manuálu.

3.3.7 SessionState

V tomto stavu jsou uloženy informace o aktuálním sezení. Stav je sdílený klientskou i serverovou částí aplikace a není ukládán do perzistentního úložiště. Mezi obsažené informace patří např. ID aktuálně zvolené vrstvy a další informace vycházející z aktuálního stavu navigace z pohledu celé aplikace.

3.3.8 ParamMappingPageState

Stránka mapování může obsahovat až 8 jednotlivých mapování, v systému se s ní pracuje dále jako s elementárním celkem. Ve stavu `ParamMappingPageState` je pak uložna aktuálně načtená stránka mapování. Stav není ukládán do perzistentního úložiště a je sdílen klientskou i serverovou částí aplikace. Kromě funkcí API pro CRUD operace obsahuje model i logiku pro správu jednotlivých mapování. Poskytuje také možnost řazení jednotlivých mapování v rámci stránky a informaci o aktuálním zvoleném mapování. Impelementována je zde i logika pro načítání uložených stránek mapování z VST efektů a nástrojů.

Třída `ParamMappingLoaderService` zajišťuje automatickou aktualizaci stavu `ParamMappingPageState` při změně aktuální vrstvy nebo presetu. Pokud je zvolena vrstva i preset, načte stránku mapování parametrů nástroje danou v presetu, jinak načte prázdnou stránku mapování.

3.3.9 PresetCategoryState

Tento stav obsahuje uložené kategorie presetů. Je sdílen klientskou a serverovou částí aplikace a jeho obsah je ukládán do perzistentního úložiště pro každý projekt zvlášť. Používá API pro CRUD operace. Dále poskytuje logiku

vyřizování CRUD operací samotných presetů z pohledu kategorie. Lze tak presety v rámci kategorie řadit, přidávat, duplikovat, či odebírat.

3.3.10 PresetState

V tomto stavu jsou obsažené informace o uložených presetech. Je sdílen klientskou a serverovou částí aplikace a jeho obsah je ukládán do perzistentního úložiště pro každý projekt zvlášť. Kromě funkcí z API pro CRUD operace obsahuje logiku pro duplikaci presetů vč. generování názvu duplikovaných presetů.

3.3.11 PresetSessionState

Tento stav pro každou vrstvu zvlášť uchovává informace o aktuálně načteném presetu. Mezi tyto informace patří např. historie naposledy použitých presetů a příznaky pro ovládání úpravy aktuálního presetu. V modelu stavu je obsažena logika pro správu historie naposledy použitých presetů. Stav je sdílen mezi klientskou a serverovou částí aplikace a není ukládán do perzistentního úložistě.

3.4 OSC a VST host Bidule

Pro obousměrnou komunikaci serverové části vlastní aplikace s VST hostitelskou aplikací přes protokoly UDP a OSC je použita knihovna `node-osc`. Tato knihovna umožňuje odesílat a přijímat OSC zprávy asynchronním způsobem. Příjem zpráv lze realizovat naslouchánímzprávám odpovídajícím zadanému vzoru. Pokud již daný příjem zpráv není relevantní, je třeba naslouchání zastavit, aby nedocházelo ke zbytečnému čerpání systémových prostředků. Proto byla zavedena třída `OscService`, která umožňuje vytvořit k naslouchání instanci třídy `Observable` z frameworku RxJS. Třída též obsahuje mapu posledních hodnot pro danou OSC adresu. Lze tak použít i přístup `BehaviorSubject` z frameworku RxJS, kdy lze naslouchat změnám hodnot, po inicializaci naslouchání je však odeslána úvodní hodnota.

3.4.1 Načítání projektů

Program Bidule umožňuje načtení projektu ze zvoleného XML souboru odesláním OSC zprávy. Tímto způsobem je ve vlastní aplikaci implementováno načítání projektů. Při otevření projektu dojde k odeslání příslušné OSC zprávy do programu Bidule. Dále se v časovém intervalu 1s odesílá tzv. beacon zpráva vyžadující odpověď. Pokud odpověď dorazí, znamená to, že je projekt již načtený. Pokud odpověď nedorazí, znamená to, že se projekt stále v programu Bidule načítá, nebo že došlo k chybě. Při zavření projektu dojde v programu Bidule k načtení dodaného prázdného projektu.

3.4.2 Parametry modulů

V programu Bidule může představovat modul buďto některý z interních modulů nebo VST plugin. Každý modul poskytuje seznam svých vlastních parametrů. Program Bidule pro tyto parametry umožňuje jejich hodnoty měnit a naslouchat jejich změnám s pomocí protokolu OSC. Zároveň pro každý modul zavádí obecné parametry, jako např. pracovní mód, pořadí vybraného presetu apod. Číselné hodnoty parametrů jsou normalizovány na hodnotu 1.

3.4.3 Zobrazení UI modulu

Přes protokol OSC lze v programu Bidule též zobrazit nebo schovat uživatelské rozhraní jakéhokoliv modulu a zpracovávat notifikace o jeho zobrazení, resp. skrytí uživatelem. Ve vlastní aplikaci nejsou notifikace zpracovávány. Zobrazení uživatelského rozhraní modulu probíhá na žádost uživatele nebo při editaci konfigurace za použití tzv. funkce learn, kdy se konfigurace mění v závislosti na akcích provedených přímo v prostředí modulu programu Bidule.

3.4.4 Ovládání pracovního módu modulů

V programu Bidule každý modul poskytuje obecný parametr představující svůj aktuální pracovní mód. Ten může nabývat následujících hodnot:

- `Processing` - modul je aktivní a zpracovává data.
- `Bypass` - modul je aktivní a jsou do něj odesílána data, výstupní data jsou však ignorvána.
- `Mute` - modul není aktivní a nevyužívá výpočetní výkon systému.

Jelikož je jedním z požadavků na systém optimalizace využití systémových prostředků, je třeba ovládat pracovní mód modulu dynamicky. Děje se tak u modulů představujících VSTi pluginy, pro danou vrstvu tak může být současně aktivní pouze jeden virtuální hudební nástroj. V budoucnu je možné použít ovládání pracovního módu i pro efekty podle předem zadaných pravidel pro hlavní mapování parametrů efektu k docílení další optimalizace systémových prostředků.

3.4.5 Volba presetů VST pluginů

Pro každý modul program Bidule zavádí obecné parametry představující číslo a název aktuálního zvoleného presetu. Tyto parametry jsou použité u jedné ze strategií inicializace presetu v rámci tzv. funkce learn, kdy dochází ke změnám konfigurace v prostředí programu Bidule, a v rámci samotné inicializace presetu.

3.4.6 Synchronizace vlastních parametrů modulů

Přes protokol OSC lze také ovládat všechny vlastní parametry modulů v projektu programu Bidule. K nastavení hodnot dochází při ovládání hardwarovým kontrolerem a inicializaci presetu. Naslouchání změnám hodnot se využívá při prezentaci hodnot na hardwarovém kontroleru a při aktivní tzv. learn funkci, kdy dochází ke změně konfigurace mapování parametrů v prostředí programu Bidule.

3.4.7 Generování MIDI zpráv

Jelikož je jednou ze strategií inicializace virtuálního nástroje odeslání MIDI zprávy typu program change, byl do architektury projektu programu Bidule zahrnutý vlastní modul, který umožňuje přes OSC odeslat požadovanou MIDI zprávu do libovolného virtuálního nástroje. Tento modul bude použitý také u vyslání obecných MIDI zpráv, mezi které patří např. zprávy typu panic, které zajistí přerušení všech znějících tónů.

3.5 Automap a hardwarový kontroler

Pro podporu komunikace s hardwarovým kontrolerem přes protokol Automap byla vytvořena speciální vrstva. V budoucnu je možno tuto vrstvu nahradit jinými implementacemi pro podporu dalších ovládacích zařízení. Vrstva využívá třídu `MidiAdapter` pro přenos zpráv podle protokolu Automap. V následujících podkapitolách jsou popsány jednotlivé části této vrstvy, která zajišťuje vstupně-výstupní operace s připojeným hardwarovým kontrolerem.

3.5.1 Inicializace zařízení

Po připojení zařízení protokol Automap přikazuje provést tzv. handshake, tedy výměnu zpráv pro zahájení komunikace. Po připojení hardwarového kontroleru dojde k automatickému odeslání zprávy do zařízení a systém čeká na příjem potvrzující zprávy. Obě zprávy jsou typu SysEx a nesou informaci o verzi protokolu a další konfiguraci zařízení.

3.5.2 Tlačítka kontroleru

Protokol Automap umožňuje přes MIDI zprávy typu CC předávat informace o stisknutí tlačítka hardwarového kontroleru. Zároveň poskytuje možnost nastavit stav LED podsvícení příslušného tlačítka. Jelikož v samotném hardwarovém kontroleru není k dispozici žádná další logika, bylo potřeba implementovat rozšířenou logiku chování tlačítek v aplikaci.

Samotná tlačítka uživatelského rozhraní kontroleru lze rozdělit do logických celků. Pro reprezentaci takového celku v kódu vznikla třída `ButtonGroup`. Tato třída obsahuje metody pro nastavení stavu jednotlivých LED a všech

LED v rámci celku. Zároveň za použití třídy `EventEmitter`, která implementuje rozhraní `Observable` z frameworku RxJS, poskytuje možnost naslouchání na událost stisku tlačítka reprezentované třídou `ButtonClickEvent`. Tlačítka jsou identifikována číslem stanoveným protokolem Automap, které je při komunikaci přenášeno v MIDI zprávě typu CC. Třída `ButtonGroup` v konstruktoru přijímá seznam identifikátorů tlačítek v daném celku. Zároveň poskytuje i možnost nastavení příznaku pro ignorování stisknutí. Tento příznak lze zadat pro jednotlivá tlačítka zvlášť nebo hromadně pro všechna tlačítka v rámci skupiny. Pokud je stisknuto tlačítko, které nemá nastavený příznak ignorování stisku, dojde zároveň k jeho podsvícení dokud není tlačítko uvolněno.

Pro instanci třídy typu `ButtonGroup` lze též nastavit objekt impelmentující rozhraní `ClickHandlerInterface`, který reaguje na stisknutí, resp. uvolnění tlačítka. Nastavení lze provést globálně pro celou skupinu tlačítek nebo pro každé tlačítko zvlášť. Rozhraní `ClickHandlerInterface` umožňuje třídě vyvolat vlastní událost představující stisknutí tlačítka. Vlastní událost musí splňovat rozhraní třídy `ButtonClickEvent`. Pro zpracování vstupu definuje rozhraní metody, které jsou zavolány po stisknutí, nebo uvolnění tlačítka. Jako výchozí implementace rozhraní `ClickHandlerInterface` je při vytváření třídy `ButtonGroup` inicializována instance třídy `SimpleClickHandler`, která zanedbává informace o uvolnění tlačítka a vyvolává událost představující stisknutí tlačítka ihned po fyzickém stisknutí tlačítka. Další implementací rozhraní `ClickHandlerInterface` je třída `MultiClickHandler`, která zajišťuje zpracování vícenásobného stisknutí tlačítka. V rámci instance lze nastavit nejvyšší počet stisknutí tlačítka a maximální časový rozestup mezi jednotlivými stisky tlačítka, než dojde k vyvolání události stisknutí tlačítka. Událost `ButtonMultiClickEvent` poté nese kromě identifikátoru stisknutého tlačítka i informaci o počtu stisknutí tlačítka. V budoucnu je možné implementovat např. třídu pro zpracování podržení tlačítka, kde budou zohledněny i vstupní události fyzického uvolnění tlačítka.

Jelikož mají logické celky tlačítek často pravoúhlý tvar, kde je výhodné použít adresaci pomocí kartézské soustavy souřadnic, byla pro tento účel implementována třída `ButtonMatrix` využívající s pomocí kompozice třídu `ButtonGroup`. V konstruktoru je navíc vyžadována informace o rozměrech mřížky a identifikátor prvního tlačítka. Identifikátory dalších tlačítek jsou vypočítány automaticky podle číslování plynoucího z protokolu Automap. Události stisknutí tlačítka v mřížce, kterým lze naslouchat, obsahují informaci o souřadnicích tlačítka. Reprezentuje je třída `ButtonMatrixEvent`, jejíž rozhraní vychází z třídy `ButtonClickEvent`. K dispozici je i vazba na původní událost typu `ButtonClickEvent`, která je řešena pomocí kompozice.

Pro podporu přerušovaného podsvícení tlačítek byla zavedena třída `InterruptionClock`. Díky této třídě lze udržovat přerušované podsvícení synchronní napříč celou aplikací. Třída podporuje čtyři stupně rychlosti blikání. S pomocí třídy `EventEmitter` z frameworku RxJS lze naslouchat obdélníkovému signálu představujícímu stav podsvícení podle zvoleného stupně frekvence přerušení.

3.5.3 Displej kontroleru

Příkazy pro ovládání displeje využívají MIDI zpráv typu SysEx. Je možné smazat celý obsah displeje a vypsat ASCII řetězec na pozici určené dvojicí souřadnice. API třídy `Display` pro ovládání displeje tyto funkce poskytuje. Tato vrstva též zajišťuje možnost vypsat text do příslušné buňky s automatickým zarovnáním, nebo zkrácením řetězců. Veškeré operace lze provést okamžitě, nebo je provádět pouze lokálně a poté odeslat do zařízení celý obsah displeje. Pro tuto optimalizace třída zahrnuje buffer obsahující aktuální obsah celého displeje.

3.5.4 Rotační enkodéry kontroleru

Hardwarový kontroler poskytuje tři druhy otočných ovladačů. Tzv. nekonečné knoby (rotační enkodéry obklopené prstencem LED), standardní potenciometry a rotační enkodér bez prstence LED pro hlavní navigaci. Všechny otočné ovladače podporují vyslaní signálu při dotyku, resp. jeho přerušení.

U nekonečných knobů lze nastavit mód zobrazení hodnoty na LED prstenci (např. zobrazení bodem, kruhovou výsečí). Při otočení enkodéru zařízení odešle MIDI zprávu určující směr a rychlost pohybu.

Rotační enkodér pro hlavní navigaci lze stisknout a odeslat tak příslušnou MIDI zprávu. Při jeho otočení zařízení odešle MIDI zprávu podobně jako u rotačních enkodérů s LED prstencem.

U klasických potencimetrů zařízení odesílá při změně hodnoty aktuální hodnotu vyjádřenou 7 bity.

Třída `Knobs` zastupuje skupinu nekonečných knobů s LED prstencem. Umožňuje pro každý knob změnit mód zobrazení hodnoty a nastavit hodnotu indikovanou LED prstencem. Též poskytuje s pomocí třídy `EventEmitter` z frameworku RxJS události, kterým lze naslouchat. Třída `KnobEvent` představuje rodičovskou třídu pro všechny události knobů. Třída `RotationKnobEvent` jí dědí a přidává informaci o směru a rychlosti otočení. Naslouchat lze událostem dotyku, uvolnení dotyku a otočení enkodéru.

3.5.5 Architektura

Třídy implementující jednotlivé části vrstvy jsou použity ve větších celcích určených rozmístením prvků na hardwarovém kontroleru. Např. třída `NavigationRegionDriver`, která představuje množinu ovládacích prvků určených k navigaci, využívá dvě instance třídy `ButtonGroup`. Třída `DisplayRegionDriver` pro ovládní prvků v oblasti displeje obsahuje instance tříd `Display`, `Knobs` a `ButtonMatrix`. Tyto celky jsou použity v implementaci logiky uživatelského rozhraní hardwarového kontroleru.

3.6 Uživatelské rozhraní hardwarového kontroleru

Jednotlivé části vrstvy uživatelského rozhraní hardwarového kontroleru jsou orchestrovány s pomocí třídy `KeyboardRouter`. Ta zároveň slouží pro navigaci v rámci uživatelského rozhraní kontroleru. Pro tento účel poskytuje metodu `navigate`, která způsobí načtení požadovaného stavu navigace. Stav navigace je vyjádřen řetězcem, jednotlivé stavy nemůžou obsahovat žádná metadata. Ta je případně vhoné přenášet ve vlastnosti `KeyboardRoute` stavu `SessionState` s využitím knihovny NGXS. Třída `KeyboardRouter` obsahuje referenci na všechny komponenty řídící jednotlivé části uživatelského rozhraní hardwarového kontroleru. Podle aktuálního stavu navigace určuje, které komponenty jsou právě aktivní. Každá komponenta pro řízení části uživatelského rozhraní kontroleru musí implementovat rozhraní `MortalInterface`, které obsahuje deklaraci metody `onInit` a `onDestroy`. Z pohledu návrhového vzoru MVC[18] se chovají takové komponenty dohromady jako view a controller. V následujících podkapitolách budou diskutovány jednotlivé komponenty pro řízení části uživatelského rozhraní.

3.6.1 Komponenty pro ovládání oblasti displeje

Jelikož lze na displeji dobře indikovat aktuální chování přidružených prvků, oblast displeje může být použita pro větší množství obrazovek. V této podkapitole budou diskutovány komponenty pro řízení oblasti displeje. Všechny komponenty pracují s vrstvou poskytující vstupně-výstupní operace s připojeným hardwarovým kontrolerem (viz kapitolu 3.5). Využívají pouze část vrstvy, která poskytuje ovládání prvků oblasti displeje, tedy třídu `DisplayRegionDriver`.

3.6.1.1 ParamMappingContoller

Tato komponenta ovládá pouze jeden řádek oblasti displeje a měla by být tedy použita v rámci komponenty obsluhující celou obrazovku. Je využitá v třídách `InstrumentDetailController`, `EffectOverviewController` a `EffectDetailController` implementující obrazovky C2 - C4 (viz kapitoly 2.8.3, 2.8.4, 2.8.5). Zajišťuje chování pro ovládání a indikaci parametrů VST pluginů. Aktuálně ovládané parametry a jejich mapování načítá ze stavu `ParamMappingPageState`. U otočných enkodérů nastavuje mód zobrazení hodnoty na LED prstenci a indikuje samotnou hodnotu podle zvolené strategie mapování. Též reaguje na události otočného enkodéru. Při jeho otočení podle strategie mapování vyvolá akci aktualizace hodnoty parametru VST pluginu. Zobrazení na displeji poskytuje standardně název ovládaného parametru, pokud však není nastavený příznak pro trvalé zobrazení hodnot. Aktuální hodnota ovládaného parametru je pak zobrazena jen pokud dojde k její změně nebo pokud se uživatel dotkne příslušného otočného enkodéru. Formát zobrazení hodnoty na

displeji je určen stragetií mapování. Hodnota parametru je zobrazena jen po určitý čas, poté se opět zobrazení vrátí na název ovládaného parametru. Toto chování je zajištěno netriviální logikou vyjádřenou za použití reaktivního programování a knihovny RxJS.

3.6.1.2 LastPresetsController

Podobně jako u třídy `ParamMappingController` je ovládán pouze jeden řádek oblasti displeje. Na řádku jsou zobrazeny záznamy historie posledních použitých presetů a aktuální preset. Při stisknutí příslušného tlačítka dojde k vyvolání akce `SelectPresetAction` určující, který preset má být načten. Třída načítá data ze stavu `PresetSessionState` pomocí metody `getCurrentHistory`, jejíž výstup je závislý na aktuální vrstvě. Při změně aktivní vrstvy tedy dojde k zobrazení záznamů historie pro danou vrstvu.

3.6.1.3 EffectSwitchController

Pro přepínání ovládaných efektů byla vytvořena třída `EffectSwitchController`. Ovládá pouze jeden řádek oblasti displeje a je používaná třídami `EffectOverviewController` a `EffectDetailController`. Data načítá ze stavů `SessionState`, `ManualState` a `VstState`. Poskytuje uživateli seznam efektů dostupných pro aktuálně zvolený kontext, který zahrnuje aktuální vrstvu a konfiguraci zobrazení efektů, tedy umístění efektů (vstup nebo výstup) a příznak pro ovládání globálních efektů. Při stisknutí tlačítka dojde k vyvolání vlastní události volby efektu s použitím třídy `EventEmitter` z frameworku RxJS. Aktivní efekt je možné nastavit s použití metody `SetActiveEffect`, která interně využívá třídu `BehaviorSubject` z frameworku RxJS.

3.6.1.4 InstrumentDetailController

Tato třída využívá v kombinaci tříd `ParamMappingController` a `LastPresetsController` tak, aby bylo dosaženo kýženého chování (viz kapitolu 2.8.3). Aktualizace stránky mapování parametrů nástroje je implementována ve třídě `ParamMappingLoaderService`.

3.6.1.5 EffectOverviewController

Obrazovka souhrnného ovládání efektů (viz kapitolu 2.8.4) je implementována třídou `EffectOverviewController`, která využívá jako podkomponenty třídy `EffectSwitchController` a `ParamMappingContoller`. Pokud je vyvolána událost volby efektu instancí třídy `EffectSwitchController`, dojde k nastavení příslušného stavu navigace v rámci hardwarového kontroleru a je načtena obrazovka detailu efektu (viz kapitolu 2.8.5). U instance třídy `ParamMappingController` je nastaven příznak pro trvalé zobrazení hodnoty parametru efektu. Aktualizace

stránky mapování parametrů probíhá automaticky při změně konfigurace dostupných efektů vyvoláním akce `LoadParamMappingPageAction`. U každého dostupného efektu je vybráno hlavní mapování a z těchto mapování je poté sestrojena výsledná stránka mapování.

3.6.1.6 EffectDetailController

Podobně jako u třídy `EffectOverviewController` je u této třídy využita komponenta `EffectSwitchController`, s pomocí které je též indikován stav aktuálně ovládaného efektu. Událost volby efektu je buď reflektována změnou stavu určením nového aktivního efektu, nebo je uživatel přesměrován zpět na obrazovku souhrnného ovládání efektů (viz kapitolu 2.8.4). Zároveň je třídou `EffectDetailController` využita i komponenta `ParamMappingController` poskytující stránku mapování určenou aktuálně ovládaným efektem. Při změně části stavu určující aktuálně ovládaný efekt je vyvolána akce načtení stránky mapování parametrů z daného efektu.

3.6.1.7 PresetController

Třída PresetController představuje implementaci obrazovky C1 (viz kapitolu 2.8.2). Zajišťuje procházení kategorií presetů a volbu a indikaci aktuálního presetu. Data získává prostřednictvím stavů `SessionState`, `PresetCategoryState`, `PresetState` a `PresetSessionState`. Aktuální zvolená kategorie je obousměrně synchronizována s desktopovým uživatelským rozhraním aplikace. Pro možnost stránkování je využitá třída pro zpracování vícenásobného stisku tlačítka `MultiClickHandler`. Netriviální logika je implementována za použití reaktivního programování ve frameworku RxJS.

3.6.1.8 EmptyController

Instance této komponenty je načtena po spuštění aplikace nebo připojení hardwarového kontroleru k systému. Nereaguje na žádné vstupy od uživatele, na displeji zobrazí informativní hlášku.

3.6.2 Ostatní komponenty

3.6.2.1 Třída LayerController

Třída `LayerController` implementuje chování čtveřice tlačítek nalevo pod displejem. Slouží k určování a indikaci aktuálně zvoleného manuálu a vrstvy. Tlačítka představují jednotlivé manuály. Rychlost blikání tlačítek značí pořadí vrstvy s využitím třídy `InterruptionClock`. Pro nastavení aktuálního manuálu a vrstvy je využita třída pro zpracování událostí stisknutí tlačítka `MultiClickHandler`. V případech, kdy kvůli omezením není možné přepínat aktuální vrstvu (např. při editaci presetu) je potlačeno veškeré ovládání a indikace

aktuální vrstvy. Komponenta je globální a její instance je tedy aktivní po celou dobu běhu aplikace.

3.6.2.2 Třída DisplayModeController

K přepínání obrazovek, tedy určení aktivní komponenty ovládající oblast displeje, slouží třída `DisplayModeController`. Dvojicí tlačítek umístěných nalevo od displeje lze zobrazit až 6 možných obrazovek (viz kapitolu 3.6.1). Chování tlačítek využívá tříd `MultiClickHandler` a `InterruptionClock` tak, aby splnily požadované chování (viz kapitolu 2.8.1).

3.7 Desktopové uživatelské rozhraní

Pro desktopové uživatelské rozhraní byl zvolen framework Angular Material, který přináší běžné prvky uživatelského rozhraní pro jejich použití ve frameworku Angular. Aplikace využívá prvky pro rozvržení částí UI, formuláře, tlačítka, seznamy a další. Stav desktopového uživatelského rozhraní je do značné míry nezávislý na stavu uživatelského rozhraní hardwarového kontroleru. Výjimky budou popsány v následujících podkapitolách, které popisují jednotlivé části desktopového uživatelského rozhraní.

3.7.1 Rozdělení do modulů

Podle doporučeného návrhu architektury je desktopová část aplikace rozdělena do modulů, které řeší jednotlivé části problémové domény[19]. V rámci modulu lze např. určit konfiguraci pro lokální navigaci, zavádět znovupoužitelné komponenty a implementovat vlastní direktivy pro šablony. Kromě kořenového modulu `AppModule` aplikace obsahuje následující klíčové moduly:

- `HomeModule`
- `MaterialModule`
- `SharedModule`
- `ParamMappingModule`
- `VstModule`
- `LayoutModule`
- `InstrumentModule`
- `EffectModule`
- `PresetModule`

3.7.2 Navigace

Pro účely navigace v rámci desktopového uživatelského rozhraní využívá aplikace nativního přístupu poskytovaného frameworkem Angular (s využitím modulu `RouterModule`). Knihovna `@ngxs/router-plugin` pak zajišťuje synchronizaci aktuálního stavu navigace s částí stavu v rámci knihovny NGXS.

V modulu `AppRoutingModule` pro zajištění hlavní navigace v rámci kořenového modulu jsou definovány následující stavy navigace identifikované prefixem cesty ve formě řetězce. Jednotlivé moduly pak řeší svojí lokální navigaci zvlášť.

- `''` - modul `HomeModule` pro zobrazení úvodní obrazovky
- `'layouts'` - modul `LayoutModule` zajišťující správu uložených projektů programu Bidule
- `'instruments'` - modul `InstrumentModule` pro práci s virtuálními hudebními nástroji
- `'effects'` - modul `EffectModule` pro práci s virtuálními efekty
- `'presets'` - modul `PresetModule` pro správu uložených presetů
- `'**'` - tzv. fallback, který zajistí zobrazení chybové obrazovky pro neplatné cesty navigace.

Jednotlivé stavy navigace mohou být aktivovány pouze za určitých omezení, např. pokud je načtený projekt programu Bidule. Omezení jsou vyjádřena tzv. guardy, které představují např. třídy `NoLayoutGuard` a `LayoutGuard` z modulu `LayoutModule`.

Ke změně stavu navigace uživatelem slouží vertikální menu definované globálně v rámci hlavní komponenty `AppComponent` z modulu `AppModule`. Pro vykreslení menu je použita komponenta `MatSidenav` z frameworku Angular Material.

Navigace v rámci uživatelského rozhraní hardwdarového kontroleru je do značné míry nezávislá na stavu navigace desktopového uživatelského rozhraní aplikace. Výjimkou je případ editace některých částí konfigurace projektu, kdy v rámci uživatelského rozhraní hardwarového kontroleru dochází k nastavení stavu navigace a odepření možnosti jeho změny.

3.7.3 HomeModule

V tomto modulu se nachází implementace komponenty pro zobrazení úvodní obrazovky aplikace. Komponenta `HomeComponent` neobsahuje žádnou logiku, obsažená je pouze jednoduchá šablona. Obrazovka je načtena pouze po inicializaci aplikace.

3.7.4 MaterialModule

Tento jednoduchý modul je importován všemi moduly, které pracují s knihovnou Angular Material. Exportuje veškeré prvky knihovny Angular Material, které jsou použity napříč celou aplikací.

3.7.5 SharedModule

V modulu `SharedModule` jsou implementovány veškeré komponenty, které mohou být použity napříč celou aplikací. Obsahuje např. komponentu pro zobrazení chybového stavu, obecný dialog pro zadání jména (třída `NamedEntityDialogComponent`) a direktivu pro potvrzení libovolné akce uživatelem (třída `ConfirmationDirective`).

3.7.6 ParamMappingModule

Tento modul poskytuje znovupoužitelnou komponentu `ParamMappingPageComponent` pro správu stránek mapování parametrů VST pluginů. Ta umožňuje přidávat, odebírat a upravovat jednotlivé položky stránky mapování. Pro zadání názvu položky využívá komponentu `NamedEntityDialogComponent` definovanou v modulu `SharedModule`. Pro každou položku umožňuje zvolit strategii mapování. Rozhraní `AbstractParamMappingStrategyComponent` představuje API pro komponenty umožňující konfiguraci konkrétního mapování. Díky této abstrakci lze v budoucnu přidávat jednoduše další strategie, aniž by bylo potřeba upravovat již existující kód. Návrh tedy splňuje charakteristiky OCP[20].

Třída `LinearParamMappingStrategyComponent` implementující rozhraní `AbstractParamMappingStrategyComponent` umožňuje konfigurovat lineární strategii mapování parametrů VST pluginů. Využívá formulářových prvků pro zadávání hodnot konfigurace. Přináší pro uživatele též možnost využití tzv. learn metody, kdy dochází ke změnám v konfiguraci přímo z prostředí VST pluginu v programu Bidule. S využitím této metody lze zadávat minimální a maximální hodnotu parametru VST pluginu.

3.7.7 VstModule

Modul `VstModule` obsahuje komponenty společné pro práci s virtuálními nástroji a efekty. Obsahuje např. komponentu `AvailabilityComponent` pro zobrazení dostupnosti VST pluginu v rámci vrstev a na globální úrovni.

3.7.8 LayoutModule

Tento modul slouží ke správě uložených projektů programu Bidule. Komponenta `LayoutListPageComponent` umožňuje zobrazit seznam dostupných projektů programu Bidule. Též přináší možnost načtění, editace, smazání určitého

projektu, či přidání nového projektu. Zadávání detailů projektu je implementováno v komponentě `EditLayoutPageComponent` s využitím formuláře.

Modul obsahuje též třídy `NoLayoutGuard` a `LayoutGuard` pro omezení navigace v případě, že právě není, resp. je načtený projekt programu Bidule.

3.7.9 InstrumentModule

V tomto modulu jsou obsaženy komponenty pro obsluhu virtuálních hudebních nástrojů dostupných v projektu programu Bidule.

Obsahuje komponentu `InstrumentListPageComponent`, která poskytuje seznam dostupných virtuálních nástrojů. Pokud je nástroj dostupný pro aktuálně zvolenou vrstvu, lze pro něj pořídit tzv. snapshot hodnot parametrů, spravovat stránky parametrů, otevřít UI VST pluginu, či zobrazit jeho dostupnost s pomocí komponenty `AvailabilityComponent` z modulu `VstModule`.

Komponenta `InstrumentParamMappingPageComponent` umožňuje spravovat seznam stránek mapování parametrů pro daný virtuální nástroj. Pro zadávání názvu stránky využívá komponentu `NamedEntityDialogComponent` z modulu `SharedModule`. Uživatel může díky této komponentě též vybrat výchozí stránku mapování pro virtální nástroj.

Pro editaci konkrétní stránky mapování je použita komponenta `ParamMappingPageComponent` z modulu `ParamMappingModule`.

3.7.10 EffectModule

Tento modul zahrnuje podobnou funkcionalitu jako modul `InstrumentModule`, pracuje však s virtuálními efekty namísto virtuálních nástrojů. Komponenta `EffectListPageComponent` zajišťuje zobrazení seznamu dostupných efektů, oproti komponentě `InstrumentListPageComponent` přináší informaci o umístění efektu.

Komponent `EffectParamMappingPageComponent` využívá podobně jako komponenta `InstrumentParamMappingPageComponent` pro editaci stránky mapování komponentu `ParamMappingPageComponent` z modulu `ParamMappingModule`. Jelikož efekt nemůže obsahovat více stránek mapování, chybí správa jednotlivých stránek, je však možné vybrat hlavní položku stránky mapování pro daný efekt.

3.7.11 PresetModule

V modulu `PresetModule` je implementována funkcionalita pro práci s presety.

Komponenta `PresetListPageComponent` slouží ke správě kategorií presetů a presetů v rámci zvolené kategorie. Umožňuje přidávat, řadit, editovat a mazat kategorie presetů i samotné presety. Pro zadávání názvu kategorií a nových presetů je využita komponenta `NamedEntityDialog` z modulu `SharedModule`. Řazení položek je implementováno s pomocí komponenty `DragDropModule` z knihovny Angular Material.

Informace o aktuálním presetu a kategorii presetu jsou synchronizovány se stavem aplikace za použití knihovny NGXS. Tyto informace závisí na aktuálně zvolené vrstvě. Jelikož je stav sdílený a jeho editace je možná z prostředí desktopové aplikace i hardwarového kontroleru, je stav desktopového uživatelského rozhraní v tomto případě automaticky synchronizován s uživatelským rozhraním hardwarového kontroleru.

Třída `PresetDetailPageComponent` představuje komponentu pro úpravu konfigurace presetu. Umožňuje zvolit jméno presetu, použitý virtuální nástroj a uloženou stránku mapování parametrů virtuálního nástroje. Poskytuje též možnost zadat detaily strategie pro inicializaci virtuálního nástroje. K tomuto účelu slouží komponenty `ProgramChangeInitStrategyComponent`, `VstPresetInitStrategyComponent` a `SnapshotInitStrategyComponent`. Jelikož informace o aktuálním presetu vč. zvolené strategie pro inicializaci jsou součástí stavu za použití knihovny NGXS, není třeba zavádět obecné rozhraní pro správu inicializačních strategií. Při přidání nové inicializační strategie stačí implementovat komponentu, která bude obsahovat logiku pro aktualizaci stavu.

3.8 Datová vrstva aplikace

Funkce datové vrstvy je z pohledu třívrstvé architektury částečně implementována díky knihovně NGXS pro správu stavu aplikace. V následujících podkapitolách bude probrána perzistence dat.

3.9 Import projektů Bidule

Jelikož je třeba pro každý projekt programu Bidule pracovat s jeho konfigurací, bylo nutné implementovat možnost importu konfigurace projektu do aplikace. Veškeré potřebné informace jsou uloženy v souboru projektu ve formátu XML. K načtení informací z formátu XML byl použitý framework `xml2js`.

Třída `BiduleLayoutParser` zajišťuje načtení projektu podle architektury popsané v kapitole 2.6. Načítá dostupné manuály a vrstvy, virtuální nástroje a efekty, a vytváří třídu `BiduleLayout`, která tyto informace obsahuje.

Zpracování informací z třídy `BiduleLayout` implementuje třída `BiduleLayoutReader`. Ta z poskytuntého objektu odešle příslušné akce za pomoci knihovny NGXS a aktualizuje tak sdílenou část stavu implementovanou ve třídě `VSTState`.

Inicializaci a indikaci načítání projektu poskytuje třída `BiduleLayoutOpener`. Přes protokol OSC do programu Bidule odesílá příkaz pro otevření projektu ze souboru zadaného v argumentu. V pravidelném intervalu pak odesílá OSC zprávy pro ověření, zda již bylo načítání projektu v programu Bidule dokončeno. V takovém případě program Bidule odpoví a aplikace rozliší, zda došlo k chybě, nebo úspěšnému načtení projektu.

3.10 Ukládání a načítání konfigurace

Perzistence konfigurace aplikace pracuje se snímkem aktuálního stavu aplikace za pomoci knihovny NGXS. Třída `StatePersisterPlugin` implementuje zásuvný modul pro knihovnu NGXS, který jednotlivé části stavu serializuje do formátu JSON a ukládá je do souborů. Perzistence probíhá na globální nebo lokální úrovni, kde je konfigurace uložena pro každý projekt zvlášť. Třída `StateLoaderHandler` zajišťuje načítání uložené konfigurace po inicializaci aplikace, nebo po načtení projektu.

K uložení informací slouží vyvolání akce `PersistStateAction`. K tomu dochází automaticky v daném časovém intervalu, nebo přes desktopové uživatelské rozhraní.

Kapitola 4

Testování

Testování probíhalo pouze na úrovni uživatelského rozhraní. V rámci aplikace nejsou zahrnuty jednotkové ani integrační testy. Testování uživatelského rozhraní bylo provedeno s pomocí heuristické analýzy dle Jakoba Nielsena[21]. Zároveň byly provedeny uživatelské testy. Podrobné informace o testování uživatelského rozhraní jsou rozebrány v následujících podkapitolách.

4.1 Heuristická analýza

4.1.1 Visibility of system status - Viditelnost stavu systému

V uživatelském prostředí desktopové aplikace uživatel vidí aktuální stav navigace. Při načítání projektu programu Bidule je zobrazen tzv. spinner, který značí, že se projekt stále načítá. Je-li právě otevřený nějaký projekt, v hlavní liště je též zobrazena aktuální vrstva a manuál. V rámci uživatelského rozhraní jednotlivých obrazovek je též indikován aktuální stav systému. U tzv. learn funkce je např. indikována aktuální hodnota parametru a je možné naslouchání novým hodnotám zastavit. V seznamu presetů je zvýrazněn aktuální preset. Kritérium viditelnosti stavu systému desktopové uživatelské rozhraní splňuje.

U hardwarového kontroleru je zobrazena volba aktuální obrazovky s pomocí LED podsvícení tlačítek, uživatel však nevidí textové vyjádření aktuální obrazovky. Aktuální manuál a vrstva jsou indikovány LED podsvícením u tlačítek představujících jednotlivé manuály. U jednotlivých obrazovek je též indikován aktuální stav pokud je to možné. Stav aplikace je převážně vyjádřen na LED prstencích umístěných okolo otočných enkodérů a LED podsvícením tlačítek.

4.1.2 Match between system and the real world - Shoda mezi systémem a reálným světem

Terminologie využitá v aplikaci se běžně používá v prostředí elektronických hudebních nástrojů a studiové tvorby. Pro zkušenějšího uživatele by neměl být problém se v uživatelském rozhraní zorientovat. Standardní chování je implementováno např. u funkce learn, kdy dochází ke změně konfigurace v prostředí třetí strany. Uživatelské prostředí hardwarového kontroleru pracuje pouze s názvy definovanými uživatelem.

4.1.3 User control and freedom - Ovládání a svoboda

U desktopového uživatelského rozhraní lze učinit krok zpět v případě editace konfigurace, dialogu pro zadávání názvu, obrazovky načítání projektu apod. V rozhraní není zahrnuto obecné tlačítko pro krok zpět, tzv. undo. Díky použití knihovny NGXS představující framework pro správu stavu aplikace by implementace takového tlačítka znamenala jednoduchou úpravu.

Jelikož je v rámci uživatelského rozhraní hardwarového kontroleru předpoklad, že uživatel bude systém ovládat úderně stručnými a frekventovanými akcemi, nebyla zde funkce pro krok zpět implementována. Výjimku představuje historie naposledy použitých presetů, kdy se lze při chybné volbě presetu rychle vrátit zpět, nebo si navolit předem více presetů.

4.1.4 Consistency and standards - Konzistence a standardizace

Toto kritérium v případě této aplikace splývá s kritériem shody mezi systémem a reálným světem (viz kapitolu 4.1.2).

4.1.5 Error prevention - Prevence chyb

V desktopovém uživatelském rozhraní se uplatňuje prevence chyb např. ve formulářích formou validace formulářových prvků, zakázáním akcí pro nedostupné VST pluginy apod. Položky hlavního menu aplikace se mění dynamicky dle stavu aplikace.

Jelikož uživatelské rozhraní hardwarového kontroleru musí být velmi stabilní, není zde připuštěno zobrazení chybové hlášky. Zamezení vstupu do neplatného stavu je provedeno přes odepření možnosti stisku tlačítek. V rámci uživatelského rozhraní se pracuje pouze s daty, ze kterých byly odstraněny nerelevantní položky (např. nedostupné VST pluginy).

4.1.6 Recognition rather than recall - Rozpoznání místo vzpomínání

V rámci desktopového uživatelského rozhraní jsou všechny prvky zřetelně označeny. U obrazovky editace presetu by mohla být lépe vyřešena volba stránky mapování parametrů tak, aby bylo možné přímo přejít na tvorbu nové stránky mapování.

Uživatelské rozhraní hardwarového kontroleru neposkytuje popis aktuální obrazovky, uživatel se může orientovat jen podle LED podsvícení tlačítek pro navigaci nebo detailů zobrazení v oblasti displeje. V tomto ohledu není kritérium splněno, u hudebního nástroje se však předpokládá velmi časté užívání, díky kterému si lze jednoduché akce zapamatovat poměrně snadno a rychle. U velké části MIDI kontrolerů není aktuální stav indikován žádným způsobem, u tohoto řešení je aktuální stav aplikace zobrazen za použití LED podsvícení a displeje.

4.1.7 Flexibility and efficiency of use - Flexibilní a efektivní použití

U desktopového uživatelského rozhraní je k dispozici tlačítko pro duplikaci presetů a stránek mapování parametrů pro zvýšení efektivity při editaci konfigurace. Zadávání hodnot parametrů VST pluginů usnadňuje též tzv. funkce learn, díky které lze hodnoty určit jednoduše přímo v prostředí VST pluginu.

Jedním z požadavků na uživatelské rozhraní hardwarového kontroleru je právě efektivita použití, celé uživatelské rozhraní je tedy navrženo pro dosažení vysoké ergonomie ovládání. Výsledná efektivita je též silně ovlivněna konfigurací zadanou uživatelem.

4.1.8 Aesthetic and minimalist design - Estetický a minimalistický design

Desktopové uživatelské rozhraní disponuje jednoduchým designem za použití frameworku Angular Material. Tento framework je často používaný při tvorbě webových a mobilních aplikací, uživatel by neměl mít problémy s orientací. Jednotlivé obrazovky obsahují jen relevantní prvky.

Hardwarový kontroler poskytuje pouze velmi omezené možnosti pro tvorbu uživatelského rozhraní. Navržené rozhraní proto jednoduše splňuje toto kritérium.

4.1.9 Help users recognize, diagnose, and recover from errors - Schopnost zotavení z chyb

V rámci desktopového uživatelského rozhraní je u formulářových prvků zobrazen popis chyby. Obecná chyba je zobrazena s pomocí modálního okna, nebo samostatné stránky. Veškeré chyby jsou zobrazeny v přirozeném jazyce.

Na displeji hardwarového kontroleru nejsou zobrazovány žádné chyby.

4.1.10 Help and documentation - Nápověda a návody

Pro uživatele může být těžké se v aplikaci zorientovat, jelikož aplikace neposkytuje žádné nápovědy ani legendy. V případě hardwarového kontroleru není zahrnuta žádná nápověda. Pro lepší orientaci uživatele by mohla být aplikace doplněna o použití tzv. tooltip prvků, kdy je nápověda zobrazena při umístení kurzoru nad ovládací prvek. Vhodné by bylo také poskytnout uživatelskou příručku, případně audiovizuálního průvodce.

4.1.11 Zhodnocení

Mezi problémy, se kterými by se mohl uživatel setkat, patří horší orientace v prostředí hardwarového kontroleru způsobená absencí jasné indikace stavu systému a funkce ovládacích prvků. Uživatel si musí zapamataovat funkce jednotlivých ovládacích prvků na globální úrovni i na úrovni jednotlivých obrazovek, jelikož účel ovládacích prvků nelze ze zobrazených údajů jasně určit. Zároveň není k dispozici náhled na globální stav systému, uživatel vždy vidí jen část stavu (např. na základě aktuálně zvolené vrstvy), což pro něj může být matoucí. U desktopové aplikace se může uživatel ztratit kvůli absenci nápovědy.

4.2 Uživatelské testování

Každý uživatel byl nejprve seznámen se systémem. Následně byl požádán o provedení testovacího scénáře. Reakce uživatele byly zaznamenány pomocí programu pro snímání obsahu obrazovky a mobilního telefonu. Zaznamenané chování uživatele v rámci testovacího scénáře bylo následně podrobeno analýze.

4.2.1 Testovací scénář

Pro účely testování byl použitý následující testovací scénář.

V desktopové aplikaci proveďte následující úkony:

1. Načtěte projekt programu Bidule.
2. Založte novou stránku mapování parametrů pro některý z virtuálních nástrojů.
3. Vytvořte vlastní kategorii presetu a nový preset. Použijte stránku mapování parametrů vytvořenou v předchozím kroku.

Poté prostřednictvím hardwarového kontroleru proveďte následující kroky:

1. Zvolte nově vytvořený preset.
2. Změňte hodnotu některého parametru virtuálního nástroje.

3. Přepněte na druhou vrstvu a vyberte jiný preset.
4. Změňte jakýkoliv parametr některého z globálních efektů.
5. Na první vrstvě přepněte na libovolný preset.
6. Přepněte na druhou vrstvu a změňte hodnotu parametru virtuálního nástroje.
7. Na první vrstvě načtěte původní preset z historie.

4.2.2 Uživatel č. 1

Prvním testovaným uživatelem byl muž, 28 let, zabývající se hrou na klávesy a kytaru převážně v zábabové kapele. Uživatel neměl předchozí zkušenosti s použitím virtuálních hudebních nástrojů, měl však zkušenosti s použitím hardwarových nástrojů.

4.2.2.1 Průběh testování

První část scénáře - desktopová aplikace.

1. Uživatel neměl problémy s výběrem a načtením uloženého projektu programu Bidule.
2. Při editaci stránek mapování parametrů virtuálního nástroje měl problém uživatel s orientací a snažil se nejprve vybrat již existující stránku mapování. Po vybrání ovládaného parametru uživatel zapomněl určit strategii pro mapování parametru.
3. Uživatel měl problém s navigací. Z popisu kroku testovacího scénáře byl zmatený a nevěděl, co má dělat. Při výběru kategorie omylem klikl na tlačítko pro editaci názvu kategorie. Po vytvoření presetu byl zmatený, jelikož tlačítko pro editaci presetu nebylo aktivní.

Druhá část scénáře - hardwarový kontroler.

1. Uživatel správně zvolil nově vytvořený preset.
2. Uživatel přepnul zobrazení na správnou obrazovku, ale zvolil jiný preset z historie. To bylo pro něj matoucí a nevěděl, jak se vrátit do předchozího stavu. Volbu požadovaného nástroje vyřešil rychle opakováním postupu z předchozího kroku. Poté již uživatel nevěděl, kde nastavení najde a přepl na obrazovku pro ovládání vstupních efektů vrstvy, o které se domníval, že slouží k ovládání globálních efektů. Rychle si však chybu uvědomil, přepl znovu na správnou obrazovku a v pořádku změnil hodnotu parametru virtuálního nástroje.

3. Dvojklik provedený uživatelem byl moc pomalý. Při druhém pokusu již došlo k přepnutí na správnou vrstvu. Volba presetu proběhla bez problémů. Uživatel též ovládal hlasitosti vrstev, kde pro něj bylo matoucí použití statických knobů.

4. Uživatel si spletl pojmy globální efekt a efekt pro vrstvu. Místo globálního efektu nastavil hodnotu parametru lokálního efektu. Jelikož v té době byla první vrstva zeslabená, nevyjádřil údiv.

5. Uživatel správně provedl přepnutí presetu.

6. Při volbě vrstvy nejprve uživatel provedl dvojklik opět moc pomalu. Změna parametru nástroje proběhla bez problému.

7. U posledního kroku uživatel zapomněl zvolit ovládanou vrstvu. Navíc zvolil obrazovku pro výběr presetů namísto obrazovky ovládání parametrů nástroje, ve které je zahrnuta historie presetů. Svji chybu si rychle uvědomil a napravil, byl ale zmatený, jelikož ovládaná vrstva byla zeslabená a tak se zdálo, že se akce nijak neprojevila.

4.2.2.2 Zpětná vazba uživatele

Uživatel uvedl následující seznam připomínek k uživatelskému prostředí hardwarového kontroleru.

- Příliš krátký čas pro rozlišení vícenásobného stisknutí tlačítek.
- Položky v historii presetů by se neměly měnit a měly by odpovídat kategoriím dle uspořádání na obrazovce přepínání presetů.
- V rámci volby presetů by mělo dojít ke změně aktivního presetu již při výběru kategorie. Toto chování je standardní u nástrojů od společnosti Roland.
- Není k dispozici jednotné zobrazení aktuálních presetů pro všechny vrstvy manuálů.
- Chování tlačítek není zřejmé, není snadné si ho zapamatovat.

4.2.2.3 Zhodnocení

Jak již uživatel poznamenal v rámci zpětné vazby, ovládací prvky prostředí hardwarového kontroleru postrádají nápovědu a bylo pro něj velmi těžké si zapamatovat jejich funkci. Uživatel byl často zmatený a nevěděl, jakým způsobem lze zvolit požadovanou obrazovku. Též indikace aktuálního stavu systému byla pro uživatele nedostatečná. Indikace je sice přítomná, ale pro uživatele je těžké si zapamatovat, jakou informaci vyjadřuje.

Zároveň pro uživatele bylo problematické vícenásobné stisknutí tlačítek, jelikož je nedokázal opakovaně stisknout s dostatečnou rychlostí.

Uživatel byl též prvotně zmatený návrhem přepínání presetů, které je odlišné od nástroje, na který je zvyklý.

U desktopové aplikace měl uživatel problémy s navigací a pochopením architektury systému. Tyto problémy můžou být způsobeny absencí zkušeností s virutálními hudebními nástroji a neznalostí terminologie.

Uživatel si též nebyl jistý kroky netriviálních procesů.

4.2.3 Uživatel č. 2

Druhým testovaným uživatelem byl muž, 22 let, zabývající se hrou na klávesy a kytaru v autorských projektech. Uživatel měl bohaté zkušenosti s využitím virtuálních hudebních nástrojů, avšak pouze v prostředí pro studiovou produkci hudby (DAW). Zároveň byl uživatel zkušený i v oblasti hardwarových hudebních nástrojů.

4.2.3.1 Průběh testování

První část scénáře - desktopová aplikace.

1. Krok načtení projektu programu Bidule provedl uživatel bez problému.

2. Na obrazovce pro editaci stránek mapování parametrů virtuálního nástroje se uživatel ihned snažil vybrat již existující stránku mapování. Po vytvoření nové stránky uživatel zapomněl zadat další informace. Při zadávání dalších informací uživatel nevěděl, jak lze nakonfigurovat mapování pro nově vytvořenou položku a měl celkově problémy s orientací v rámci procesu definice vlastního mapování parametrů. U použití funkce pro výběr ovládaného parametru uživatel předpokládal, že lze vytvořit více položek stránky mapování postupným ovládáním různých parametrů v uživatelském prostředí virtuálního nástroje.

3. Při vytváření presetu uživatel nejprve načetl obrazovku z předchozího kroku. Brzy zvolil správnou položku menu a vytvořil nový preset. Uživatel byl zmatený kvůli neaktivnímu tlačítku pro editaci presetu. Konfigurace presetu dále již proběhla bez problému.

Druhá část scénáře - hardwarový kontroler.

1. Uživatel správně zvolil kategorii, ale byl zmatený ořezáním příliš dlouhého názvu presetu. Nevhodný znak šipky indikující ořezání navíc u uživatele vytvořil mylný dojem, že lze seznam posunout. Uživatel měl navíc problém s oddělením ovládání kategorií a presetů.

2. V druhém kroku měl uživatel problém s přepnutím na požadovanou obrazovku, jelikož si způsob ovládání pamatoval pouze matně. Se změnou hodnoty parametru nástroje pak neměl problém.

3. Uživatel si vybavil způsob ovládání aktuální vrstvy, ale měl problém s přepnutím na obrazovku pro volbu presetu. Samotná volba presetu pak již proběhla v pořádku.

4. Uživatel si ihned vybavil způsob přepnutí na obrazovku efektů. Měl však problém s vícenásobným stisknutím tlačítka, které nebylo provedeno s dostatečnou rychlostí.

5. S přepnutím presetu na první vrstvě neměl uživatel problémy.

6. Uživatel postupoval nejistě ve strachu, že způsobí neočekávanou změnu. Nakonec však krok změny parametru nástroje provedl správně.

7. Uživatel měl problémy s využitím historie použitých presetů. Nepamatoval si, jak funkce funguje, a krok nedokončil.

4.2.3.2 Zpětná vazba uživatele

Pro uživatele by byla podstatným zlepšením možnost zadávat položky mapování parametrů virtuálních nástrojů v uživatelském rozhraní pluginu najednou.

4.2.3.3 Zhodnocení

Při ovládání systému za pomoci hardwarového kontroleru uživatel často nevěděl, jak zajistí přepnutí na požadovanou obrazovku. Také měl problémy s vícenásobným stisknutím tlačítek, které občas stiskl příliš pomalu. V průběhu testu se jeho orientace v ovládacích prvcích zlepšovala. V závěru testování byl však zmatený a nebyl schopný určit celkový stav systému.

V uživatelském prostředí desktopové aplikace byl uživatel při složitějších procesech dezorientovaný a nevěděl, jaké kroky je třeba provést a jak docílit provedení požadovaných úkonů.

4.3 Výsledky testování

Uživatelské testování potvrdilo problémy zjištěné v rámci heuristické analýzy a ukázalo na další podstatné problémy.

V následujících podkapitolách jsou rozebrány jednotlivé problémy a je navrženo řešení pro jejich eliminaci.

4.3.1 Uživatelské rozhraní HW kontroleru

4.3.1.1 Vícenásobné stisknutí tlačítek

Uživatelé měli často problém s vícenásobným stisknutím tlačítek, jelikož bylo třeba tlačítko opakovaně stisknout s vysokou frekvencí. Uživatel opakoval stisknutí moc pomalu a tak aplikace vyhodnotila vstup jako jednoduché stisknutí tlačítka.

Problém lze vyřešit triviálně nastavením delšího časového okénka při zpracovávání stisknutí tlačítek.

4.3.1.2 Omezená indikace stavu

V rámci provádění testovacího scénáře měli uživatelé často problém s určením stavu systému. Stav je v omezené míře indikován, uživatel však neví, jaký mají informace význam. Tento problém byl zjištěn již při heuristické analýze, je však způsobený krátkou dobou používání systému a lze tak předpokládat, že při intenzivnějším používání systému problém přestane být aktuální.

Chybí však možnost hromadného zobrazení aktuálních presetů pro všechny vrstvy a další souhrnná zobrazení celého stavu systému.

Úpravou pro lepší orientaci uživatele by mohlo být zavedení obrazovky pro souhrnné zobrazení informací.

4.3.1.3 Určení funkce ovládacích prvků

Podobně jako u problému omezené indikace stavu způsobeného absencí popisu zobrazované informace, dochází k problému i při určení účelu ovládacích prvků.

Jednoduchým řešením je doplnění nápovědy přímo na povrch hardwarového kontroleru.

4.3.1.4 Nevhodný znak pro indikaci ořezání názvů

Na displeji kontroleru lze použít pouze omezenou sadu znaků. Znak vlnovky, který je použitý pro indikaci ořezání názvu, kontroler neumí zobrazit.

Řešením je použití jiného znaku vhodného pro indikaci ořezání názvu, který bude možné zobrazit na displeji hardwarového kontroleru.

4.3.1.5 Konzistence přepínání presetů

Problém nastává při přepínání presetů. Uživatel je zmatený, jelikož může preset vybrat buď z obrazovky přepínání presetů, nebo z obrazovky ovládání parametrů načítáním záznamu historie.

Jelikož systém cílí na dlouhodobé uživatele a počítá s křivkou učení, bylo toto chování navrženo záměrně. V praxi přinese velkou výhodu možnost rychlého přepínání presetů přímo ze stránky ovládání parametrů nástroje.

Zároveň pro uživatele může být matoucí způsob výběru presetu z kategorie, kdy k přepnutí zvuku dojde až při volbě daného presetu a ne po volbě kategorie, jak je tomu např. u některých nástrojů společnosti Roland. Toto chování bylo též navrženo záměrně, neboť dává uživateli větší svobodu při procházení presetů.

4.3.2 Uživatelské rozhraní desktopové aplikace

4.3.2.1 Nedostatečná nápověda

Jelikož není dostupná žádná nápověda, uživatel má často problém s určením příčiny nedostupnosti ovládacích prvků, nebo jejich účelu.

Řešením je doplnění vhodného popisu a nápovědy k ovládacím prvkům, případně zobrazení důvodu nedostupnosti některých prvků.

4.3.2.2 Nízká úroveň dekompozice rozhraní

V rámci jedné obrazovky lze provést i složitější procesy. Uživateli dělá problém se zorientovat v postupu složitějších procesů, jako je např. založení nové stránky mapování parametrů pluginu. Uživatel snadno vynechá některé kroky procesu, nebo je pro něj těžké určit, v jaké části obrazovky může akci provést.

Řešením je dekompozice uživatelského rozhraní na menší části, které lze orchestrovat např. pomocí průvodce, či přepínatelných záložek.

4.3.2.3 Způsob zadávání položek stránky mapování parametrů

Implementovaný způsob zadávání položek stránek mapování parametů pluginů je sice funkční, ale mohl by být efektivnější. Řešením je implementace možnosti určení ovládaných parametrů hromadně přímo v uživatelském prostředí pluginu.

Závěr

Cílem této práce bylo vytvoření prostředí pro živou hudební produkci za použití virtuálních hudebních nástrojů.

Nejprve byla provedena analýza požadavků a průzkum existujících řešení. Na základě analýzy bylo navrženo vlastní řešení kombinující počítač a připojený kontroler s prezentačními prvky. Aplikace podle návrhu integruje hardwarový kontroler a tzv. VST host prostředí, které zajišťuje provoz virtuálních nástrojů a zpracování zvukového signálu. V rámci analýzy byly též provedeny testy kompatibility jednotlivých komponent. Ve fázi návrhu byl mimo jiné vytvořen doménový model aplikace a drátěné modely uživatelského rozhraní desktopové aplikace a hardwarového kontroleru.

V rámci implementace bylo ve velké míře využito reaktivní programování. Vznikla vrstva pro nízkoúrovňovou komunikaci s USB MIDI zařízeními a komunikaci přes protokol Automap včetně modelu uživatelského rozhraní hardwarového kontroleru. Testování výsledné aplikace probíhalo pouze na úrovni uživatelských testů za účasti vybraných hudebníků. Při testování byly nalezeny problémy, které byly analyzovány a případně bylo navrženo řešení pro jejich eliminaci.

Výsledná aplikace splňuje požadované vlastnosti a celé prostředí lze použít v praxi pro živá hudební vystoupení. Aplikace by dále mohla být rozšířena o pokročilejší funkce užitečné při živých vystoupeních hudebníků. Zároveň by bylo vhodné doplnit desktopové uživatelské rozhraní aplikace o nápovědu.

Literatura

[1] Kogan, R.: Brief History of Electronic and Computer Musical Instruments. 2008.

[2] Leider, C. N.: *Digital audio workstation*. McGraw-Hill, Inc., 2004.

[3] Steinberg: Virtual Studio Technology - Plug-In Specification 2.0 – Software Development Kit. [cit. 2019-11-15]. Dostupné z: http://jvstwrapper.sourceforge.net/vst20spec.pdf

[4] Steinberg: Steinberg Audio Streaming Input Output Specification - Development Kit 2.2. [cit. 2019-11-15]. Dostupné z: http://read.pudn.com/downloads119/sourcecode/multimedia/audio/506331/ASIOSDK2/ASIO%20SDK%202.2.pdf

[5] Wright, M.: The Open Sound Control 1.0 Specification. 2002, [cit. 2019-11-16]. Dostupné z: http://opensoundcontrol.org/spec-1_0

[6] Association, T. M. M.: The Complete MIDI 1.0 Detailed Specification. [cit. 2019-11-16]. Dostupné z: https://www.midi.org/downloads?task=callelement&format=raw&item_id=92&element=f85c494b-2b32-4109-b8c1-083cca2b7db6&method=download

[7] Novation DMS: SLMKII MIDI Programmer's Reference.

[8] Ashour, G.; Brackenridge, B.; Tirosh, O.; aj.: Universal Serial Bus Device Class Definition for MIDI Devices. [cit. 2019-11-16]. Dostupné z: https://www.usb.org/sites/default/files/midi10.pdf

[9] Young, G.: CQRS Documents by Greg Young. 2011, [cit. 2019-11-21]. Dostupné z: https://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf

[10] Fowler, M.: Command Query Separation. 2005, [cit. 2019-11-21]. Dostupné z: https://martinfowler.com/bliki/CommandQuerySeparation.html

[11] libusb-1.0 API Reference. [cit. 2019-11-21]. Dostupné z: http://libusb.sourceforge.net/api-1.0/

[12] Clow, M.: Observers, Reactive Programming, and RxJS. In *Angular 5 Projects*, Springer, 2018, s. 291–307.

[13] Prasanna; Dhanji, R.: Dependency injection. 2009.

[14] Facebook Inc.: Flux: In-Depth Overview. 2019, [cit. 2019-11-22]. Dostupné z: https://facebook.github.io/flux/docs/in-depth-overview

[15] Hsu, J.: What the Flux? (On Flux, DDD, and CQRS). [cit. 2019-11-22]. Dostupné z: https://jaysoo.ca/2015/02/06/what-the-flux/

[16] Mansilla, S.: *Reactive Programming with RxJS 5: Untangle Your Asynchronous JavaScript Code*. Pragmatic Bookshelf, 2018.

[17] Abramov, D.; the Redux documentation authors: Redux documentation: Normalizing State Shape. 2018, [cit. 2019-11-22]. Dostupné z: https://redux.js.org/recipes/structuring-reducers/normalizing-state-shape

[18] Reenskaug, T. M. H.: The original MVC reports. 1979.

[19] Angular - Introduction to modules. [cit. 2019-11-25]. Dostupné z: https://angular.io/guide/architecture-modules

[20] Martin, R. C.: The open-closed principle. *More C++ gems*, ročník 19, č. 96, 1996: str. 9.

[21] Nielsen, J.: 10 usability heuristics for user interface design. *Nielsen Norman Group*, ročník 1, č. 1, 1995.

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

ASCII American Standard Code for Information Interchange

ASIO Audio streaming input output

CRUD Create, Read, Update, Delete

CQRS Command Query Responsibility Segregation

CQS Command Query Separation

DAW Digital audio workstation

IPC Inter process communication

JSON JavaScript Object Notation

MVC Model, View, Controller

OSC Open sound control

VST Virtual studio technology

VSTi Virtual studio technology instrument

XML Extensible markup language

Příloha **B**

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `test` ..... soubory spojené s testováním
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF