VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÉ ROZHRANÍ PRO OPRAVU AUTOMATICKÉHO PŘEPISU A TAGOVANÍ

WEB INTERFACE FOR HUMAN CORRECTIONS OF AUTOMATIC TRANSCRIPT AND TAGGING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN PLHAL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Plhal Jan**
Program: Informační technologie
Název: **Webové rozhraní pro opravu automatického přepisu a tagovaní**
Web Interface for Human Corrections of Automatic Transcript and Tagging
Kategorie: Uživatelská rozhraní

Zadání:

1. Nastudujte workflow kontroly přepisů radiové VHF komunikace pilot-věž.
2. Nastudujte základy moderních webových frameworků (zejména frontend část). Nastudujte základy UX a UI, zaměřte se na tagování, úpravu textu, obrázky a audio.
3. Navrhněte a implementujte webové komponenty, které umožní rychlé a efektivní opravu přepisu, tagování slov a celých segmentů, navigaci v audiu, zobrazování obrázků s pomocnými informacemi.
4. Testujte tyto komponenty na odpovídající skupině uživatelů. Ověřte přívětivost a snadnost používaní rozhraní. Upravte váš návrh a iterujte.
5. Diskutujte dosažené cíle a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Dále dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a část bodu 3 ze zadání.

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

Vedoucí práce: **Szőke Igor, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 27. dubna 2022

Abstrakt

Cílem této práce je návrh a implementace webové aplikace pro opravu automatického přepisu a anotace rádiové VHF (very high frequency) komunikace pilot-věž. Základní inspirací je ATCO2 anotační služba SpokenData firmy ReplayWell, jež byla od základu přepracována do uživatelsky přívětivější podoby s řadou uživatelsky přizpůsobitelných prvků. Na základě nastudovaných principů návrhu dobře použitelného uživatelského prostředí a poznatků z průzkumu anotačních aplikací byl proveden návrh funkcionality a také grafického rozhraní aplikace formou drátěného modelu. Navrženou aplikaci se prostřednictvím zvolených technologií podařilo úspěšně implementovat. Použitelnost aplikace byla ověřována a vylepšována pomocí uživatelského testování a aplikace byla také nasazena na web SpokenData.

Abstract

The goal of this thesis is the design and implementation of a web application for human corrections of automatic transcript and annotation of VHF (very high frequency) pilot-tower radio communication. The main inspiration is the ATCO2 SpokenData annotation service from the company ReplayWell, which was remade from the ground up into a more user-friendly form with a number of user customizable elements. Based on studied principles of designing a well usable user interface and insights gained from research of annotation applications was created a functional design and also a graphical design of the application in the form of a wireframe. The designed application was successfully implemented using the selected technologies. The usability of the application was verified and improved with the help of user testing and the application was also deployed on the SpokenData website.

Klíčová slova

UI, UX, použitelnost, React, anotace, řízení letového provozu, webová aplikace, Bootstrap, testování použitelnosti, SUS

Keywords

UI, UX, usability, React, annotation, air traffic control, web application, Bootstrap, usability testing, SUS

Citace

PLHAL, Jan. *Webové rozhraní pro opravu automatického přepisu a tagovaní.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Webové rozhraní pro opravu automatického přepisu a tagovaní

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho, Ph.D. Další informace mi poskytl Jan Kukuczka a s nasazením aplikace pro testování mi asistoval Ing. Josef Žižka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Plhal
7. května 2022

Poděkování

Děkuji vedoucímu práce panu Igorovi Szőkeovi za kvalitní konzultace a vstřícný přístup v průběhu semestru. Taktéž děkuji panu Janu Kukuczkovi za předvedění jeho implementace anotační aplikace a Ing. Josefovi Žižkovi za asistenci při nasazení aplikace pro testování.

Obsah

Kapitola 1

Úvod

Cílem této práce je návrh a implementace webové aplikace pro opravu automatického přepisu a anotaci (přidávání dodatečných informací) rádiové VHF (very high frequency) komunikace pilot-věž. Základní inspirací je ATCO2 anotační služba SpokenData firmy ReplayWell, jež byla od základu přepracována do uživatelsky přívětivější podoby s řadou uživatelsky přizpůsobitelných prvků.

ATCO2[1] je projekt zaměřený na tvorbu platformy pro sběr, organizaci a předzpracování hlasové komunikace řízení leteckého provozu. Hlasové záznamy jsou strojovým učením automaticky přepsány do textové podoby a anotovány řadou značek pro klíčové části v komunikaci, identifikaci mluvčích apod. Toto strojové učení však není dokonalé a pro jeho natrénování je potřeba desítky hodin manuálně opravených dat – právě toto je účelem vyvíjené aplikace.

Začátek následující kapitoly se zabývá zásadami pro návrh dobře použitelného uživatelského prostředí, které byly v rámci návrhu následovány. Následně je zkoumána stávající aplikace anotační služby SpokenData pro zjištění funkcionality, jež je třeba v nové aplikaci zachovat, a použitelnostních slabin, které lze eliminovat. V rámci průzkumu trhu jsou pak zkoumány další anotační aplikace pro získání inspirace. Na závěr druhé kapitoly je na základě získaných znalostí proveden návrh cílové funkcionality výsledné aplikace.

Třetí kapitola se zabývá srovnáním moderních frameworků pro JavaScript a kaskádové styly, na základě kterého je vybrán ten nejvhodnější pro účely této práce. Ke konci kapitoly jsou pak popsány knihovny třetích stran vybrané pro urychlení implementace navržené funkcionality.

Čtvrtá kapitola obsahuje grafický návrh cílové podoby aplikace a detailní popis jednotlivých návrhů modulů.

V páté kapitole je popsán vývojový design aplikace a spojené technologie, způsob fungování a technologie backendu SpokenData, se kterým byla aplikace propojena a stručný popis vybraných implementačních detailů.

Šestá kapitola se zabývá způsoby testování použitelnosti s účastníky v rámci testovacích sezení a následně popisuje samotný průběh testování aplikace a jeho vyhodnocení na základě zjištěných poznatků. Na konci této kapitoly jsou detailně popsány výhledy na budoucí vývoj.

Závěrem jsou zrekapitulovány a zhodnoceny dosažené výsledky a stručně shrnuty diskutované možné směry budoucího vývoje.

[1] https://www.atco2.org/

Kapitola 2

Shrnutí dosavadního stavu a návrh

Začátek této kapitoly se zabývá principy návrhu dobře použitelného uživatelského prostředí. Poté je zkoumána současná aplikace anotační služby SpokenData, jež bude v rámci této práce přetvořena a také další anotační aplikace v rámci průzkumu trhu, z nichž bude čerpána inspirace pro návrh. Na závěr kapitoly je na základě získaných znalostí proveden návrh funkcionality výsledné aplikace.

2.1 Použitelnost

V této části jsou uvedeny principy návrhu dobře použitelného webového prostředí, resp. uživatelského prostředí obecně, kterými se budu při vývoji řídit a v rámci testování jejich splnění ověřovat.

2.1.1 Přístup uživatele

Vývojář typicky na jím vyvíjenou aplikací nemá zcela objektivní pohled, není schopný se na ni kriticky podívat „z venku“. Proto je dobré si ujasnit, jak k aplikaci bude pravděpodobně přistupovat běžný uživatel a návrh tomu přizpůsobit. Zdrojem myšlenek v této části je kniha Don’t Make Me Think [4] od Steva Kruga.

Představa vs. realita

Při vývoji aplikace můžeme mylně nabýt dojmu, že si uživatel pečlivě projde veškerý obsah, přečte uvedený text, zváží možnosti pro dosažení svého cíle a teprve potom začne konat. Ve většině případů však obrazovku pouze v rychlosti skenuje očima, zaměří se na odkazy či tlačítka s klíčovými slovy, která připomínají to, co v aplikaci hledá a zkusí na ně kliknout. Tento přístup je však logický – uživatel ví, že všechen text číst nemusí a snaží se úkol provést co nejrychleji. Nejde o věc specifickou pro uživatelská prostředí, ale platí to pro veškerý obsah a schopnosti skenováním hledat věci, které nás mohou dovést k řešení nějakého problému, nebo nás zkrátka zajímají, využíváme každý den.

Jakmile uživatel najde něco připomínající to, co hledá, místo dalšího hledání zkrátka na danou věc zkusí kliknout. V první řadě je to rychlejší a při špatné volbě jsou důsledky typicky minimální – stačí se pár kliknutími vrátit zpět, nebo jen kliknout zase na něco dalšího. Navíc optimalizování a zvažování více možnosti je náročné a při hůře navržených aplikacích ani nemusí zvýšit naše šance na úspěch. Hádání a zkoušení je jednodušší a přináší element šance, což práci dělá zábavnější v případě, že věc odhadneme dobře.

Pochopení aplikace

Dalším faktorem je, že lidé často věci používají bez skutečného pochopení, jak vlastně fungují. Jen málo lidí si dá práci se čtením instrukcí, místo toho se učí praxí a vytváří si vlastní představy o tom, jak a proč věci fungují. Nicméně uživatelé jsou takto schopni pracovat, a to dokonce efektivně i přesto, že prvky využívají jinak, než bylo původně zamýšleno.

Takto to má do určité míry každý uživatel, nehledě na jeho zdatnosti s počítači. Správný postup práce je důležitý pro vývojáře dané aplikace, ostatní zajímá hlavně to, aby ji byli schopni používat a pokud něco funguje, lepší způsob už nehledají. Je tedy správné pochopení aplikace vůbec důležité, pokud je uživatel schopen pracovat i bez něj? Ano, protože to zvýší šanci, že najde to, co hledá, zlepší to pochopení částí aplikace v kontextu celku a také uživatelův pocit z práce s aplikací, protože opravdu rozumí tomu, co a proč dělá.

2.1.2 Behaviorální vzory

Ačkoliv je každý z uživatelů svým způsobem unikátní, jejich chování je většinou předvídatelné. Lze ho popsat určitými behaviorálními vzory, které pokud při návrhu prostředí vezmeme v potaz, značně v něm uživatelům usnadníme dosáhnutí jejich cílů. Zdrojem myšlenek v této sekci je kniha Designing Interfaces: Patterns for Effective Interaction Design [6] od Jenifer Tidwell.

Bezpečný průzkum

Uživatel by měl mít možnost se s prostředím učit zkoušením různých funkcí, aniž by za své experimenty nesl následky. Pokud tuto možnost má, pravděpodobně se s prostředím naučí pracovat podstatně rychleji a s pozitivnějším přístupem. Naopak při absenci této možnosti může jeho experiment vést k částečné či úplné ztrátě postupu, což může způsobit značnou frustraci a ztrátu motivace prostředí dále zkoumat.

Příkladem tohoto chování může být návštěvník webu nějaké společnosti, který kliká na různé odkazy a prozkoumává, co na stránce je. Přitom spoléhá na to, že ať už se prokliká na stránce kamkoliv, vždy se může pár kliknutími na tlačítko „zpět“ vrátit na domovskou stránku. Pokud však stránka na tlačítko „zpět“ reaguje nepředvídatelně, uživatele to může zmást a dezorientovat, což může vést k úplné ztrátě jeho zájmu.

Dalším příkladem průzkumu prostředí je uživatel grafického editoru, který zkouší aplikovat efekty na fotografii. Po prohlédnutí výsledku se rozhodne, že mu daný efekt nevyhovuje a pomocí tlačítka pro vrácení akce fotografii jednoduše vrátí do původního stavu, odkud může znovu zkoušet jiné efekty, dokud nedosáhne žádaného výsledku.

Okamžité uspokojení

Je typické, že lidé chtějí, aby výsledky jejich činů byly okamžité. Pokud je nový uživatel ve svých prvních chvílích práce s prostředím schopný dosáhnout žádoucích výsledků, bude mít silnější motivaci s prostředím dále pracovat, i kdyby další operace byly obtížnější. Získá totiž větší pocit sebevědomí a důvěry v aplikaci, než kdyby nad operací musel nejprve pracně přemýšlet.

Toho lze dosáhnout například identifikací nejpravděpodobnějšího prvního kroku práce nového uživatele s prostředím, který pro něj uděláme co nejsnazší, například nápovědou, nebo jiným způsobem, kterým jej k jeho provedení navedeme. To však také naopak znamená, že bychom jej od tohoto kroku neměli žádným způsobem oddalovat, ať už reklamami,

registrací, nebo dlouhými instrukcemi. Tím se totiž vzdáváme okamžitého uspokojení uživatele, a tak jej od aplikace odrazujeme.

Zvyklosti

Při časté práci s prostředím si uživatel časem vytvoří zvyklosti pro provádění nejčastějších operací. Nemusí pak nad jejich provedením přemýšlet a dělá je automaticky, což pochopitelně vede ke zvýšení jeho produktivity a také k lepšímu pocitu zabrání do pracovního procesu. Pokud se však uživatel pokusí ze zvyklosti provést nějakou operaci, která v dané situaci nefunguje, nebo dokonce provede něco destruktivního, je vytrhnut z pracovního procesu a musí o práci s prostředím opět přemýšlet, nemluvě o případné opravě následků.

Například kombinace klávesových zkratek `Ctrl+A`→`Ctrl+X`→`Ctrl+S` provede v textovém editoru Emacs posunutí kurzoru na konec dokumentu a následné uložení, zmáčknutí těchto zkratek se pro uživatele Emacsu stává zvyklostí. Jenže pokud tu samou kombinaci zmáčkneme v Microsoft Wordu, provede se výběr celého obsahu dokumentu, vyjmutí výběru (tedy celého dokumentu) a uložení. Proto je důležité následovat zavedené konvence a zajistit tak co největší konzistenci napříč aplikacemi.

Důležitá je však i konzistence v aplikaci samotné. Není výjimkou, že vývojáři aplikací zavedou úkon, který provede nějakou operaci, ovšem v jiném specifickém kontextu provede operaci odlišnou. To zcela jistě povede k omylům uživatelů, a to zejména těch zkušených, kteří úkon provádí automaticky ze zvyklosti.

Zvyklosti jsou také důvodem, proč potvrzovací dialogová okna často uživatele neochrání před nechtěnými operacemi. Pokud zavedeme dialogová okna pro často prováděné operace, jako je mazání nějaké položky, pro uživatele se jejich potvrzování stane zvyklostí a nechtěnou operaci automaticky potvrdí. Ve většině případů je tedy lepší dialogová okna zcela vynechat a raději například zavést robustní systém pro vrácení změn.

Prostorová paměť

Při manipulaci s objekty a dokumenty je často uživatelé později opět najdou tím, že si pamatují, kam je umístili, a ne podle jejich názvu.

Příkladem může být plocha desktopového operačního systému, kterou mnoho uživatelů používá pro uložení dokumentů, často používaných aplikací a dalších věcí. Mezi nimi se pak orientují pomocí prostorové paměti, například jejich umístěním do skupin, nebo pamatováním jejich relativní pozice vůči ostatním položkám. Tento jev má samozřejmě přesah i do reálného světa, například mnoho lidí má stoly zaskládané hromadou věcí, mezi kterými jsou však schopni snadno nalézt, co potřebují.

Je vhodné pozicovat dialogová tlačítka typu „OK“ a „Zrušit“ na předvídatelná místa, jelikož pro tyto prvky je prostorová paměť velmi silná. Jejich nekonzistentní umístění by pravděpodobně vedlo k omylům, jelikož uživatel popisy těchto tlačítek přestane číst. V komplexnějších aplikacích si pak uživatelé mohou pozice položek pamatovat relativně k ostatním, například nástroje v nástrojové liště, položky v menu apod. Je tedy třeba být obezřetný s reorganizací položek v rozhraní, jelikož tím můžeme narušit prostorovou paměť uživatelů a jejich schopnost orientace.

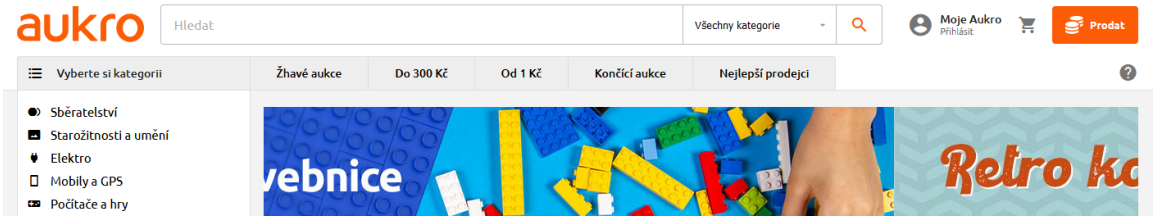
Poskytnutí prostoru, na kterém si uživatel může sám pozicovat a organizovat položky, je dobrým nápadem právě kvůli prostorové paměti. Pokud si uživatel umístí položky sám, je větší pravděpodobnost, že je následně znovu najde. Nicméně ne vždy je to vhodné, zejména při vysokém počtu položek.

2.1.3 Použitelnostní zásady designu prostředí

Tato sekce se zabývá principy z knihy Don't Make Me Think [4] od Steva Kruga pro vizuální návrh přehledného uživatelského prostředí, ve kterém se uživatel snadno zorientuje a nalezne, co potřebuje.

Designové konvence

Jedním z nejlepších způsobů, jak urychlit orientaci na webové stránce či aplikaci, je použití fungujících, časem ověřených konvencí. Ty upravují rozložení, vzhled i způsob fungování. Jako příklad lze uvést stránky určené pro online nakupování (ačkoliv některé z dále uvedených konvencí platí i pro stránky obecně), jako je online tržiště Aukro zobrazené na obrázku 2.1: vlevo nahoře očekáváme název webu, kterým se kliknutím můžeme vrátit na úvodní stránku, nahoře či na straně vlevo hlavní navigaci, vpravo nahoře ikonu nákupního košíku, která nás vezme do virtuálního košíku, do kterého si při prohlížení sortimentu přidáváme položky apod. Díky tomu z většiny odpadá potřeba se s daným prostředím učit a uživatel se může zaměřit na to, na čem záleží – na nákup samotný. Podobná sada konvencí existuje i pro většinu ostatních druhů webů a aplikací. Kdy má však smysl se od konvencí odchýlit?



Obrázek 2.1: Webová stránka online tržiště Aukro

Pokud se od zavedených konvencí chceme odchýlit, musíme si být jisti, že naše nové řešení je buď tak jasné, že jej není třeba vysvětlovat, nebo výhody řešení stojí za potřebu uživatele se ním naučit. Pokud je to možné, není na škodu poskytnout i konvenční alternativu.

Tvorba vizuální hierarchie

Je důležité, aby aplikace byla jasně vizuálně rozčleněna na logické celky, které jsou dále hierarchicky členěny na další podcelky. Uživatel tak při správně zvolených nadpisech může pro něj nepodstatné části s klidným svědomím ignorovat a dostat se tak rychleji k tomu, co hledá.

Pro správnou tvorbu vizuálních hierarchií je třeba následovat několik pravidel:

- Důležité části je třeba zvýraznit, ať už například větším fontem, tučným stylováním, větším odsazením od zbytku nebo jejich kombinací.
- Musí být zřejmé, který obsah k sobě patří. Toho lze docílit stejným stylováním, odsazením pod nadpis nebo jejich umístěním do jinak vizuálně oddělené sekce – například s odlišným pozadím, ohraničené rámečkem apod.
- Pro správnou reprezentaci hierarchie musí být části vnořené do nadřazeného celku vnořené i vizuálně, aby bylo zřejmé, co pod co patří. Toho lze dosáhnout pomocí

různých úrovní odsazení, velikostí nadpisů apod. Zároveň by například neměl nadpis jedné části zasahovat i nad sousední nesouvisející část.

Pokud správně zvolíme a zvýrazníme nejdůležitější části a vizuálně reprezentujeme logickou hierarchii, uživatel pak již nemusí pracně hledat klíčová slova a vytvářet si vlastní smysl orientace, vše je pro něj totiž nachystáno.

Potlačení vizuálního šumu

Příkladem vizuálního šumu mohou být obchodní domy, které mají po celé vnější ploše náhodně rozmístěné křiklavé reklamní bannery. Každý se snaží uchytit pozornost kolemjdoucího a navodit dojem, že právě ta jeho reklama je ta nejdůležitější. Jenže pokud má být důležité všechno, výsledkem je, že důležité se vlastně nezdá nic.

Při snaze o potlačení vizuálního šumu je tedy třeba si položit následující otázky:

- Co je v prostředí pro většinu uživatelů opravdu důležité a mělo by být zvýrazněno? Pokud prostředí přesytíme pestrým stylováním, animacemi, vyskakovacími okny apod., kromě orientace se také zhoršuje schopnost uživatele udržet u obsahu pozornost.
- Jak by mělo být prostředí organizováno? Je třeba se vyhnout pozicování prvků různě po ploše. Dobrým způsobem, jak rozložit obsah, je pomocí mřížkového rozložení.
- Co je na stránce či obrazovce aplikace skutečně potřeba a co na ni spíše nepatří? Co bylo vhodné schovat/přesunout jinam? Je třeba být s přidáváním obsahu obezřetný, místo přidání informační hodnoty ji totiž přeplněné prostředí ztrácí odváděním pozornosti od skutečně důležitého obsahu.

2.2 SpokenData ATCO2 anotační služba

ATCO2 anotační služba SpokenData[1] firmy ReplayWell (viditelná na obrázku 2.2) slouží k manuální opravě automatických přepisů hlasové komunikace řízení letového provozu a anotaci jejich klíčových částí. Takto opravené a anotované přepisy jsou následně využity pro zpětné trénování strojového učení automatického přepisu. ATCO2 je projekt zaměřený na tvorbu platformy pro sběr, organizaci a předzpracování hlasové komunikace řízení leteckého provozu.

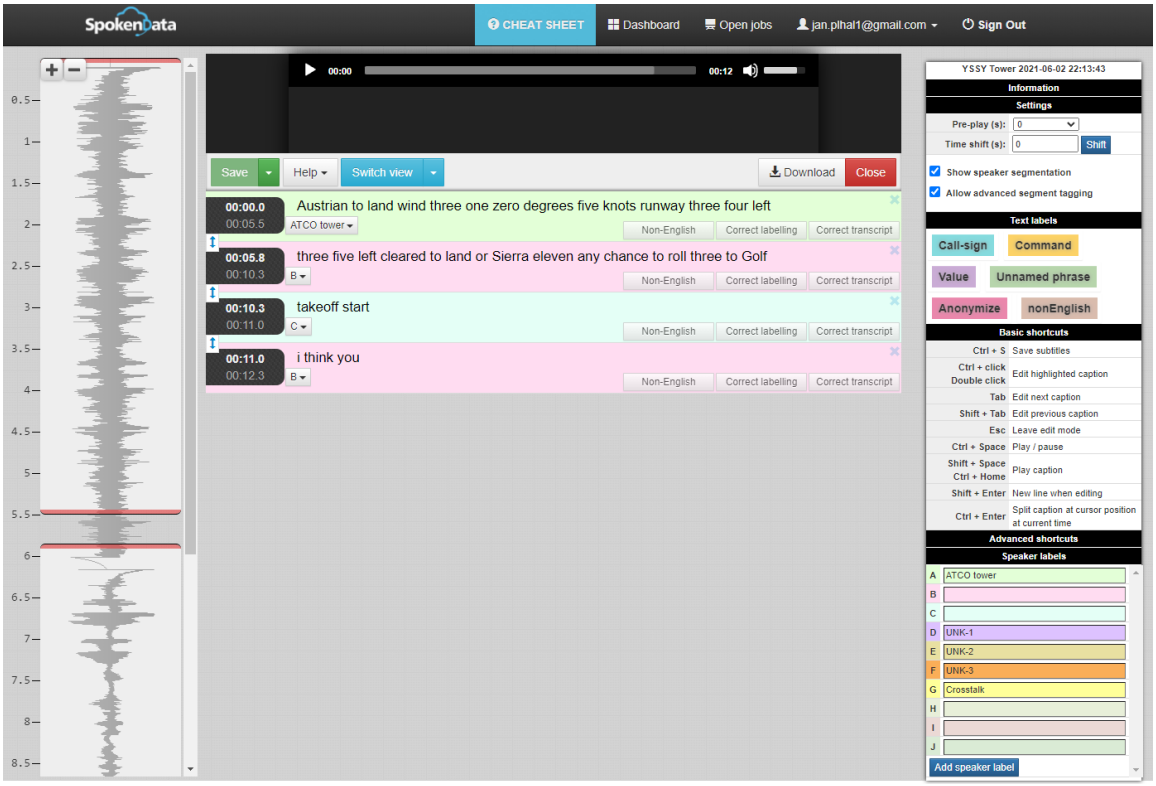
Jelikož jde o aplikaci, která bude v rámci práce přetvořena, zejména za účelem zdokonalení její použitelnosti, průzkum její funkcionality a případných slabin je pro návrh vyvíjené aplikaci zcela stěžejní. Tato sekce se bude pouze zabývat prvky prostředí, které mají jisté nedostatky a diskuzí o možnostech jejich zlepšení. Detailní popis funkcionality, která bude zachována i v nové aplikaci, bude obsažen na konci kapitoly v rámci návrhu funkcionality.

Editační mód segmentů

Pro úpravu textu segmentu je třeba jej nejprve uvést do editačního módu, načež se také zpřístupní možnost segment přehrát a přidat nový segment za ten editovaný. Segment je uveden do editačního módu dvojklikem nebo pomocí klávesové zkratky.

Tato funkce mi přijde jako zbytečný mezikrok, všechny možnosti by mohly být jednoduše dostupné vždy a editace segmentu by mohla být započata pouhým umístěním kurzoru do

[1] https://www.spokendata.com/atco2



Obrázek 2.2: Webová aplikace ATCO2 anotační služby SpokenData (oříznuto na obsah)

textu. Takto uživatel nejprve musí zjistit, že něco jako editační mód existuje a také jak se do něj dostat. Navíc použití dvojkliku pro velice často prováděnou operaci není příliš vhodné i přes zavedení alternativní klávesové zkratky. Jak totiž bylo zmíněno výše v této kapitole v podsekci 2.1.1, jakmile uživatel najde první způsob, další již v brzké době nehledá.

Jediným přínosem této funkce je cílení klávesových zkratek na segment v editačním módu, nicméně klávesové zkratky by šlo cílit i na segment, který má ve svém textu aktuálně umístěný kurzor.

Anotace sekcí textu

Anotace sekcí textu v rámci segmentů přepisu probíhá zakliknutím tlačítka značky a kliknutím na cílová slova. Taktéž lze kliknout pouze na první a poslední slovo z cílové sekce textu pro označení celé sekce zároveň. Ačkoliv je tento způsob označování efektivní, je velmi nekonvenční a uživatel tak musí nejprve na způsob fungování přijít. Pravděpodobně první způsob, který uživatel vyzkouší, je selekce cílové sekce textu a kliknutí na značku. Tak totiž fungují textové editory typu Microsoft Word, se kterými již pracoval skoro každý.

Vizualizace zvukové stopy

Na vizualizaci zvukové stopy jsou zobrazeny jednotlivé segmenty přepisu umístěné ve svých časových rozsazích. Segmenty lze na stopě pozicovat změnou jejich velikosti, avšak chybí možnost segment bez změny velikosti přetáhnout na jinou pozici.

Orientace vizualizace na výšku mi osobně přišla nepřirozená a hůře se mi tak v ní orientovalo, ačkoliv na efektivitu práce to efekt nemělo, jelikož po uvedení segmentu do

editačního módu je daný segment zascrollován do výhledu a graficky zvýrazněn. Orientaci nepomáhá ani fakt, že všechny segmenty (kromě segmentu v editačním módu) vypadají stejně. Kdyby byla vizualizace orientována na šířku a jednotlivé segmenty by měly barvu značky svého mluvčího, měla by větší okamžitou informační hodnotu.

Přizpůsobitelnost prostředí

Většina nástrojů a dalších prvků je umístěna ve volně pozicovatelném sloupci, který je logicky rozdělen na skrytelné sekce. Kvůli umístění všech částí do jednoho celku a absenci pozicovatelnosti ostatních prvků jsou však možnosti přizpůsobení prostředí velmi omezené.

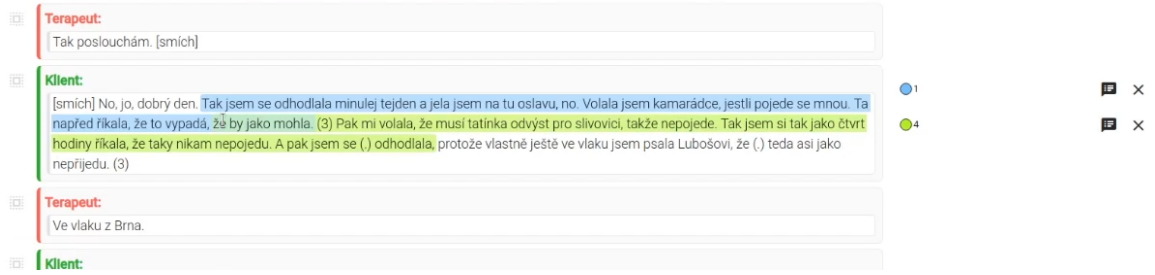
2.3 Průzkum trhu

Průzkum existujících řešení je důležitý pro pozdější návrh aplikace, jelikož poskytuje možnost získat inspiraci a představu o případných konvencích aplikací daného typu.

2.3.1 Deepsy anotační služba

Prostřednictvím konzultace mi byla předvedena rozpracovaná anotační služba Jana Kukuczky vyvíjené v rámci jeho diplomové práce (k vidění na obrázcích 2.3 a 2.4). Hlavním účelem této služby je oprava a anotace přepisů z psychiatrických sezení. Frontend aplikace je implementován v TypeScriptovém frameworku pro webové aplikace Angular.

Obrázek 2.3: Webová aplikace anotační služby Deepsy – detail záhlaví



Obrázek 2.4: Webová aplikace anotační služby Deepsy – detail segmentů přepisu

Vzhled rozhraní

Relativně menší rozsah funkcionality aplikace dovolil tvorbu minimalistického a přehledného rozhraní. Ačkoliv mnou vyvíjená aplikace bude obsahovat rozsáhlejší funkcionalitu a nevyhnutelně tedy větší počet prvků na obrazovce, toto provedení si lze jistě vzít příkladem.

Oprava přepisu

Jelikož prostředí neumožňuje zarovnávání částí přepisu na časové ose nahrávky a přepis je na časové rozsahy nahrávky zarovnán na úrovni slov, oprava přepisu probíhá po slo-

vech prostřednictvím dialogového okna. To se může stát obzvlášť neefektivním, zejména při opravě delších částí přepisu, navíc pokud strojové učení nesprávně přepis časově zarovná, nic s tím již v aplikaci nezmůžeme. Na druhou stranu zarovnání po slovech umožňuje zvýrazňování jednotlivých slov přepisu v době, kdy jsou řečena v průběhu přehrávání nahrávky, což uživateli usnadňuje orientaci v textu.

Označování textu

Označení sekce textu probíhá její selekcí a následným kliknutím na tlačítko značky, což je konvenční a pro uživatele rychle pochopitelný způsob. Značky lze taktéž libovolně kombinovat, jelikož to pro zadavatele práce a daný účel bylo potřebné. Množina značek je pak zobrazena stranou vedle označené části textu segmentu, kde je lze i odstranit. Jde o dobré a pravděpodobně jedno z mála možných umístění těchto prvků při podpoře kombinace značek textu. Alternativou by mohlo být zobrazení jmen jednotlivých značek při najetí myší na označený text a možnost značky odstranit by mohla být v kontextovém menu. Nicméně jelikož po stranách segmentů nejsou žádné další prvky, použité umístění je zcela ideální.

2.3.2 Prodigy

Prodigy[2] je placený anotační nástroj pro vyhodnocování dat a trénování modelů strojového učení. Jde o webovou aplikaci, kterou si uživatel stáhne a spouští lokálně. Chybí jakékoliv uživatelské účty či správa projektů – Prodigy je designován pro malé týmy pracující na rychle se vyvíjejících projektech. Mimo jiné nabízí Pythonovou knihovnu s řadou předpřipravených workflows a komponenty pro implementaci svých vlastních workflow skriptů. Skripty pak mohou specifikovat, jak jsou data načtena a uložena, změnit otázky pokládané v anotačním rozhraní a také definovat vlastní HTML a JavaScript pro úpravu chování frontendu. Ačkoliv je aplikace placená, je dostupné demo (viditelné na obrázku 2.5), což pro prozkoumání výchozího rozhraní postačí.

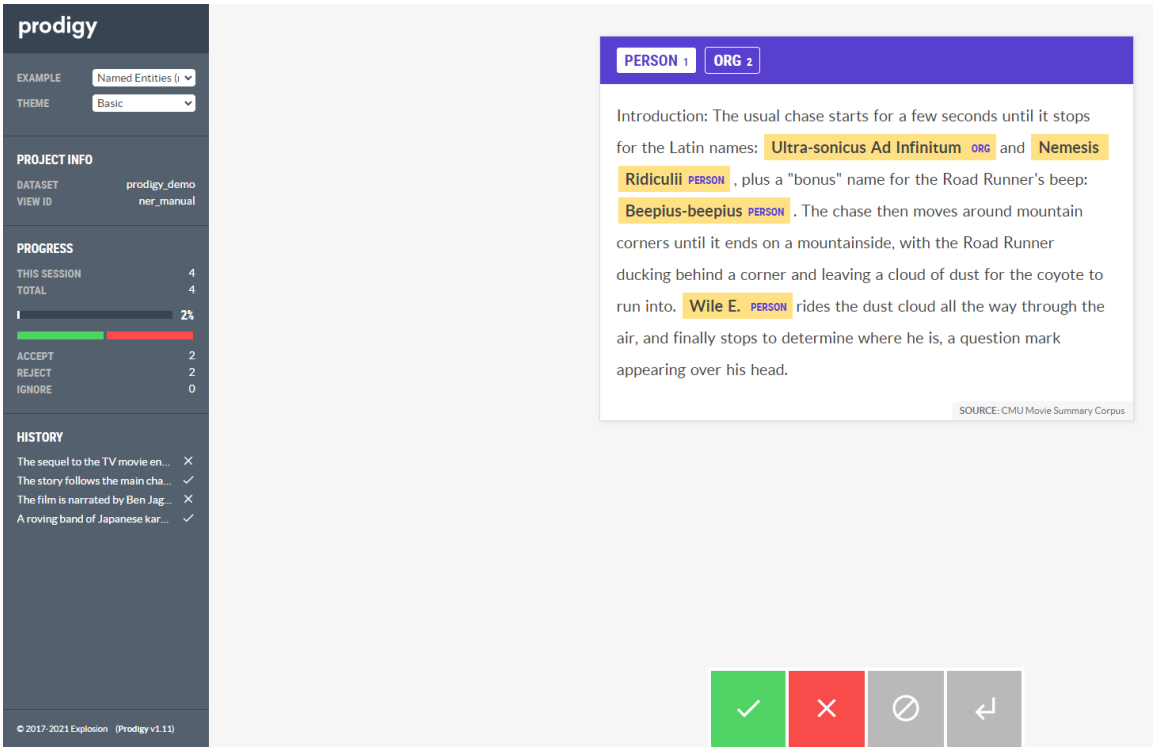
Rozložení rozhraní

Ve spodní části obrazovky se nachází ovládací prvky pro přepínání mezi anotovanými celky a schválení, či odmítnutí anotace. V levém panelu pak lze nalézt pouze informace o postupu anotace, projektu jako celku a nastavení témata vzhledu rozhraní. Informace o postupu anotace jsou zobrazeny i formou grafického indikátoru průběhu, což je prvek gamifikace, který zvyšuje zájem uživatele v práci pokračovat. Veškeré prvky týkající se anotace samotné jsou umístěny na středu obrazovky a uživatel tak vždy ví, kde anotační funkcionalitu hledat.

Anotace textu

Anotace textu, taktéž viditelná na obrázku 2.5, probíhá zakliknutím požadované značky a selekcí cílových slov, načež jsou ihned označena. Stačí i selekce části cílového slova, které je následně označeno celé. Pro anotaci, kde nedává smysl označit pouze část slova, je tato mechanika nutností, jelikož zamezuje nechtěné nevalidní anotaci a je rychlejší, než precizní selekce celých slov. Místo barevného odlišení značek je zde název značky vždy vyobrazen u každého označeného celku. Ačkoliv se tak uživatel nemusí učit barevné kódy jednotlivých značek, vizuální orientace je vždy rychlejší než čtení popisku. Možností je i kombinace popisků a barevných kódů, nicméně popisky také zasahují do proudu textu a ten je tak

[2] https://prodi.gy/



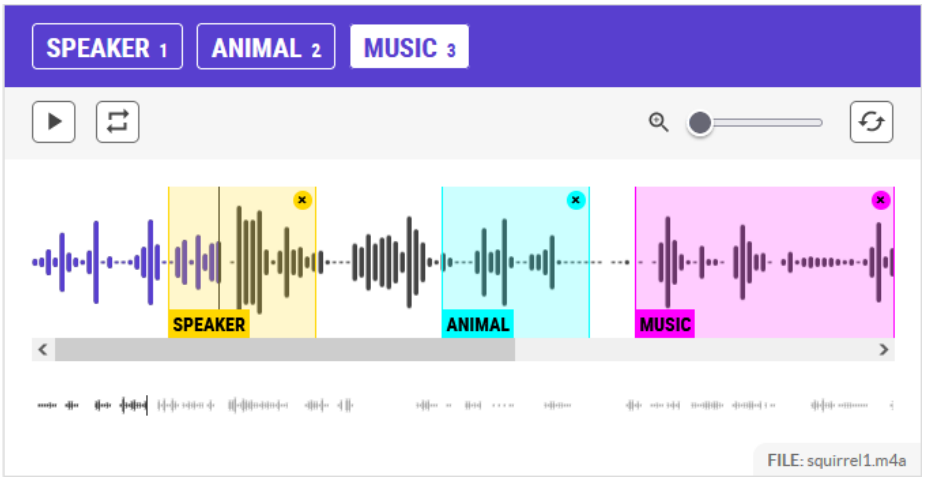
Obrázek 2.5: Webová aplikace anotační služby Prodigy (oříznuto na obsah)

o něco hůře čitelný. Kliknutím kdekoliv na značku je značka odstraněna, což je v tomto případě dobré řešení, jelikož samotný text nelze upravovat.

Anotace zvuku

Anotace zvuku probíhá opět zakliknutím tlačítka značky a přetáhnutím myší po vizualizaci zvukové stopy, načež je na daném místě vytvořena značka, což je jednoduchá a intuitivní mechanika. Jsou zobrazeny dvě zvukové křivky, ze kterých ta větší vrchní slouží pro anotaci a ta spodní funguje jako minimapa pro běžné zobrazení aktuální pozice ve zvuku a provedení jejího posunu. Značky u sebe mají opět popisek s jejich názvem a v tomto případě mají i svoji barvu, díky čemuž se mezi značkami lépe orientuje.

Dostupné jsou možnosti zvuk spustit/zastavit, přehrát od začátku, přiblížit si zvukovou křivku a také resetovat veškeré anotace. Resetování přitom není nijak graficky odlišené od ostatních tlačítek a nelze vrátit zpět. Bylo by vhodné přidat alespoň dialogové okno pro potvrzení resetování, při nepozornosti by totiž mohlo dojít ke ztrátě neuložené práce. Chybí možnost upravit hlasitost zvuku, která by se při nahrávkách s různými úrovněmi hlasitosti jistě mohla hodit.



Obrázek 2.6: Anotace zvuku v aplikaci Prodigy

Tooltipy

Opravdu dobře vymyšlené jsou tooltipy tlačítek, které kromě popisu akce, kterou provádí, obsahují i klávesovou zkratku, kterou lze použít jako alternativu k jejich zmáčknutí. Tím byla pro tyto zkratky eliminována potřeba je někde zvlášť uvádět a uživatel si je může intuitivně zjistit.

Vzhledová témata

V demu je dostupných celkem 5 vzhledových témat, což osobně považuji za přežitek, avšak některá skupina uživatelů to může ocenit a volba mezi světlým a tmavým tématem je určitě vždy výhodou. Nicméně zde mi v tmavém rozhraní přijde kontrast barvy textu s pozadím příliš výrazný. Pokud jde o aplikace, u kterých uživatel stráví řady hodin, a mělo by být zavedené pouze jedno neměnitelné téma, dává smysl použít tmavé rozhraní za účelem snížení zátěže na oči uživatele, zejména ve večerních hodinách.

Obrázek 2.7: Tmavé rozhraní aplikace Prodigy

2.4 Návrh funkcionality aplikace

Výsledná aplikace by měla obsahovat, pokud možno, veškerou funkcionalitu původní aplikace anotační služby SpokenData, kterou je místy třeba zdokonalit, a novou funkcionalitu zvyšující efektivitu a rychlost práce. Část z přidané funkcionality je inspirována zpětnou vazbou k původní aplikaci od uživatelů zprostředkovanou vedoucím práce.

2.4.1 Převzatá funkcionalita původní aplikace

Základní funkcionalita

Tyto funkce jsou jádrem původní aplikace a jejich zahrnutí je pro použitelnost pro daný účel zcela nutné. Jde zejména o tyto prvky:

- Načítání a zobrazení výstupu automatického přepisu spolu s dynamickými značkami a dalšími podpůrnými daty.
- Tvorba a úprava značek mluvčích segmentů.
- Tvorba a úprava anotačních segmentů s přepisem nahrávky umístěných v konkrétním časovém rozsahu zvukové stopy, k nimž je možné přiřadit mluvčího a další anotační příznaky.
- Přehrávání jak celé zvukové stopy, tak i jednotlivých segmentů zvlášť.
- Úprava přepisu uvnitř jednotlivých segmentů a značkování slov, případně delších částí textu. V přepisu je taktéž třeba vyznačovat určitá místa zvláštními značkami, které nejsou vázány k textu.
- Možnost pohybovat se napříč rozpracovanými přepisy a uzavřít aktuální přepis jako dokončený, nebo jej odmítnout. Tyto operace lze provést i mimo aplikaci na webu SpokenData, nicméně jejich zavedení přímo do aplikace je pro uživatele o něco pohodlnější.

Doplňující funkcionalita

Mezi doplňující funkcionalitu jsou zahrnuty prvky ne zcela nutné pro základní pracovní postup anotace. Tyto prvky však mohou práci s prostředím značně zefektivnit/urychlit:

- Zobrazování dynamických podpůrných materiálů přiložených k nahrávce (zejména text a obrázky poskytující kontext pro přepis či odkazy na externí zdroje). Tato funkce se dá téměř považovat za funkcionalitu základní, jelikož konkrétně nahrávky komunikace pilot-věž jsou často ve špatné kvalitě a bez dalšího kontextu je provedení správného přepisu místy nemožné nebo velmi obtížné.
- Nástroje/nastavení pro urychlení provádění vybraných úkonů (například oprava zarovnání všech segmentů na zvukové stopě zároveň).
- Klávesové zkratky pro často prováděné úkony (přehrání nahrávky, uložení přepisu...), případně zvláštní operace prováděné pouze klávesovými zkratkami (například rozdělení segmentu v aktuálním čase a pozici kurzoru v textu).
- Sloučení dvou po sobě jdoucích segmentů do jednoho (automatický přepis může promluvy mluvčího místy nevhodně rozdělit do dvou segmentů).

2.4.2 Návrh změny funkcionality

Návrh nové funkcionality a úpravy té stávající se řídí jak zjištěnými poznatky uživatelů a vedoucího práce, tak poznatky vlastními získanými důkladným seznámením se stávajícím prostředím a postupu práce s ním. Zdrojem inspirace byly také aplikace zkoumané v rámci průzkumu trhu a principy návrhu dobře použitelného prostředí zkoumané v rámci druhé kapitoly.

Hlavním cílem je práci s prostředím urychlit a zefektivnit zdokonalením použitelnosti včetně rozšíření upravitelnosti prostředí pro individuální potřeby uživatelů. Všechny prvky prostředí budou rozdělené do modulů, které lze vyměnit, přidat či odstranit. Takto modulární prostředí pak lze v budoucnu relativně jednoduše přetransformovat například pro anotaci videí či jiné účely.

Přidaná funkcionalita

- Jednotlivé prvky budou rozděleny do modulů volně pozicovatelných uživatelem na mřížkovém rozložení. Tyto moduly bude taktéž možné minimalizovat, nebo úplně zavřít a později otevřít z modulu nastavení. Díky tomu může uživatel mít na obrazovce pouze přesně to, co v danou chvíli potřebuje a tam, kde mu to nejvíce vyhovuje. Jak bylo popsáno výše v sekci zabývající se behaviorálními vzory, pokud si uživatel prvky napozicuje sám, je větší šance, že je díky prostorové paměti rychle znovu najde.

- Uživatel bude mít možnost posouvat se ve svých akcích zpět a dopředu. Tím tak získá jistotu, že pokud něco pokazí, vždy se může vrátit zpět a nemusí chybu pracně opravovat. Zároveň se novému uživateli otevírají dveře pro beznásledkové experimentování s prostředím a jeho funkcemi, čímž může nabýt lepšího pochopení aplikace, které je pro efektivní práci s ní zásadní.

- Rychlost přehrávání nahrávky bude snížitelná, jelikož komunikace řízení letového provozu často probíhá velmi rychle a uživatelé mívají problém s ní držet krok.

- Postup práce bude periodicky automaticky ukládán. Jde o další jistotu pro uživatele z hlediska zachování postupu v situacích, kdy například ztratí internetové připojení či omylem zavře okno prohlížeče. Ztráta postupu může být velmi odrazující, zejména v tomto případě, kdy uživatel anotaci provádí zcela zdarma ve svém volném čase. Automatické ukládání bude možné uživatelem vypnout, či přenastavit jeho interval.

- Klávesové zkratky budou jednoduše přenastavitelné uživatelem, tudíž si je může případně upravit dle svých zvyklostí z jiných aplikací. Zároveň bude klávesa `Ctrl` na operačním systému Windows automaticky zaměnitelná s klávesou `Cmd` na systému MacOS. V původní aplikaci mělo množství zkratek pevně nastavenou klávesu `Ctrl`, což zapříčinilo jejich nepoužitelnost pro uživatele na MacOS. Tlačítka pro provedení operací, které jsou alternativně proveditelné i klávesovou zkratkou, budou mít klávesovou zkratku uvedenou v tooltipu daného tlačítka (inspirováno aplikací Prodigy zkoumanou v rámci průzkumu trhu).

- Obrázky bude možné přibližovat a posouvat se v nich. V původní aplikaci byly obrázky často špatně čitelné.

Úpravy původní funkcionality

- Značky přidávané mezi text budou součástí uživatelského prostředí, podobně jako tomu je u značek sekcí textu. Tyto značky byly původně pouze manuálně vepisované do textu na základě jejich znalosti z přiloženého anotačního manuálu. Díky jejich začlenění do prostředí si je uživatel nebude muset pamatovat z manuálu, ani si mezi ním a aplikací pro jejich nalezení přepínat. Jejich vložení bude probíhat jednoduše označením cílového místa kurzorem a kliknutím na danou značku.
- Označení sekce textu značkou bude probíhat více konvenčním způsobem a sice selekcí dané sekce a kliknutím na tlačítko některé značky. Jelikož na tomto principu fungují textové editory, které někdy používal nejspíše každý, bude tento způsob pro nové uživatele familiární a rychle pochopitelný.
- Tvorba nových segmentů bude probíhat intuitivně přetažením myší po vizualizaci zvukové stopy.
- Přílohy jako živé mapy a obrázky budou součástí jednoho modulu se záložkou pro každou přílohu. Uživatel tak může mezi přílohami rychleji přepínat.
- Každá textová příloha bude fungovat jako svůj vlastní modul. Uživatel tedy může mít otevřených více textových příloh současně, či mít otevřenou textovou přílohu a obrázek/mapu.

Kapitola 3

Průzkum a výběr implementačních technologií

Začátek této kapitoly se zabývá srovnáním nejpopulárnějších moderních frameworků pro JavaScript a kaskádové styly, na základě kterého je vybrán ten nejvhodnější pro účely této práce. Ke konci kapitoly jsou pak popsány ty nejdůležitější vybrané knihovny třetích stran.

3.1 JavaScriptové frameworky

JavaScriptové frameworky poskytují kromě řady nástrojů a funkcionalit, které tak nemusíme psát od nuly, také jistou standardizaci kódu, která je pro jeho srozumitelnost ostatním vývojářům stěžejní (zejména při práci v týmu). Správná volba frameworku se odvíjí od typu aplikace, můžeme sáhnout například po frameworku s pevně danou strukturou projektu a řadou vestavěných nástrojů, nebo po frameworku lehčím a flexibilnějším.

Tato sekce se bude věnovat srovnání třech v tuto dobu nejpopulárnějších JavaScriptových frameworků, na základě kterého bude vybrán ten nejvhodnější pro účely této práce. Zdrojem myšlenek je, pokud není jinak uvedeno, článek [2] od Andrewa Hurskiye a Sofiye Merenych.

React

React je JavaScriptová knihovna pro tvorbu uživatelských prostředí, která je vyvíjena společností Meta (dříve Facebook). Ta ji používá pro své produkty, tudíž Facebook, Instagram a WhatsApp. Původní vydání proběhlo v roce 2013 a aktuálně je ve verzi 18[1].

Mezi výhody Reactu patří:

- Používá virtual DOM (Document Object Model), což urychluje renderování. Virtual DOM funguje na principu vytvoření zjednodušené verze DOMu, která je později porovnána se skutečným DOMem a při nalezení rozdílů jsou přerenderovány pouze změněné části.
- Není vyžadováno použití tříd a aplikace tedy lze tvořit pomocí funkcí, což zjednodušuje zdrojový kód projektu.

[1] https://en.wikipedia.org/wiki/React_(JavaScript_library))

- Velká komunita, díky které lze snadno nalézt pomoc s vývojem a případné doplňující nástroje a knihovny.
- Možnost použití TypeScriptu.

Zápory Reactu zahrnují:

- Jelikož je React technicky knihovna a ne framework, jeho možnosti jsou v základu značně omezené. Pro rozšíření funkcionality je třeba se spoléhat na knihovny a moduly třetích stran. Díky aktivní komunitě je však z čeho si vybírat.
- Absence předdefinované struktury aplikace. Ta je tak zcela závislá na vývojáři.
- HTML kód je vepisován přímo do JavaScriptu za použití JSX (JavaScript Expressions). To může snižovat čitelnost kódu, nicméně záleží i na zvyku.

Angular

Angular je JavaScriptový framework založený na TypeScriptu vyvíjený společností Google. První vydání proběhlo v roce 2010 pod názvem „AngularJS“. V roce 2016 byla vydána od základu přepsaná druhá verze, tentokrát již pod zjednodušeným názvem „Angular“. Nové verze Angularu jsou vydávány pravidelně v intervalu 6 měsíců, přičemž nejnovější je v době psaní práce verze 13[2].

Mezi výhody Angularu patří:

- Používá architekturu MVW (Model-View-Whatever – značící jistou flexibilitu v tomto ohledu), která je tradičně používána jako MVC (Model-View-Controller). Aplikace je tak rozdělena do tří propojených vrstev, což umožňuje psát dobře strukturovaný kód, zejména pro komplexní projekty.
- Šablony pro tvorbu komponent jsou z většiny tvořeny standardními HTML tagy a jsou tak dobře čitelné. JavaScriptový kód je od HTML oddělený.
- Velká škála zabudovaných nástrojů pro validaci formulářů, správu stavů, směrování apod.
- Jednoduchá implementace dvojsměrné datové vazby (jakékoliv změny ve view se ihned projeví v modelu a naopak) – vhodné zejména pro jednoduché aplikace kvůli náročnosti na zdroje.
- Velká komunita, stejně jako u Reactu.

Zápory Angularu zahrnují:

- Objemný kód, který musí být stažen ze serveru před zobrazením aplikace v prohlížeči. To zapříčiňuje snížení rychlosti a výkonu. Lze však použít techniku tree-shaking, jež z aplikace odstraní nepoužitý kód.
- Místo virtual DOM používá incremental DOM, jenž je pomalejší, ale méně náročný na paměť.

[2] https://en.wikipedia.org/wiki/Angular_(web_framework)

- Použití TypeScriptu je v podstatě požadováno vzhledem k jeho použití ve veškeré dokumentaci.

Vue

Vue je nejnovějším z porovnávaných frameworků. Byl vydán bývalým zaměstnancem Googlu Evanem You v roce 2014. Ačkoliv není zastřešen velkou firmou, v posledních letech zažívá velký nárůst v popularitě, přičemž nejpopulárnější je stále v Číně, odkud autor pochází. Projekt je finančně podporován na síti Patreon. Aktuálně je ve verzi 3, která přinesla mimo jiné přechod na TypeScript. Úvodní informace o frameworku Vue byly převzaty z tohoto článku[3].

Mezi výhody Vue patří:

- Používá architekturu MVC stejně jako Angular.
- Jde o framework s velmi nízkou velikostí souborů.
- Dobře čitelné HTML šablony komponent, oddělené HTML a JavaScript. Nicméně lze použít i JSX.
- Používá virtual DOM, stejně jako React.
- Možnost použití TypeScriptu od verze 3.0.
- Jak zmiňuje oficiální dokumentace Vue[4], závislosti komponent jsou automaticky kontrolovány při renderu a systém tak ví, které komponenty je třeba znovu renderovat při změně stavu. V Reactu je defaultně znovu renderován celý podstrom komponent a je třeba tuto optimalizaci provést manuálně.

Zápory Vue zahrnují:

- Malá komunita, což znamená těžší hledání pomoci při vývoji.
- Menší počet dostupných knihoven pro rozšíření funkcionality.
- Vue je nejpopulárnější v Číně, a tak dokumentace pro mnoho z knihoven třetích stran může být dostupná pouze v čínštině.

Konečný výběr JavaScriptového frameworku

Ačkoliv Angular obsahuje mnoho zabudovaných nástrojů, pro účely této práce by většina z nich zůstala nevyužita a hledání knihoven třetích stran by i tak bylo třeba. Je sice možné nevyužitý kód odstranit pomocí tree-shakingu, ale to by mařilo účel použití takového frameworku. Angular je vhodný spíše pro komplexní aplikace většího měřítka.

Popularita frameworku Vue v posledních letech sice roste, ale komunitní podpora stále není zdaleka na úrovni Reactu. Oficiální knihovny pro věci, jako je správa stavů, jsou sice plusem, ale React má pro tyto účely stejně spolehlivé knihovny třetích stran. Vue má i další drobné výhody, jako je volba mezi použitím JSX a „obyčejným“ HTML, ale komunitní podporu a s tím spojenou kvalitu a kvantitu knihoven třetích stran nevyrovnají.

[3] https://www.codeinwp.com/blog/angular-vs-vue-vs-react/
[4] https://v2.vuejs.org/v2/guide/comparison.html?redirect=true

V konečném důsledku by pro účely této práce byl vhodný jak React, tak i Vue, nicméně React je mezi komunitou více zaběhlý, a navíc s ním již mám nějaké zkušenosti, nemělo by tedy smysl se učit s novým frameworkem od nuly bez zásadního důvodu.

3.2 Frameworky pro kaskádové styly

Frameworky pro kaskádové styly poskytují větší flexibilitu a podstatně urychlují vývoj předpřipravenými komponentami, nástroji pro rozložení stránky a dalšími prvky, které jsou také konzistentní napříč prohlížeči. Zdrojem porovnání zástupců frameworků v této sekci je článek [5] od Alexandera Rubanaua.

Bootstrap

Bootstrap je framework pro kaskádové styly vhodný pro tvorbu responzivních webových aplikací použitelných i na mobilních zařízeních. Z řad frameworků pro kaskádové styly je aktuálně tím nejpopulárnějším.

Bootstrap byl původně pod názvem Twitter Blueprint vyvíjen jako interní framework Twitter týmu s cílem ulehčit vývojářům práci a zavést standardizaci v rámci interních nástrojů. Jako open-sourcový projekt byl vydán v roce 2011. Dnes Bootstrap využívají aplikace jako je Airbnb, Dropbox, Apple Music a samozřejmě i Twitter.

Mezi výhody Bootstrapu patří:

- Flexibilní a plně responzivní mřížkové rozložení.
- Mnoho zabudovaných komponent, jako jsou tlačítka, karty, upozornění a další.
- Vysoký počet volně dostupných témat a šablon.
- Konzistentní (avšak upravitelný) vzhled, který je pro uživatele jiných aplikací používajících Bootstrap familiární.
- Umožňuje tvorbu přehledného a dobře čitelného prostředí bez prvků odvádějících pozornost.

Zápory Bootstrapu zahrnují:

- Velmi podobný vzhled aplikací používajících Bootstrap kvůli zaměření na standardizaci.
- Objemné soubory vedoucí k delším načítacím časům a rychlejšímu vybíjení baterie na mobilních zařízeních.

Material UI

Material UI je framework pro React následující designové principy Material Designu. Material design je sada komponent, nástrojů a pokynů pro tvorbu (zejména mobilních) uživatelských prostředí vyvinutá Googlem v roce 2014, kdežto Material UI, jeho adaptace pro React, byl vyvinut bezejmenným týmem v roce 2017. Material Design je používán Googlem ve všech jeho produktech a Material UI používají společnosti, jako jsou Amazon a Nasa.

Mezi výhody Material UI patří:

- Velmi vhodný pro mobilní aplikace, poskytuje řešení pro nedostatek místa, jako jsou animace, vrstvy a vyskakovací okna. Komponenty poskytují okamžitou zpětnou vazbu na dotek pomocí animací.
- Flexibilní a plně responzivní mřížkové rozložení podobné tomu v Bootstrapu.
- Nabízí řadu komponent založených na Material Designu.
- Poskytuje prostor pro moderní a originální designy.

Zápory Material UI zahrnují:

- Při častém používání aplikace mohou být animace příliš ohromující a odvádět pozornost. Na desktopu animace uzpůsobené pro zpětnou vazbu dotyku ztrácejí význam.
- Unikátně designovaná prostředí mohou být hůře uchopitelná pro nové uživatele.
- Efekty jako animace, stíny a přechody jsou náročné na výkonovou režii.

Konečný výběr frameworku pro kaskádové styly

Pro účely této práce byl vybrán framework Bootstrap 5, jelikož poskytuje nástroje pro rychlou a efektivní tvorbu přehledných uživatelských prostředí. Originální design není v tomto případě důležitým aspektem a vizuální prvky Material Designu by mohly při práci s prostředím působit rušivě. Zároveň díky velké rozšířenosti a standardizaci vzhledu Bootstrapu bude prostředí pro uživatele působit familiárně. Pozornost však bude muset být věnována výkonu aplikace, přičemž v případě potřeby je z frameworku možné pročistit nepotřebné prvky.

3.3 Výběr knihoven

Správná volba knihoven třetích stran je důležitým aspektem návrhu aplikace. Vývojář typicky nemá čas ani důvod znovu vynalézat kolo, jestliže nemá velmi specifické požadavky. Je však třeba aplikovat značnou míru rozvážnosti, protože méně populární a udržované knihovny mohou brzy zcela ztratit podporu a použití zbytečně velkého počtu či zbytečně rozsáhlých knihoven zatěžuje aplikaci a přidává práci s jejich aktualizací a případnou opravou breaking changes (zásadních změn rozhraní knihovny rozbíjející aplikace postavené na její starší verzi).

Wavesurfer.js

Wavesurfer.js[5] je vysoce flexibilní knihovna pro tvorbu zvukových přehrávačů s vizualizací zvukové stopy postavená na Web Audio API a HTML5 Canvas. Jde o osvědčenou (v době psaní slaví 10 let jejího vývoje) a dobře udržovanou knihovnu. Má detailně dokumentované rozhraní a několik pluginů (zásuvných modulů), ze kterých je pro účely této práce nejpodstatnější plugin pro tvorbu regionů na zvukové stopě, viditelný na obrázku 3.1, jež bude využit pro anotační segmenty mluvčích.

[5] https://wavesurfer-js.org/

Existuje i nezávisle vyvíjený wrapper (knihovna tvořící vrstvu nad jinou knihovnou za účelem změny jejího rozhraní) pro React, nicméně knihovna je v Reactu dobře použitelná i ve své původní čistě JavaScriptové verzi, tudíž je lepší se vyhnout přidání dalšího článku, kterému v budoucnu může skončit podpora.

Obrázek 3.1: Příklad vizualizace zvukové stopy knihovny Wavesurfer.js s regiony.

Slate

Slate[6] je knihovna/framework pro tvorbu rich text editorů (editorů textu obohaceného o formátování) inspirovaná knihovnami, jako jsou Draft.js, Prosemirror nebo Quill. Na rozdíl od těchto knihoven je Slate zaměřený na vysokou přizpůsobitelnost a umožňuje tak implementaci velmi specifických případů použití. Tato knihovna bude použita pro implementaci značkování a úpravy textu přepisu. Taktéž obsahuje oficiální plugin pro pohyb mezi jednotlivými kroky uživatele, který bude kombinován s vlastní implementaci pro ostatní moduly.

Knihovna je stále ve verzi beta a vývojář nevylučuje možné zavedení breaking changes. V rámci průzkumu jsem si prakticky vyzkoušel i konkurenční Draft.js, avšak po nějaké době se stalo zřejmým, že jeho použití pro účely této práce by bylo značně obtížné. Naproti tomu se Slate díky jeho přizpůsobitelnosti zdá být zcela ideální a jeho rozhraní mi osobně přišlo znatelně uchopitelnější. Stojí tedy za to přijmout možnou potřebu úpravy implementace při případné budoucí aktualizaci knihovny.

React grid layout

Jde o knihovnu[7] umožňující tvorbu uživatelsky upravitelných a responzivních „nástěnek“ na mřížkovém rozložení. Prvky umístěné na této nástěnce může uživatel jednoduše a intuitivně přetahovat na libovolné pozice, měnit jejich velikost a při implementaci nějaké formy výběru prvků i prvky za běhu odebírat či přidávat. Uživatelská změna velikosti prvků však nebude využita, jelikož to pro moduly v aplikaci příliš nedává smysl, možná až na moduly příloh, kde by si tak uživatel mohl měnit velikost textových polí a obrázků. Nicméně textová pole již mají uživatelsky měnitelnou výšku a obrázky budou volně přibližovatelné.

[6] https://docs.slatejs.org/
[7] https://github.com/react-grid-layout/react-grid-layout

Souhrn funkcionality pokryté knihovnami

- Úprava textu segmentů, označování sekcí textu, vkládání značek mezi text – **Slate**
- Přibližování obrázkových příloh – **react-zoom-pan-pinch**[8]
- Přehrávač s vizualizací zvukové stopy, manipulace se segmenty na zvukové stopě – **Wavesurfer.js**
- Převod událostí zmáčknutí kláves na řetězcový formát (pro uživatelsky přenastavitelné zkratky) – **key-event-to-string**[9]
- Uživatelsky upravitelná nástěnka – **React-Grid-Layout**

[8] https://www.npmjs.com/package/react-zoom-pan-pinch
[9] https://www.npmjs.com/package/key-event-to-string

Kapitola 4

Návrh grafického rozhraní

Pro ujasnění cílové podoby grafického rozhraní byl vytvořen jeho wireframe (drátěný model), který zachycuje hlavní rozložení a způsob fungování aplikace, bez detailního designu a barev, které by mohly v této fázi zbytečně odvádět pozornost od důležitějších částí návrhu.

Podoba grafického rozhraní se bude v některých aspektech přibližovat původní aplikaci, jelikož musí být zachována její funkcionalita a pokud některé části už roky fungují a fungují dobře, není třeba je nutně inovovat. Výhodou je, že rozhraní tak bude rychleji uchopitelné pro uživatele původní verze.

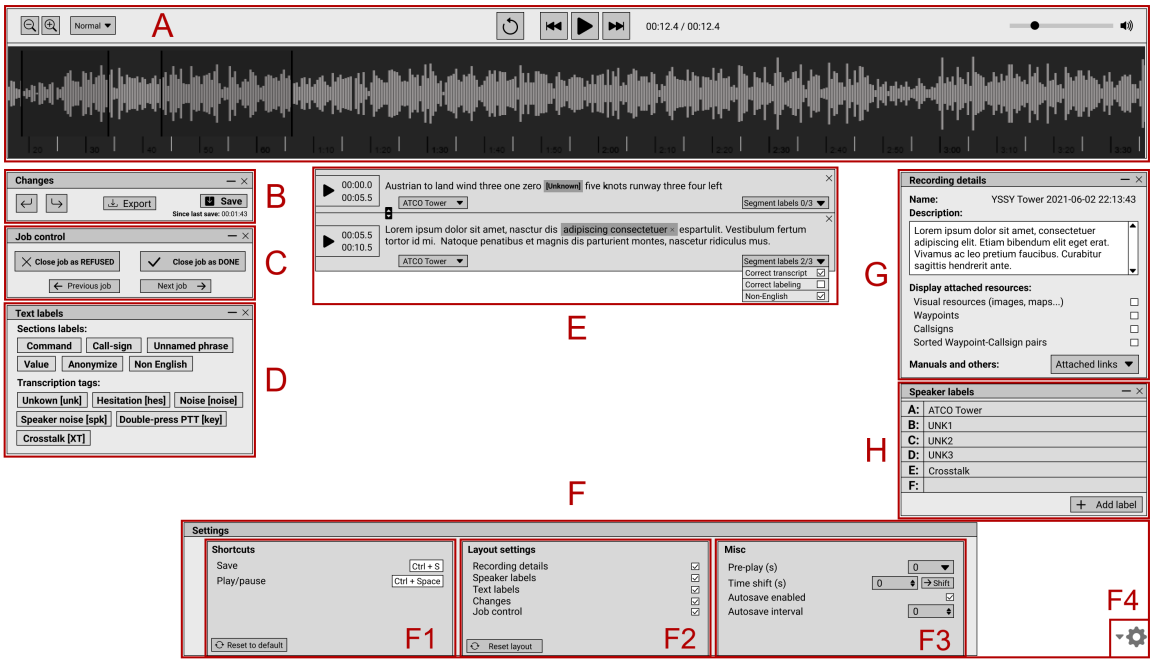
4.1 Wireframe aplikace

Pro tvorbu wireframu rozhraní aplikace byl použit volně dostupný webový nástroj Figma[1]. V jedné z jeho prvotních podob lze vidět na obrázku 4.1. Wireframe zachycuje výchozí rozložení modulů na nástěnce, které však bude uživatelem volně upravitelné.

Návrh byl průběžně iterován a ve své finální podobě je zobrazen na obrázku 4.2. Největší změnou z pohledu rozložení je kombinace všech modulů nastavení do jednoho vysouvacího okna za účelem redukce vizuálního šumu a uvolnění místa pro přílohové moduly. Detailní popis jednotlivých modulů a případných změn, kterými prošly bude uveden níže.

[1] https://www.figma.com

Obrázek 4.1: Prvotní wireframe rozhraní aplikace



Obrázek 4.2: Finální wireframe rozhraní aplikace: **A** – přehrávač, **B** – modul Změny, **C** – modul správy přepisů, **D** – textové značky, **E** – segmenty mluvčích, **F** – modul nastavení, **F1** – nastavení klávesových zkratek, **F2** – nastavení rozložení, **F3** – ostatní nastavení, **F4** – tlačítko pro otevření modulu nastavení, **G** – detaily přepisu, **H** – značky mluvčích

4.2 Detailní popis modulů

Přehrávač

Co se ovládacích prvků týče, v přehrávači přibyla tlačítka na přetáčení nahrávky a tlačítko na přehrání nahrávky od začátku je nyní vždy dostupné. V původní aplikaci se toto tlačítko objevilo pouze po přehrání až do konce, což obvykle stačí, nicméně při anotaci je typicky třeba si nahrávku pouštět několikrát po sobě, a ne vždy si ji uživatel potřebuje přehrát celou.

Segmenty bude možné vytvářet přetažením myší po prázdné části zvukové stopy, pozicovat přetáhnutím a měnit jejich velikost tažením za jeden z krajů. Segmenty budou taktéž mít barvu značky mluvčího pro jejich rychlejší identifikaci.

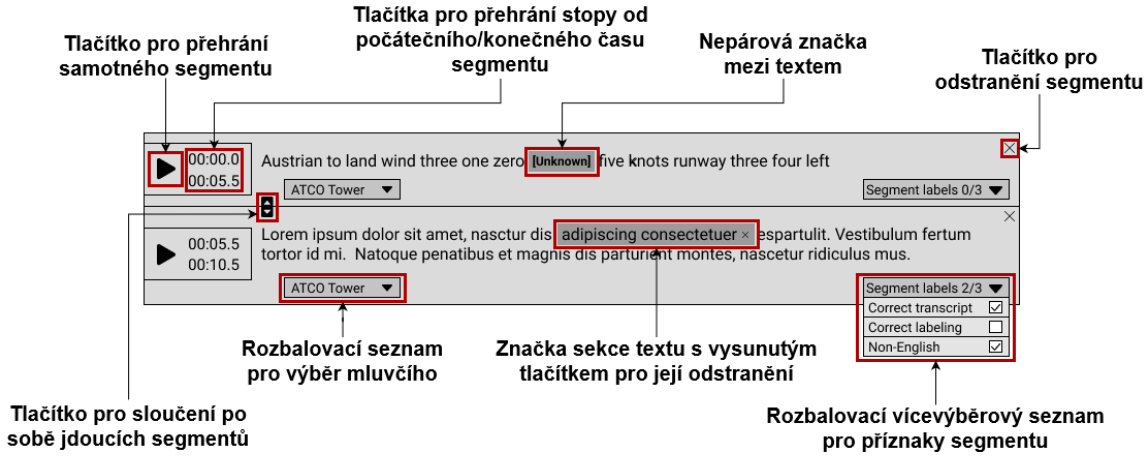
Vizualizace zvukové stopy byla přeorientována na šířku pro snazší orientaci a manipulaci se segmenty.

Obrázek 4.3: Popsaný návrh přehrávače

Segmenty mluvčích

Způsob práce se segmenty přepisu byl oproti původní aplikaci o něco zjednodušen – místo dvojkliku na segment pro jeho uvedení do editačního módu a zobrazení všech možností jsou všechny možnosti dostupné již od začátku a úprava přepisu je započata jednoduše kliknutím dovnitř textu. Použití dvojkliku pro velmi často prováděné operace mi nepřijde jako dobrá volba i přes to, že pro pohyb mezi editačními módy jednotlivých segmentů byla k dispozici klávesová zkratka. Uživatel totiž použije první zjištěný způsob, jak akci provést a další ve většině případů již nehledá, což jsem pozoroval dokonce i na sobě. Každý způsob/alternativa pro provedení akce by tedy měla být co nejefektivnější. Navíc proč by měl uživatel segment uvádět do editačního módu pro získání možnosti jeho přehrání? Může si například při editaci jednoho segmentu potřebovat pustit ten následující pro získání kontextu.

Příznaky segmentů byly umístěny do rozbalovacího vícevýběrového seznamu, protože jsou do prostředí vkládány dynamicky a může jich být i větší počet, než se do okna segmentu fyzicky vleze. Kvůli tomu však uživatel ihned nevidí, kterými příznaky jsou segmenty označeny a pro jejich úpravu musí provést o jeden klik navíc. Do popisku rozbalovacího seznamu byl tedy alespoň přidán počet vybraných příznaků. To ale situaci zcela neřeší, dobrým kompromisem by bylo zobrazení příznaků napřímo do určitého počtu a při překročení hranice je sbalit do seznamu.

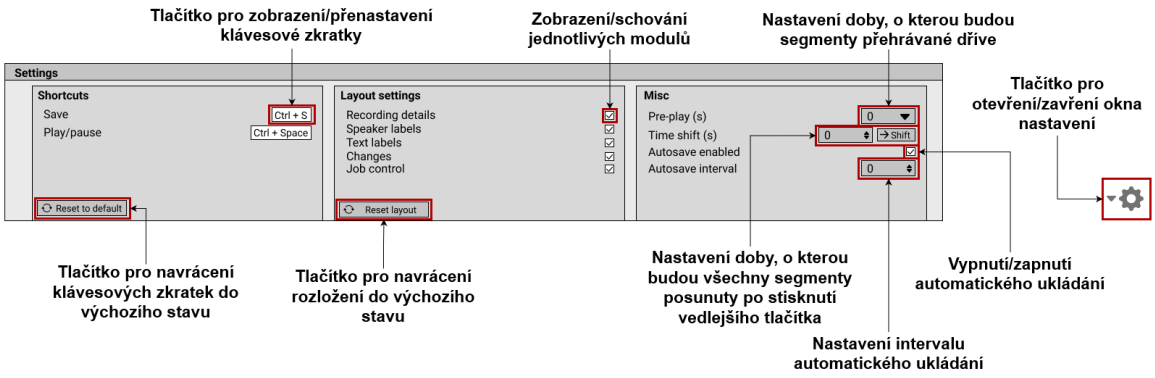


Obrázek 4.4: Popsaný návrh segmentů mluvčích

Co se změn oproti prvotnímu návrhu týče, od značek textu byl odstraněn popisek s názvem značky, protože narušoval proud textu a ten byl tak špatně čitelný. Jeho zobrazení při přejetí myší zase ztěžovalo přístup k tlačítku pro odstranění značky, byla tedy ponechána pouze identifikace dle barvy. Odstraněno bylo také překrývání značek, hlavně z časových důvodů, nicméně podoba, v jaké bylo navrhnuto v prvotním návrhu, by ani nedovolovala překryv ze všech stran. Tlačítko pro odstranění značky by tedy muselo být přesunuto buďto do kontextového menu, nebo úplně mimo vedle segmentu, jak to měl implementované Jan Kukuczka v jeho aplikaci.

Okno nastavení

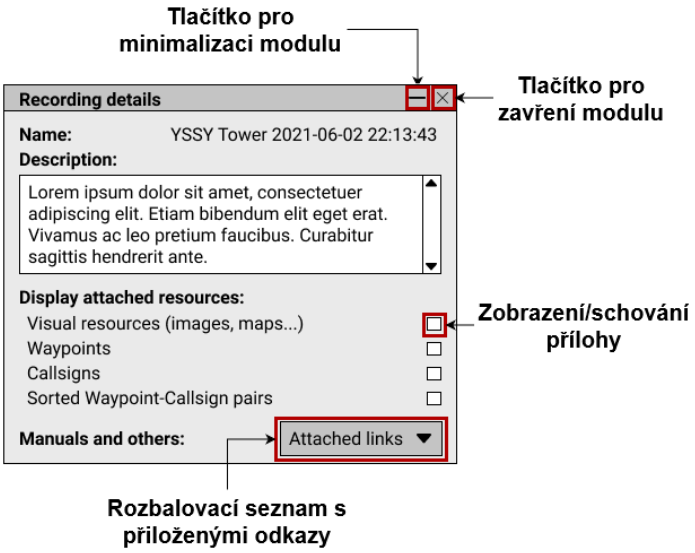
Nastavení je jediným modulem, který není volně pozicovatelný na hlavní ploše. A to z toho důvodu, že uživatel typicky provede potřebná nastavení a na nějakou dobu s ním již interagovat nepotřebuje. Nemá tedy smysl, aby zabíralo limitované místo na ploše a tvořilo vizuální šum odvádějící pozornost od důležitějších, častěji používaných prvků, nebo aby si jej uživatel na plochu přidával a hned znovu odebíral. Poněkud nekonvenční umístění v dolní části obrazovky je provedeno proto, že moduly na ploše automaticky plují do nejvyšší možné pozice. Pokud tedy někde bude prostor, kde okno nastavení nebude nic překrývat, bude to v dolní části obrazovky.



Obrázek 4.5: Popsaný návrh okna nastavení

Detaily přepisu

Tento modul kromě názvu a popisu přepisu také obsahuje přílohy k němu přidané. Původní návrh počítal se zvláštním modulem pro každou „vizuální“ přílohu (obrázky, mapy…), ale vyšlo najevo, že obrázky příliš uživatelů nepoužívá a mít zobrazeno více obrázku zároveň není třeba již vůbec. Proto budou přiložené obrázky sloučeny do jednoho modulu se záložkou pro každý z nich. V původní aplikaci se obrázky zobrazovaly výběrem z rozbalovacího seznamu, překlikávání mezi záložkami je tedy alespoň o něco rychlejší.

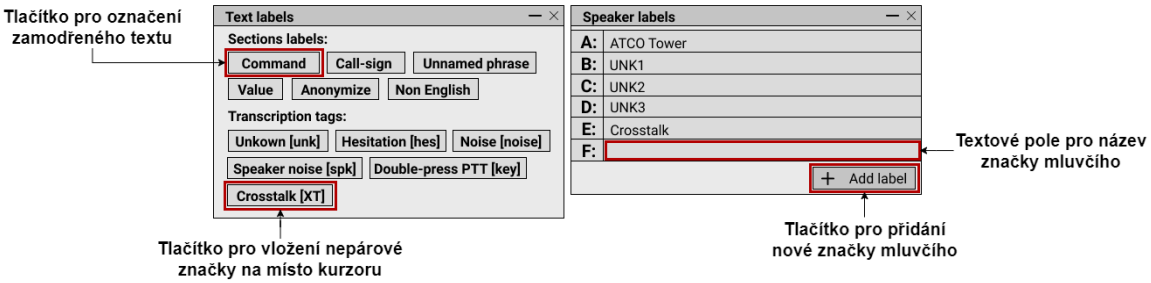


Obrázek 4.6: Popsaný návrh modulu detailů přepisu

Nejvíce používané jsou textové přílohy, které zároveň mohou být o velikosti modulů boční lišty a zachovat si čitelnost, proto ty jsou tedy separátní moduly ponechány.

Textové značky a značky mluvčích

Vzhled a fungování těchto modulů zůstalo z velké části nezměněno. Jak již bylo zmíněno v návrhu funkcionality, do textových značek přibyla sekce se značkami vkládanými mezi text a značky sekcí textu budou vkládány selekcí textu a kliknutím na příslušné tlačítko.



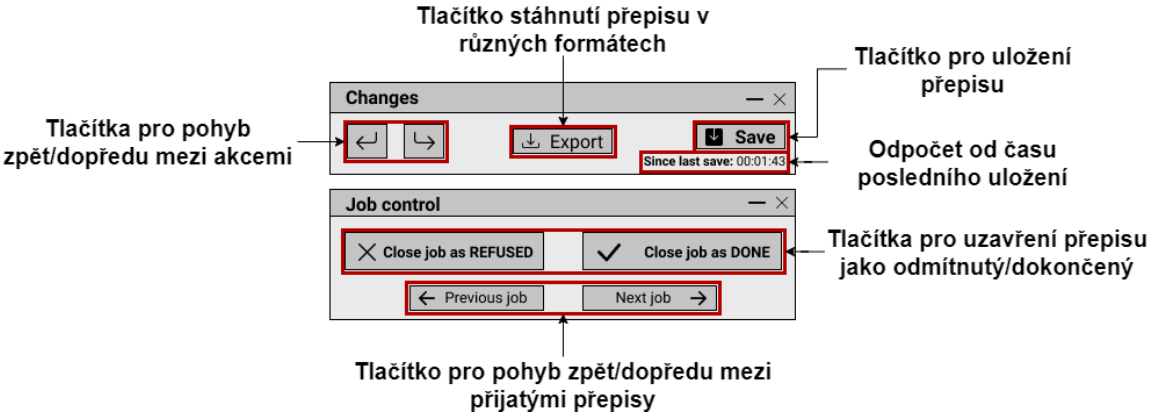
Obrázek 4.7: Popsaný návrh modulů textových značek a značek mluvčích

Je však možné, že by stávající uživatelé ocenili zachování i původního způsobu fungování, tedy zakliknutí tlačítka a následné vyklikávání slov. Tento způsob má jednu výhodu a sice, že dovoluje označovat více sekcí stejnou značkou bez opětovného zmáčknutí tlačítka. Tyto

způsoby fungování se nijak nevylučují a mohou fungovat pospolu. V případě, že by uživatel neměl žádnou aktivní selekci, tlačítko by zůstalo zakliknuté a umožnilo by se označování kliknutím na slova. Zdali tato alternativní funkce bude implementována bude rozhodnuto i na základě zpětné vazby uživatelů z testování.

Změny a správa přepisů

V modulu Změny se nachází tlačítka pro nově přidaný pohyb mezi kroky jakýchkoliv úprav přepisu, přičemž deaktivováním příslušného tlačítka bude indikováno dojití na konec akcí v daném směru. Dále je zde tlačítko pro export přepisu ve formátu JSON a tlačítko pro uložení změn, které bude deaktivováno při absenci změn od posledního uložení. Po uložení se zobrazí odpočet od posledního uložení, který bude po určitých časových úsecích měnit barvu mezi zelenou, oranžovou a červenou pro lepší indikaci, zda by si uživatel měl svoji práci uložit.



Obrázek 4.8: Popsaný návrh modulů změny a správa přepisů

V modulu správy přepisů byla oproti původní aplikaci přidána tlačítka pro uzavření přepisu jako odmítnutý či dokončený, která byla dříve uschována ve vyskakovacím okně otevřeném po pokusu o navigaci mezi uživatelem přijatými přepisy. Nyní má uživatel možnost přepis pouze uzavřít a navrátit se do svého přehledu přepisů, nebo navigovat mezi přepisy a současně otevřený přepis případně i uzavřít. Díky tomu se také nemusí při jediném přijatém přepisu z aplikace nejprve vracet, aby jej uzavřel.

Kapitola 5

Implementace

Tato kapitola se věnuje zejména popisu struktury zdrojového kódu projektu a stručnému popisu vybraných implementačních detailů. Je zde také popsáno SpokenData API, se kterým byla aplikace propojena pro komunikaci s backendem, a formát dat jím přenášených. Finální podoba implementovaného prostředí lze pak vidět v příloze A.

5.1 Vývojový design aplikace

Aplikace využívá funkcionální komponenty, které mezi sebou komunikují zejména přes centralizovaný stav. Technologie umožňující tento vývojový design jsou popsány níže.

Na závěr této podkapitoly je popsána adresářová struktura projektu, která umožňuje snadnou orientaci a práci s jeho zdrojovými soubory.

React Hooks

Hooks[1] jsou funkce zavedené v Reactu od verze 16.8, které umožňují „zaháknutí“ do Reactových prvků stavu a životního cyklu z funkcionálních komponent bez použití tříd. Používání těchto prvků s třídovými komponentami často postupně vedlo k tvorbě komplexních a špatně čitelných struktur. Hooks umožňují extrakci stavové logiky z komponent a její znovupoužití bez změny hierarchie komponent. Mimo jiné umožnuje i rozdělení akcí prováděných ve fázích životního cyklu komponenty do separátních menších funkcí, a tak nesouvisející akce nemusí být míchány na jednom místě.

Redux

Redux[2] je knihovna a vývojový vzor pro správu stavu aplikace a manipulaci s ním přes události zvané „akce“. Vytváří centralizovanou úschovnu pro stav aplikace, který je možné použít v jakékoliv z částí aplikace.

Obyčejně pokud bychom měli vnořenou strukturu několika komponent a potřebovali bychom předat data z komponenty na nejvyšší úrovni komponentě na úrovni nejnižší, museli bychom tato data předat každé komponentě uprostřed hierarchie jako parametr i přes to, že je nepotřebují. Data uložená v centralizovaném stavu jsou přístupná v každé komponentě, které stačí si o ně jednoduše zažádat.

[1] https://reactjs.org/docs/hooks-intro.html
[2] https://redux.js.org/tutorials/fundamentals/part-1-overview

Manipulace se stavem probíhá voláním zvláštních funkcí, tzv. „tvořičů akcí“. Jednotlivé tvořiče akcí vždy tvoří akci nějakého konkrétního typu a volitelně mohou přijímat i parametry pro danou akci. Takto vytvořená akce je následně odeslána do tzv. „reduktoru“, který na základě typu akce a případně i na základě předaných parametrů stav zmanipuluje. Komponenty pak mohou na tyto změny stavu naslouchat a provést potřebné úkony.

Adresářová a souborová struktura projektu

Adresářová struktura projektu byla inspirována článkem[3] od Manujith Pallewatte. Na nejvyšší úrovni adresářové struktury zdrojových souborů projektu se nachází následující adresáře:

- Adresář `components`, obsahující veškeré funkcionální komponenty.
- Adresář `enums`, obsahující všechny číselníky (současně pouze číselníky barev segmentů a textových značek).
- Adresář `state`, který obsahuje veškeré části Reduxového centralizovaného stavu.
- Adresář `styles` se soubory kaskádových stylů.
- A adresář `utils` obsahující pomocné funkce.

V adresáři `components` jsou pak související či do sebe vnořené komponenty seskupeny do své vlastní složky. Nachází se zde také složka `renderless-components`, kde jsou umístěny komponenty, které nevykreslují žádné elementy do DOM. Ty jsou užitečné pro případy, kdy je třeba využívat plnou funkcionalitu komponent, avšak pouze pro manipulaci dat a stavu aplikace.

V adresáři `state` jsou vytvořeny zvláštní podadresáře pro každou z částí stavu aplikace, ve kterých je umístěn jak příslušný tvořič akcí, tak i reduktor. Výhodou tohoto strukturování oproti seskupení všech tvořičů akcí a reduktorů do jednoho adresáře je, že při přidávání nové akce stačí pracovat pouze s jediným adresářem dané části. Zároveň se tak části stavu dají lépe odebírat či přidávat. V současném rozsahu projektu by byla relativně schůdná i druhá varianta, avšak stav byl takto strukturován pro lepší udržitelnost v budoucnosti.

5.2 Technologie a způsob fungování backendu

Na výstupu strojového učení provádějící automatický přepis jsou data ve formátu JSON (JavaScript Object Notation), která jsou následně přenášena do frontendu pomocí REST API (Representational State Transfer Application Programming Interface), jehož struktura je zdokumentována pomocí Swaggeru.

REST API

REST API[4] je sada pravidel pro vzájemnou komunikaci aplikací či zařízení, která splňuje principy REST architektury. Nejen díky jeho vysoké míře flexibility se stal běžnou metodou pro komunikaci komponent a aplikací v mikroslužbové architektuře. Poprvé bylo definováno v roce 2000 doktorem Royem Fieldingem v rámci jeho disertační práce.

[3] https://www.pluralsight.com/guides/how-to-organize-your-react-+-redux-codebase
[4] https://www.ibm.com/cloud/learn/rest-apis

Komunikace REST API probíhá prostřednictvím HTTP (Hypertext Transfer Protocol) požadavků od klienta provádějících operace, jako jsou tvorba, čtení, úprava či smazání nějakého zdroje v databázi na serveru. Přenášená data mohou být prakticky v jakémkoliv formátu, nejpopulárnějším je však JSON (detaily a další příklady formátů viz další sekce).

Mezi zmíněné principy REST architektury patří šest bodů:

- Jednotné rozhraní – požadavky o jeden zdroj by měly vypadat identicky, ať už pochází odkudkoliv.
- Rozdělení klienta a serveru – klient a server musí být vzájemně zcela nezávislý, jediný způsob komunikace je prostřednictvím HTTP požadavků.
- Bezstavovost – každý požadavek musí obsahovat veškerá data potřebná k jeho zpracování, nejsou uchovávána žádná data o sezení.
- Možnost ukládání do mezipaměti – zdroje by měly být, pokud možno, ukládatelné do mezipaměti na straně klienta či serveru. Cílem tohoto je zlepšení výkonu na straně klienta a škálovatelnosti na straně serveru.
- Vrstvená architektura systému – požadavky a odpovědi prochází různými vrstvami a REST API musí být navržené tak, aby server ani klient nepoznal, zda komunikuje s cílovou vrstvou, nebo s prostředníkem.
- Kód na vyžádání (volitelné) – REST API typicky posílá statické zdroje, nicméně může posílat i spustitelný kód. Takto poslaný kód by měl být spuštěn pouze na vyžádání.

JSON

JSON [1] je formát používaný pro výměnu dat mezi různými platformami, podobně jako například XML (Extensible Markup Language), případně CSV (Comma-separated Values) pro spíše tabulková data. Je založený zejména na doslovné reprezentaci objektů s jejich vlastnostmi ve formátu dvojic název-hodnota. Ačkoliv je JSON založený na objektové notaci, po jeho dekonstrukci na primitivní datové typy (nesoucí elementární, dále nestrukturované údaje) je použitelný i pro jazyky s absencí objektového datového typu, je tedy vysoce univerzální.

Swagger

Swagger[5] je sada open-sourcových nástrojů založená na specifikaci OpenAPI (dříve Swagger specifikaci), což je formát pro kompletní popis REST API, včetně dostupných endpointů (koncových bodů rozhraní), jednotlivých operací na nich proveditelných spolu s jejich parametry, způsobu autentifikace a dalších informací. Mezi hlavní nástroje Swaggeru patří Swagger Editor, což je prohlížečový editor pro psaní OpenAPI specifikací a Swagger UI, který vyobrazuje OpenAPI specifikace jako interaktivní dokumentace.

Dokumentace vygenerovaná pomocí Swagger UI obsahuje kromě výše zmíněných prvků i příklady výstupů jednotlivých endpointů a možnost si volání endpointů vyzkoušet. Úsek takto vygenerované dokumentace SpokenData API lze vidět na obrázku 5.1.

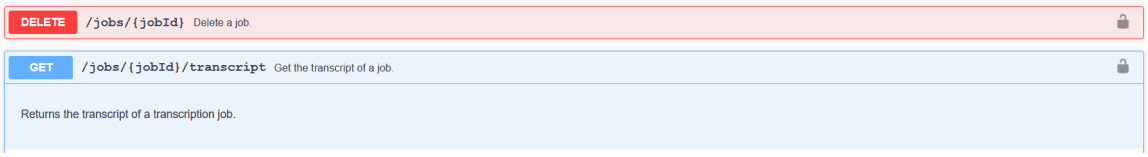
[5] https://swagger.io/docs/specification/about/

Struktura dat přepisu

Data přepisů jsou rozdělena do dvou struktur ve formátu JSON. V první z nich jsou data spojená s „jobem“ (pojem pro přepis určený k opravě a anotaci, uživatel přijímá jednotlivé joby, které vypracuje a následně je označí jako dokončené), mezi která patří:

- Název jobu
- Popis
- Status
- Odkaz na zvukovou stopu
- Dostupné značky mluvčích, segmentů a jednotlivých slov
- Odkazy na externí zdroje (manuály…)
- Přílohy (zejména textové a obrázkové)
- A další data a metadata…

V druhé struktuře je samotný přepis rozdělený do segmentů, které se dále dělí na jednotlivá slova. Příklad struktury segmentu lze vidět na výpisu 1.



Obrázek 5.1: Úsek dokumentace SpokenData API vygenerované pomocí Swaggeru

```
{
  "start": 5.6,
  "end": 7.21,
  "speaker": "B",
  "language": "English",
  "segment_tags": [
    "checked"
  ],
  "words": [
    {
      "label": "one",
      "text_tags": [
        "argument"
      ]
    }
  ]
}
```

Výpis 1: Příklad JSON struktury segmentu

5.3 Stručný popis vybraných částí implementace

Inicializace přepisu

Za inicializaci dat přepisu je zodpovědná komponenta `Initializator`. Ta ze současné URL načte ID upravovaného přepisu, pro které načte data jobu a následně data přepisu samotného pomocí příslušných dotazů na SpokenData API. Data jobu jsou uložena v úschovně reduktoru `JobReducer` a data přepisu v reduktoru `TranscriptReducer`, ze kterých si je jakákoliv komponenta, která s nimi potřebuje pracovat, může vyžádat.

Nástěnka

Komponenta nástěnky `Dashboard` je zodpovědná za správu zobrazení všech modulů. Při inicializaci nejprve zkontroluje, zda má uživatel v local storage (úložiště klienta v prohlížeči) již uložená data obsahující seznam otevřených komponent a jejich pozic, jež jsou ukládána po každé uživatelem provedené změně rozložení. Pokud ne, je použita výchozí konfigurace. Nástěnka má svůj reduktor `DashboardReducer`, který je zodpovědný za provádění akcí, jako je přidání/odebrání modulů z nástěnky, navrácení nástěnky do výchozího nastavení apod. Podporováno je zobrazování jak předem nastavených modulů, tak i modulů dynamických pro textové přílohy, kterých může být libovolný počet. Nástěnka je také responzivní a mění šířku sloupců mřížky plochy a pozice modulů s měnící se šířkou okna.

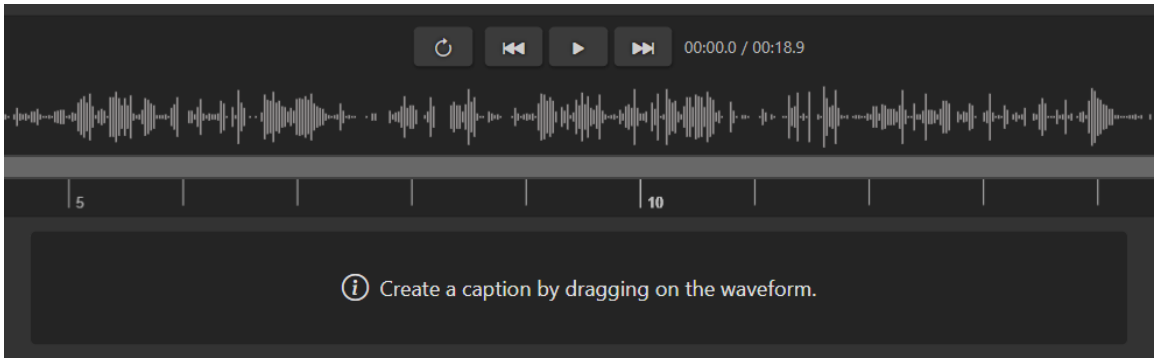
Textové segmenty

Za zobrazovaní segmentů a práci s nimi jsou zodpovědné 3 komponenty. Na nejvyšší úrovni je komponenta `AnnotationTextSegmentContainer`, která načte všechny segmenty ze stavu reduktoru přepisu a pro každý z nich vytvoří komponentu `AnnotationTextSegment`, na kterou vytvoří referenci, jež uloží do reduktoru `ReferenceReducer` akcí vyvolanou tvořičem `createActionSegmentReferencesInitialize`. Tyto reference jsou použity pro dynamické scrollování segmentů, viz níže. Jednotlivé komponenty `AnnotationTextSegment` si pak dle ID segmentu předaného jako parametr načtou z reduktoru přepisu potřebná data. Tato komponenta pak obsahuje komponentu `AnnotationTextEditor`, která je zodpovědná za úpravu textu přepisu a jeho anotaci.

Editor textu má formát obsahu odlišný od JSONového formátu načteného ze SpokenData API. Proto je třeba data textů segmentů načtená z API při prvotní inicializaci nejprve převést do tohoto formátu a při uložení přepisu je převést zpět. Stejně tak při operacích, jako je sloučení segmentů či rozdělení segmentu na dva, je třeba nejprve převést obsah editoru na JSONový formát. To je provedeno pomocí akce, jež zažádá konkrétní editory segmentů o převod a ty potom převedený text pošlou další akcí do reduktoru přepisu. V reduktoru je segment modifikován, a nakonec jsou dané editory akcí vyzvány k znovunačtení upraveného textu z JSONu.

Selekce textu provedená uživatelem je před jejím označením anotační značkou automaticky rozšířena na celá slova, jelikož označení části slova není možné. Díky tomu tak uživatel nemusí precizně selektovat celá slova a pro označení jednoho slova do něj stačí pouze umístit kurzor.

Při absenci existujících segmentů je uživatel nápovědou vyzván k jejich vytvoření přetažením po vizualizaci zvukové stopy, jak lze vidět na obrázku 5.2. Pokud tedy uživatel přepis vytváří od nuly, jeho první krok je obzvlášť jednoduchý a získává tak okamžité uspokojení ze své práce s prostředím, jak bylo popsáno v druhé kapitole v podsekci 2.1.2.



Obrázek 5.2: Vzhled modulu s textovými segmenty při absenci existujících segmentů

Přehrávač zvukové stopy

Přehrávač (komponenta `AudioPlayer`) si při své inicializaci získá URL zvukové stopy z dat jobu a z dat přepisu si načte veškeré segmenty mluvčích, ze kterých vytvoří regiony na vizualizaci zvukové stopy. Tvorba a úprava segmentů probíhá pomocí tvořičů akcí `createActionTranscriptSegmentCreate` a `createActionTranscriptSegmentUpdate` volaných na základě příslušných událostí Wavesurferu (použité knihovny pro přehrávač), které v reduktoru `TranscriptReducer` data segmentů upraví.

Přehrávač má taktéž svůj vlastní reduktor `AudioReducer`, který obsahuje akce pro různé operace pro přehrávání stopy, na jejichž vyvolání přehrávač reaguje. Při vstoupení přehrávače do rozsahu z některého ze segmentů je nalezena reference na komponentu příslušeného textu segmentu a ta je scrollována do viditelného prostoru.

Historie operací

Ukládání provedených operací a pohyb mezi nimi probíhá na dvou úrovních. Zaprvé jsou v reduktoru `HistoryReducer` ukládány prováděné operace pouze s informací o tom, odkud daná operace přišla. Je zde také uchováván index současné operace, který je modifikován přidáváním nových operací a posunem o krok vzad či vpřed. Samotné detailní informace o operacích, jež jsou třeba pro jejich vrácení či znovuprovedení, jsou ukládány lokálně v komponentách či reduktorech, které jsou za danou část stavu, nad kterou byla operace provedena, zodpovědné. Ty svůj stav modifikují v reakci na příslušné akce v `HistoryReducer`.

V případě, že operace byla provedena nad přepisem (kromě samotného textu přepisu), její historie je zpracovávána v reduktoru `TranscriptReducer`. Zde je pro každou akci uchováván její typ a data umožňující její navrácení či znovuprovedení, v tomto případě tedy související segmenty, se kterými bylo operací manipulováno. Pro správu operací úpravy textu přepisu je pak utilizován zásuvný modul použité rich text editor knihovny Slate.

Klávesové zkratky

Klávesové zkratky jsou ukládány v reduktoru `HotkeyReducer` v lidsky čitelném řetězci, do kterého jsou z událostí zmáčknutí kláves převedeny pomocí knihovny „key-event-to-string“. V tomto řetězcovém formátu jsou zmáčknutí kláves `Ctrl` (Windows) a `Cmd` (Mac OS) uloženy jako „Ctrl/Cmd“, tudíž jsou zaměnitelné. Úprava zkratek probíhá v okně nastavení jednoduše zakliknutím zkratky ke změnění, načež je započato naslouchání na události zmáčknutí kláves, a následném zmáčknutí cílové kombinace, která je do reduktoru

uložena tvořičem akcí `createActionHotkeySet`. Zkratky jsou zároveň ukládány do local storage prohlížeče klienta, odkud jsou případně při inicializaci znovu načteny.

Za samotné naslouchání na zmáčknutí nastavených zkratek a provádění souvisejících operací je zodpovědná komponenta `HotkeyListener`, která zmáčknuté klávesy převádí do výše zmíněného formátu a porovnává je s nastavenými klávesovými zkratkami. V případě nalezení shody je pak zablokována výchozí reakce prohlížeče na zkratku pro prevenci případné kolize zkratek a následně je vyvolána související akce.

Kapitola 6

Testování

Začátek této kapitoly se zabývá popisem způsobu testování použitelnosti s účastníky v rámci testovacích sezení a způsobu hodnocení celkové použitelnosti účastníky po dokončení jejich sezení. Následně jsou získané znalosti použity v praxi a je popsán průběh samotného testování aplikace spolu s vyhodnocením testování na základě získaných poznatků. Kapitola je zakončena detailní diskuzí o výhledech na budoucí vývoj.

6.1 Testování použitelnosti

Jak je uvedeno v knize Rocket Surgery Made Easy [3] od Steva Kruga, ze které bude čerpat tato podkapitola, testování použitelnosti se dá rozdělit na dvě skupiny – **kvantitativní** a **kvalitativní**. Při kvantitativním testováním chceme něco dokázat, například zda je náš produkt lépe použitelný než produkt konkurence, nebo jestli je nová verze našeho produktu lepší než ta předešlá. Toto provádíme pořizováním metrik, jako je rychlost provedení úkolu, podíl úspěšnosti provedení úkolu apod. Nevýhodou tohoto typu testování je, že si vyžaduje značně vědecký přístup – musíme si přesně definovat testovací protokol, od kterého nemůžeme odbočit, musíme získat dostatečně velký vzorek účastníků, aby naše výsledky byly statisticky významné a účastníci musí zároveň být ze skupiny skutečných cílových uživatelů našeho produktu, abychom výsledky mohli extrapolovat na větší populaci. Musíme tedy přesně vědět, co děláme, a aplikovat při tom velkou míru obezřetnosti.

Dále bude rozebírán druhý typ testování použitelnosti – kvalitativní. Při kvalitativním testování nám nejde o to něco dokázat, nýbrž získat podklady k vylepšení vyvíjeného produktu. Díky tomu můžeme použít mnohem méně formální přístup a stačí nám menší vzorek účastníků, pokud od nich získáme potřebné znalosti. Není třeba za každou cenu zachovávat stejný testovací protokol pro každého účastníka – například pokud zjistíme, že je nějaký úkol špatně proveditelný a víme proč, u dalšího účastníka jej můžeme přeskočit. Kdybychom to samé udělali u kvantitativního testování, znehodnotili bychom si výsledky.

Kvalitativní testování ve zkratce probíhá tak, že si facilitátor sedne do místnosti s účastníkem, zadá mu nějaké úkoly a řekne mu, aby při jejich provádění přemýšlel nahlas. Zbytek vývojového týmu a další zainteresované osoby průběh testu sledují přes software na sdílení obrazovky a po konci testů se sejdou, porovnají si poznámky, identifikují nejzávažnější problémy a rozhodnou se, které z nich v blízké době opraví. Pokud je nějaký problém obtížný na opravení a zároveň je pravděpodobné, že na něj většina uživatelů vůbec nenarazí, může se vyplatit jeho opravení odložit. Tento typ testování má dobré výsledky, protože když před

vyvíjený produkt posadíte téměř kohokoliv, nevyhnutelně narazí na některé problémy, na které narazí i většina uživatelů.

Tvorba testovacího protokolu

Pokud chceme účastníky sledovat, jak pracují s naším produktem, musíme jim vymyslet, co konkrétně mají dělat. To lze rozdělit do dvou kroků:

- Nejprve určíme **úkoly**, které chceme, aby účastník vyzkoušel. Příkladem úkolu může být nalezení spoje hromadné dopravy na webu vytvořeném pro tento účel.
- Následně na základě vymyšlených úkolů vytvoříme **scénáře**, které účastníkovi dají kontext potřebný k vykonání úkolů. Scénář by tedy již obsahoval detailní informace, jako je například čas, destinace, typ dopravy a důvod, proč účastník chce spoj vůbec nalézt.

Role facilitátora

První z povinností facilitátora je udělování úkolů účastníkům a kontrola jejich postupu. Nesmíme však účastníkům žádným způsobem napovídat, je totiž na nich samotných, aby na řešení úkolů přišli. Druhou povinností je donutit účastníky přemýšlet nahlas – měli by říkat, na co se dívají, co se snaží udělat, co jim dělá problém apod.

Právě díky přemýšlení nahlas je testování použitelnosti tak efektivní. Nahlédnutí do mysli někoho, kdo o předmětu testování neví zdaleka tolik jako my, nám dá cenné poznatky, které bychom jinak nebyli schopni získat. Nejvíce nás zajímají projevy negativních emocí, jako je frustrace a zmatenost. Pokud účastník přestane mluvit, můžeme se jej zeptat, nad čím přemýšlí. Nemusíme se však zbytečně ptát pořád dokola, pokud je vidět, že účastník nemá s úkolem problém – stačí se jej zeptat, pokud je například zmatený a skutečně nevíme, o čem přemýšlí.

Je také třeba zůstat naprosto neutrální, jelikož účastníka nechceme žádným způsobem ovlivňovat. Pokud víme o nějakém našem zaujetí vůči předmětu testování, musíme ho potlačit. I pokud žádné zaujetí nemáme, musíme se vyhnout napovídání, odpovídání na otázky účastníka (většinu otázek je třeba zodpovědět opět otázkou) a vyjadřování jakýchkoliv názorů či pocitů.

Úvod testovacího sezení

Testovací sezení vždy začíná přečtením základních instrukcí, které účastníkovi sdělí obecné informace o průběhu testování. Je vhodné připravené instrukce opravdu přečíst a vyhnout se improvizaci, jelikož je velká šance, že je nevhodně zformulujeme a účastník si o něčem vytvoří mylnou představu.

Po základních instrukcích se účastníka můžeme zeptat na několik předtestových otázek za účelem získání představy o jeho znalosti v oblasti předmětu testování a jeho celkové počítačové zdatnosti. Tím získáme kontext pro lepší interpretaci průběhu testování a zároveň účastníka rozmluvíme. Můžeme se zeptat například na jeho povolání, kolik času tráví průměrně denně na počítači, co na počítači dělá nejčastěji apod.

Provádění úkolů

Po předtestových otázkách přejdeme k jádru testu – k samotným úkolům. Na začátku každého úkolu dáme účastníkovi text aktuálního scénáře, který následně doslovně přečteme.

Scénář předčítáme z toho důvodu, že někteří účastníci si jej sami nemusí přečíst pořádně a alespoň se tak ujistíme, že jej slyšeli v plném znění.

Jakmile účastník začne s prováděním úkolu, je třeba jej vyrušovat co možná nejméně. Jeho koncentrace musí být upřena na úkol a na přemýšlení nahlas. Jakmile účastník úkol dokončí, pokračujeme dalším scénářem. Pokud si účastník mylně myslí, že úkol dokončil, můžeme se ho zeptat, jestli by to mohl zkusit jiným způsobem, díky čemuž si svoji chybu většinou uvědomí. Pokud vidíme, že se účastník s úkolem opravdu trápí, nemá smysl jej v tom příliš dlouho nechávat a můžeme se přesunout k dalšímu úkolu. To samé platí, pokud úkol účastníkovi zkrátka trvá příliš dlouho a nemáme na to čas, nebo pokud dostaneme pocit, že se pokračováním již nic nenaučíme – v tom případě můžeme ještě chvíli počkat a následně jít dále, přičemž v polovině případů se v této době navíc ještě stane něco užitečného.

Pokud účastník úkol nedokončí a my se rozhodneme jít dále, počkáme na přirozenou pauzu a řekneme mu něco ve smyslu „Skvěle, to bylo velmi užitečné. Teď se přesuneme dále, protože toho ještě máme hodně před sebou.“. Věta je úmyslně napsána v množném čísle, protože nechceme nijak naznačit, že se posouváme kvůli selhání účastníka.

Dotazování

Jelikož v průběhu testování nechceme odvádět účastníkovu pozornost dotazy, je dobré si nechat trochu času po konci na doptání na detaily, co nás v průběhu testování zajímaly. Není problém se v průběhu zeptat na jednoduché otázky s krátkou odpovědí, ale ty hlubší otázky bychom si měli sepsat a nechat až na konec. Typicky se účastníků chceme zeptat na věci, jako proč si něčeho všimli nebo proč udělali nějaká konkrétní rozhodnutí. Můžeme je i v případě potřeby k nějakému úkolu vrátit a například je požádat, aby jej provedli jiným způsobem nebo z jiného počátečního bodu.

6.2 Hodnocení použitelnosti po testovacím sezení

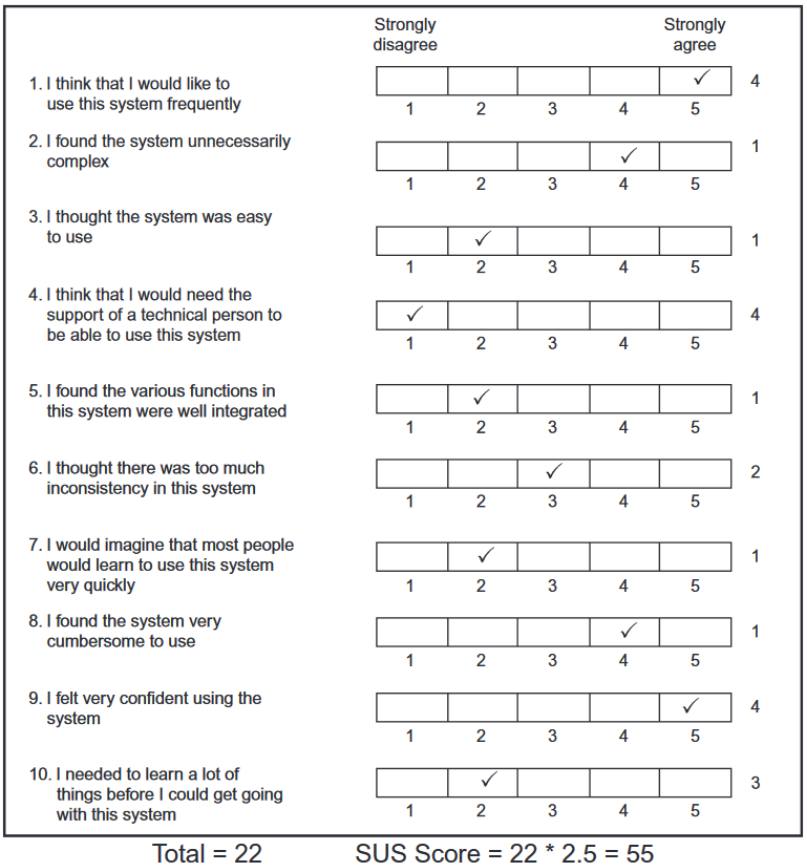
Tato sekce čerpá z principů uvedených v knize Measuring the User Experience [7] od Thomase Tullise a Williama Alberta, kde je uvedeno, že jedním z nejčastějších použití metrik nahlášených účastníky testování je měření celkové použitelnosti na základě informací od nich získaných po dokončení jejich interakce s testovaným prostředím. Výsledky těchto měření pak lze použít pro referenci celkové použitelnosti napříč produkty či jednotlivými verzemi produktů zejména v tom případě, že si zvolíme jediný způsob jejich vyhodnocení a výsledky si průběžně uschováváme.

Je však důležité odlišovat data sbíraná v průběhu sezení, kdy účastníkům zadáváme úkoly zaměřené na interakci s prostředím a účastník je následně průběžně hodnotí, a data sbíraná až po dokončení celého sezení. Agregací dat sbíraných průběžně získáme přehled o průměrném vnímání použitelnosti v průběhu času. Naproti tomu agregací dat sbíraných po dokončení sezení získáme představu o posledním dojmu účastníka z celé zkušenosti s prostředím. Poslední dojem je to, s čím účastník odejde a co velmi pravděpodobně ovlivní jeho budoucí rozhodnutí o daném produktu. Následující sekce se bude zabývat technikami pro získání dat po sezení a jejich vyhodnocení.

System Usability Scale

Jde o jeden z nejčastěji používaných nástrojů pro posuzování použitelnosti systémů či produktů. Byl vyvinut Johnem Brookem v roce 1986 v rámci jeho práce v Digital Equipment Corporation. Skládá se z deseti výroků, k nimž účastník testování vyjádří míru svého souhlasu na škále 1 až 5, kde vyšší číslo znamená vyšší míru souhlasu s daným výrokem. Polovina z výroků vyjadřuje pozitivní názor a druhá polovina názor negativní k testovanému produktu. Součástí tohoto systému je i technika pro kombinaci všech 10 odpovědí do celkového skóre, které se pohybuje na škále od 0 do 100, kde 100 reprezentuje perfektní skóre.

Výpočet SUS skóre pak probíhá následujícím způsobem. Nejprve jsou sečtena skóre získaná ve všech částech dotazníku. Skóre získané z části neodpovídá zvolenému číslu na škále, nýbrž pro části 1, 3, 5, 7 a 9 odpovídá číslu na škále sníženému o jedna a pro zbylé části odpovídá číslu na škále odečtenému od pěti. Součet skóre je pak vynásoben 2,5, čímž je získáno výsledné SUS skóre. Takto vypočítaná skóre od jednotlivých účastníků pak mohou být agregována aritmetickým průměrem. Podoba dotazníku a příklad výpočtu lze vidět na obrázku 6.1.



Obrázek 6.1: Dotazník SUS s příkladem jeho vyhodnocení (převzato z literatury [7])

SUS je volně použitelný pro studie použitelnosti, a to jak pro výzkum, tak i pro komerční použití, za předpokladu uvedení jeho zdroje. I díky tomu je široce rozšířený a na základě analýzy mnoha různých SUS skóre pro širokou škálu systémů bylo stanoveno doporučení pro interpretaci SUS skóre:

- Pod 50: Nepřijatelné
- V rozsahu 50-70: Hraniční
- Nad 70: Přijatelné

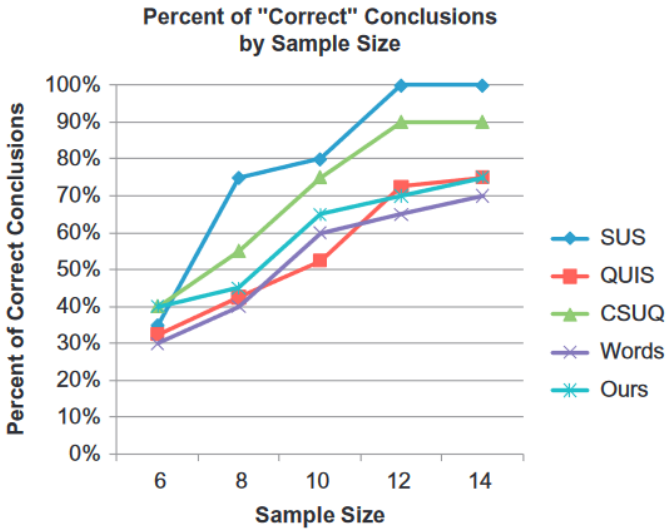
Je sporné, zda je vhodné v dotazníku mít jak pozitivní, tak negativní výroky. Někteří tvrdí, že negativní výroky pomáhají udržet pozornost účastníka a jiní, že účastníka mohou zmást a způsobit tak chybné odpovědi. V roce 2011 provedl Jeff Sauro spolu s Jimem Lewisem studii, kde porovnali tradiční verzi SUS a verzi pouze s pozitivními výroky. Nenalezli žádnou výraznou odchylku mezi průměrnými SUS skóre tradiční verze a verze s pouze pozitivními výroky. Avšak při zkoumání 27 sad dat SUS skóre byly nalezeny důkazy, že 11% studií, ze kterých data pocházela, obsahovala chybné kódování a 13% jednotlivých SUS dotazníků obsahovalo chyby od účastníků. Pro vyhnutí těmto možným problémům tedy doporučují použití verze SUS s pouze pozitivními výroky. Příklad verze s pouze pozitivními výroky je uveden níže.

Computer System Usability Questionnaire

CSUQ byl vyvinut Jimem Lewisem v roce 1995 pro provedení celkového zhodnocení použitelnosti systému na konci testovací studie. Je velmi podobný Post-Study System Usability Questionnaire od stejného autora, s tím rozdílem, že CSUQ byl navržen pro vyplnění přes mail nebo online a PSSUQ pro vyplnění osobně. Skládá se celkově z 19 pozitivních výroků, ke kterým účastník vyjadřuje míru souhlasu na stupnici od 0 do 7, kde 7 znamená naprostý souhlas a účastník má také možnost výrok přeskočit.

Srovnání způsobů hodnocení použitelnosti po sezení

Ve studii provedené Thomasem Tullisem a Jacquelinem Stetsonem v roce 2004 byla použita řada různých typů dotazníků pro hodnocení použitelnosti po testovacím sezení pro změření uživatelských reakcí na dvě webové stránky. Mezi použitými dotazníky byly SUS, QUIS (Questionnaire for User Interaction Satisfaction), CSUQ, Microsoft's Product Reaction Cards a další proprietární dotazník. Všechny tyto dotazníky byly mírně upraveny pro použití pro testování webových stránek (například slova systém byla nahrazena za webovou stránku apod.). Do studie bylo zapojeno celkem 123 účastníků a každý z nich po konci sezení o dvou úkolech použil jeden z dotazníků pro zhodnocení obou webových stránek. Výsledky studie jsou vizualizovány na obrázku 6.2. Ze všech z pěti typů dotazníků vyšlo najevo, že stránka 1 získala podstatně lepší hodnocení, než stránka 2. Data byla následně dále analyzována pro zjištění, jak se výsledky budou měnit při změně velikosti vzorku od 6 po 14. Při velikosti vzorku 6, pouze 30–40% vzorků by označilo stránku 1 jako podstatně lepší. Avšak při velikosti vzorku 8, která je při testech použitelnosti velmi častá, by SUS označil stránku 1 jako tu podstatně lepší v 75% případech, což je drasticky vyšší hodnota než u ostatních metod. Jedním z možných vysvětlení drasticky vyšší přesnosti při nízkém počtu vzorků je potvrzení argumentu o lepším udržení pozornosti účastníka při kombinaci negativních a pozitivních výroků.



Obrázek 6.2: Graf ilustrující přesnost výsledku při změně velikost vzorků od 6 po 14 (převzato z literatury [7])

6.3 Testování použitelnosti implementované aplikace

Testování použitelnosti proběhlo ve dvou kolech dle postupu kvalitativního testování popsaného v této kapitole v sekci 6.1. Každý účastník po svém testovacím sezení taktéž dostal k vyplnění použitelnostní dotazník SUS, jež byl vybrán pro jeho značně vyšší přesnost při nízkém počtu vzorků. Aplikaci se také úspěšně podařilo nasadit na web SpokenData.

Komunikace s účastníky probíhala přes schůzku na aplikaci Zoom, přičemž práce s testovanou aplikací probíhala přes vzdálené ovládání plochy aplikací TeamViewer. Průběh sezení byl také nahráván přes aplikaci OBS pro pozdější zkoumání.

Při každém úkolu byl účastníkovi nejprve přečten scénář, který měl zároveň vždy před očima na sdílené obrazovce, a následně účastník úkol prováděl, přičemž verbalizoval svoje myšlenky. Na závěr sezení byl každému účastníkovi k vyplnění odeslán použitelnostní dotazník SUS popsaný v této kapitole v sekci 6.2.

Testovaná funkcionalita

Úkoly byly zaměřené v první řadě na základní operace, které jsou jádrem aplikace a musí být funkční a dobře použitelné, jde tedy zejména o:

- Opravu přepisu segmentu
- Označování sekcí textu
- Vkládání značek mezi text
- Tvorba, úprava a použití značek mluvčích
- Označování segmentů
- Dohledání a otevření anotačního manuálu
- Otevření příloh

- A tvorbu a úpravu segmentů na zvukové stopě

Dále byly testovány vybrané nově přidané funkce, které by teoreticky nemusely být každému jasné a u kterých bylo třeba míru jejich použitelnosti ověřit:

- Zpomalení rychlosti přehrávání
- Práce s přizpůsobitelnou nástěnkou modulů – zavření, otevření a pozicování modulů
- Přenastavení klávesových zkratek

Testovací protokol je v plném znění k dispozici v příloze B.

6.3.1 První kolo testování

Účastníci prvního kola testování byli celkem tři, z nichž všichni byli značně počítačově zdatní, první z nich měl rozsáhlé zkušenosti s používáním původní aplikace služby SpokenData, druhý ji pouze jednou přibližně před rokem krátce vyzkoušel a poslední neměl žádné zkušenosti s anotačními aplikacemi.

Poznatky ze sezení prvního účastníka

První z účastníků měl rozsáhlé zkušenosti s původní aplikací, které mu paradoxně způsobily problémy s úkolem zahrnujícím vytvoření nového segmentu. Účastník totiž hledal tlačítko z původní aplikace pro přidání nového segmentu, a tak jej nenapadlo zkusit segment „nakreslit“ přetáhnutím po vizualizaci zvukové stopy.

Způsob tvorby segmentů je popsán v nápovědě, která je viditelná pouze, pokud neexistují žádné segmenty. Při použití aplikace pro opravu automatického přepisu však vždy alespoň jeden segment již existuje. Nabízí se tedy otázka, zda by bylo vhodné nápovědu přidat ještě jinou formou. Na druhou stranu ostatní účastníci způsob přidání segmentu odhalili bez větších problémů.

Účastník nebyl spokojen se způsobem fungování zpomalení rychlosti přehrávání, které spolu s rychlostí snižuje i tón nahrávky. Tato nedokonalost silně snižuje jeho použitelnost pro vysokou míru zpomalení.

Dále účastník upozornil na mírnou nekonzistenci ve vkládání značek mezi text. Má totiž již z předchozích zkušeností zapamatovanou textovou formu těchto značek z manuálu a více mu vyhovuje je vepisovat manuálně. Ačkoliv manuální vepisování mezitextových značek funguje, nejsou takto přetransformovány do grafické podoby, jako při jejich vkládání příslušným tlačítkem značky.

Poznatky ze sezení druhého účastníka

Druhý účastník, který s anotačními aplikacemi neměl žádné zkušenosti, měl problém nalézt rozbalovací seznam se značkami segmentů, resp. jej našel a rozbalil, ale z nějakého důvodu si jméno seznamu a položky v něm vůbec nepřečetl a ihned jej zavřel. Dále se pak snažil segment označit kliknutím na jeho rozsah na vizualizaci zvukové stopy. Ačkoliv kliknutí levým tlačítkem příliš nedává smysl, mohlo by označení segmentu a případně další operace být alternativně prováděny přes kontextové menu otevřené kliknutím pravým tlačítkem myši na segment na zvukové stopě.

Při použití tlačítek pro přetáčení nahrávky očekával, že budou fungovat pro přeskakování mezi jednotlivými segmenty, což je dobrý postřeh a tato funkce by jistě byla více užitečná než obyčejné přetáčení, které se hodí spíše pro klasické přehrávače.

Poznatky ze sezení třetího účastníka

Třetí a poslední účastník testování měl pouze krátkou zkušenost s původní aplikací. Při úkolu zahrnujícím opravu značky sekce textu jej nejprve nenapadlo nesprávnou značku nejdříve odstranit a text znovu označit správnou značkou. Současná implementace nedovoluje označit text, který již označený je – nic se v tomto případě nestane. Při zachování tohoto způsobu fungovaní by uživatel měl dostat zpětnou vazbu o selhání označení, alternativou pak je existující značky automaticky odstranit a přidat novou.

Při úkolu, který vyžadoval otevření okna nastavení, měl účastník nejprve problém jej nalézt. Po skončení sezení jsem se ho zeptal, proč si myslí, že jej nemohl nalézt. Očekával, že menu s nastavením bude více konvenčně umístěné, například na vrchu obrazovky. Jak jsem však zmínil ve čtvrté kapitole v sekci 4.2, kvůli automatickému plutí modulů na nástěnce do nejvyšší možné pozice je nejvhodnější umístění pro okno nastavení v dolní části obrazovky, jelikož tam bude nejméně zavazet ve výhledu. Tlačítko pro otevření okna nastavení je však poměrně nevýrazné, tudíž jeho grafické zvýraznění by pravděpodobně značně usnadnilo jeho nalezení.

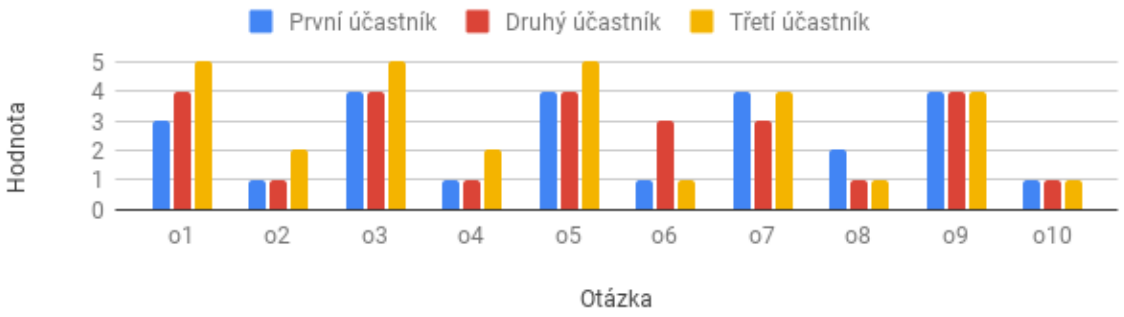
Vyhodnocení prvního kola testování

Co se samotné organizace týče, testování proběhlo na první pokus bez větších zádrhelů a zadané úkoly byly pro účastníky dobře pochopitelné.

Většinu úkolů byli účastníci schopni provést rychle a bez problémů, přičemž žádný z úkolů se neukázal být špatně proveditelný pro více než jednoho účastníka. Každý z účastníků však poskytl užitečné poznatky, které mohou být použity pro vylepšení použitelnosti aplikace. Na základě zvážení poměru obtížnosti opravy a závažnosti nalezených problémů byly opraveny ty následující:

- Tlačítko pro otevření okna nastavení bylo graficky zvýrazněno pro jeho snazší nalezení.
- Povoleno označování již označených sekcí textu, existující značky jsou přepsány.

Výsledné použitelnostní skóre SUS z tohoto kola testování, spočtené aritmetickým zprůměrováním skóre z jednotlivých dotazníků, je po zaokrouhlení rovno **84,17**, což je dle doporučené interpretace popsané v této kapitole v sekci 6.2 v rozsahu „přijatelné“ včetně dostatečné rezervy. Surová data z dotazníků lze vidět na obrázku 6.3.



Obrázek 6.3: Surová data SUS dotazníků z prvního kola testování

6.3.2 Druhé kolo testování

Druhého kola testování se opět zúčastnily tři osoby, z nichž dvě byly značně počítačově zdatné, avšak bez jakýchkoliv zkušeností s anotačními aplikacemi, a jedna byla letecký nadšenec se zkušenostmi s původní aplikací, tudíž z cílové skupiny uživatelů aplikace. Testovací scénáře zůstaly oproti prvnímu kolu beze změn.

Poznatky ze sezení prvního účastníka

První účastník neměl žádné předešlé zkušenosti s anotačními aplikacemi. Při plnění úkolu zahrnující přehrání segmentu očekával, že jeho přehrávání bude schopen tlačítkem pro přehrání segmentu opět zastavit. Tato funkcionalita by pravděpodobně měla být zahrnuta (původní aplikace ji taktéž neměla), nicméně její implementace je mírně problematická a byla přeskočena z časových důvodů. Bylo by třeba s každým pozastavením získat aktuální pozici v přehrávání a uložit nový interval od dané pozice do konce segmentu, v rámci kterého bude nahrávka přehrána při příštím přehrání segmentu. Tento uložený interval by následně musel být resetován při přehrání segmentu jiného, přehrání nahrávky jako celku nebo při jakémkoliv manuálním posunutí pozice přehrávání. Navíc by tak uživatel bez přidání dalšího tlačítka ztratil možnost segment opětovně přehrát od začátku bez jeho přehrání do konce.

Účastník měl dále chvíli potíže s identifikací rozbalovacího seznamu s možnostmi pro snížení rychlosti přehrávání nahrávky kvůli absenci popisku, jediný popisek je obsažen v tooltipu zobrazeném po najetí myší. Ačkoliv žádný z účastníků v zásadě neměl s nalezením tohoto prvku velký problém, popisek by jistě měl být na první pohled viditelný pro eliminaci jakýchkoliv pochybností.

Poznatky ze sezení druhého účastníka

Druhý účastník byl taktéž nezkušený s anotací. Stejně jako předchozí účastník poukázal na absenci popisku u změny rychlosti přehrávání. Taktéž mu chvíli trvalo nalezení možnosti pro otevření přílohy seznamu callsignů, což jsem do určité míry pozoroval i u některých dalších účastníků. Je otázkou, zda by přílohy neměly být lépe graficky zvýrazněné, aby lépe uchytily pozornost uživatele bez přečtení popisků, jelikož jsou pro anotaci v některých situacích značně nápomocné.

Poznatky ze sezení třetího účastníka

Třetí účastník měl zkušenosti s původní aplikací, avšak nebyl velmi počítačově zdatný, což vedlo k jisté míře zmatenosti a delší době potřebné na provedení několika úkolů. Stejně jako první účastník z prvního kola testování nebyl schopen nalézt způsob přidání nového segmentu. Všichni ostatní účastníci s nulovou či velmi letmou zkušeností s původní aplikací byli schopni intuicí segment vytvořit, avšak uživatelé původní aplikace měli pohled zkreslený a hledali tlačítko pro přidání segmentu z původní aplikace. Taktéž se pochopitelně pokusil o označení sekce textu vyklikáváním, jako v původní aplikaci, nicméně po selhání ihned došel ke správnému způsobu.

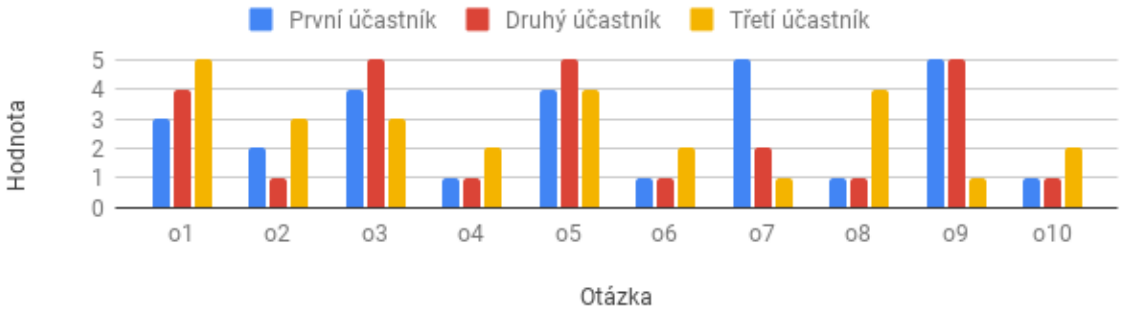
I přes veškeré zaměření na intuitivnost a použitelnost můžou jednotlivcům některé prvky dělat potíže. Zejména pro méně počítačově zdatné uživatele a uživatele původní aplikace by měl být dostupná nějaká forma návodu, ať už zabudovaná do prostředí nebo jako separátní dokument.

Vyhodnocení druhého kola testování

Výsledné průměrné použitelnostní skóre SUS z toho kola testování je rovno **76,67**, což je stále v přijatelném rozsahu. Nižší skóre oproti prvnímu kolu testování je způsobeno hodnocením třetího účastníka, které bylo těsně nad spodní hranicí rozsahu „hraniční“. Surová data z dotazníků lze vidět na obrázku 6.4.

Ačkoliv je vždy prostor pro zlepšení použitelnosti, je třeba přijmout fakt, že jde o relativně komplikovanou aplikaci a někteří uživatelé mohou potřebovat k provedení některých úkonů návod. Vhodný by byl návod pro nové uživatele zabudovaný přímo do rozhraní, který by je postupně provedl hlavní funkcionalitou aplikace. Velká část uživatelů si totiž pravděpodobně v lepším případě separátní manuál otevře až po dlouhé řadě neúspěšných pokusů, v tom horším případě úplně ztratí o aplikaci zájem.

Na základě získané zpětné vazby byl k vícevýběrovému seznamu pro změnu rychlosti přehrávání přidán popisek pro jeho snazší identifikaci.



Obrázek 6.4: Surová data SUS dotazníků z druhého kola testování

6.4 Testování nasazené aplikace

Aplikace byla ve spolupráci s panem Ing. Josefem Žižkou nasazena na web SpokenData jako alternativní anotační editor pro účely otevřeného testování. Nasazená aplikace používá novou verzi SpokenData API, která byla panem Žižkou průběžně dovyvíjena a také debugována na základě mé zpětné vazby. Do aplikace byla zabudována knihovna pro monitorování uživatelských sezení LogRocket[1] a do záhlaví stránky s aplikací byl vložen odkaz na volitelný dotazník SUS. Kvůli absenci aktivních uživatelů se však tímto způsobem testování podařilo opravit pouze jednu chybu a sice nesprávnou změnu focusu editorů textu segmentů, vedoucí k chybnému vložení značky do vícero segmentů. Nasazení tedy alespoň demonstruje integrovatelnost do webu SpokenData.

6.5 Výhledy pro budoucí vývoj

Tato sekce se zabývá funkcionalitou, která z časových důvodů nebyla implementována, a aspekty, na kterých je třeba dále zapracovat, včetně neopravených problémů zjištěných v rámci testování.

[1] https://docs.logrocket.com/docs

Velké cíle

- **Podpora překryvu/kombinace značek sekcí textu** – současné rozhraní s překryvem značek nepočítá a pro podporu všech možných překryvů by bylo třeba tlačítko pro odstranění značky přesunout na jiné místo (současně je vysouváno z kraje značky po najetí myší). Bez zobrazení množiny značek vedle textu jako v anotační službě Deepsy není možné ve všech situacích zajistit snadnou vizuální identifikaci jednotlivých kombinovaných značek a jelikož je rozložení aplikace dynamické a po stranách mohou být další moduly, tento přístup nelze použít.

  Alternativou je tedy zobrazení množiny značek v tooltipu zobrazeném po najetí myší na dané označené místo v textu, což je stále dostatečně rychlé. Možnost pro odstranění jednotlivých značek by pak byla dostupná skrze kontextové menu otevřené kliknutím pravým tlačítkem na dané místo. Implementaci překryvu značek hodnotím jako značně obtížnou a je tedy vhodné se o ni pokusit pouze, pokud bude opravdu potřebná.

- **Podpora anotace vícekanálového zvuku** – použitá knihovna pro přehrávač Wavesurfer.js podporuje rozdělení zvukové stopy na jednotlivé kanály[2]. Ty pak mohou mít každý svoji vlastní vizualizaci zvukové stopy, které mohou být zobrazeny pod sebou, nebo překryté na sobě s odlišnými barvami. Je však třeba prozkoumat dostupné možnosti použití, zda lze například přehrát pouze jeden z kanálů zvuku a zejména kompatibilitu zásuvného modulu pro regiony/segmenty na zvukové stopě. Implementace tak může být buďto poměrně jednoduchá, nebo naprosto nemožná (bez vlastních zásahů do zdrojového kódu knihovny), vše závisí na možnostech podporovaných knihovnou.

- **Nápovědy/návod zabudovaný do rozhraní** – zabudování návodu přímo do rozhraní není obtížné, otázkou je však jeho forma. Jednou z možností je návod pro nové uživatele v podobě malého vyskakovacího okna v rohu obrazovky. Okno by obsahovalo šipky pro pohyb mezi částmi návodu a každá část návodu by spolu se zobrazením instrukcí také vizuálně zvýraznila související moduly/tlačítka apod. Noví uživatelé by mohli být identifikování pomocí uložení příznaku při prvním otevření aplikace do local storage prohlížeče, nebo pro stoprocentní přesnost i napříč prohlížeči a zařízeními do SpokenData API. Návod by bylo možné přeskočit a následně kdykoliv znovu spustit v modulu nastavení.

Menší cíle

- **Transformace textové formy značek vkládaných mezi text do grafické podoby** – tyto značky jsou nově součástí rozhraní, nicméně uživatelé původní aplikace mohou preferovat jejich manuální vepisování. Manuální vepisování funguje taktéž, ačkoliv takto vepsané značky jsou ponechány v textové formě nekonzistentní se značkami vloženými skrz prostředí. Textovou formu značky lze automaticky přetransformovat například pomocí normalizace[3] z knihovny Slate. Implementace není nijak zvlášť obtížná, ale je třeba mít značnou míru porozumění principů, na kterých knihovna funguje.

[2] https://wavesurfer-js.org/example/split-channels/index.html
[3] https://docs.slatejs.org/concepts/11-normalizing

- **Dynamické sbalení segmentových značek do vícevýběrového seznamu** – současně jsou segmentové značky vždy sbaleny do vícevýběrového seznamu, jelikož jich může být více, než se do prostředí fyzicky vleze. Používání segmentových značek je tak o něco pomalejší a uživatel ihned nevidí, kterými konkrétními značkami je segment označený. Značky by tak mohly být do určitého počtu zobrazeny jako tlačítka napřímo (jako v původní aplikaci) a při překročení hranice sbaleny do seznamu, implementace tohoto je poměrně triviální.

- **Možnost pozastavení přehrávání segmentu** – přehrávání segmentu spuštěné tlačítkem u segmentu nelze pozastavit, stejně jako v původní aplikaci. Resp. lze pozastavit tlačítkem u přehrávače, ale po znovuspuštění se přehrávání nezastaví na konci segmentu. Z implementačního hlediska by po pozastavení bylo třeba získat aktuální pozici v přehrávání a vytvořit nový interval od dané pozice do konce segmentu, ve kterém by se v přehrávání pokračovalo. Tento interval by následně musel být resetován při přehrání jiného segmentu, přehrání celé stopy či manuálním posunutí pozice v přehrávání. Změnou tlačítka pro přehrání u právě přehrávaného segmentu na tlačítko pro pozastavení uživatel ztrácí možnost si segment opětovně přehrát od začátku bez přehrání do konce, tudíž pro tento účel by muselo být přidáno další separátní tlačítko. Implementace této funkce je tedy jistě složitější, než se může na první pohled zdát.

Chyby a nedodělky

- **Oprava chyb v systému pro vrácení/znovuprovedení změn** – kvůli množství navrátitelných operací je implementace poměrně komplexní a v určitých situacích může dojít k chybám. Při běžném použití k těmto chybám v drtivé většině případů nedojde, nicméně stále by bylo vhodné tyto chyby debugovat a opravit.

- **Oprava chyby v systému pro zabránění překryvu segmentů** – výchozí chování regionů na vizualizaci zvukové stopy dovoluje jejich vzájemné překrytí. Z tohoto důvodu byl zaveden jednoduchý algoritmus, který tomuto překryvu předchází. Tento algoritmus však nefunguje pro tvorbu segmentu, resp. algoritmus funguje dle očekávání, avšak potřebné změny délky segmentu se pro nový segment neprojeví. Pravděpodobně jde o problém s knihovnou, který bude třeba nějakým způsobem obejít, případně lze i zažádat o radu/opravu na githubu knihovny.

- **Propojení správy přepisů s API** – jelikož v novém SpokenData API nebyly k dispozici endpointy pro uzavření přepisu jako odmítnutý/dokončený, tyto funkce jsou dosud nefunkční. Implementace těchto funkcí je po zavedení potřebných endpointu jednoduchá.

Kapitola 7

Závěr

Cílem této práce byl návrh a implementace dobře použitelné webové aplikace pro efektivní a rychlou opravu automatického přepisu a anotaci rádiové VHF (very high frequency) komunikace pilot-věž.

Základní předlohou pro návrh byla aplikace anotační služby SpokenData od firmy ReplayWell, jejíž pracovní workflow jsem si nastudoval a prakticky vyzkoušel. Tím jsem získal přehled o dané problematice a nedostatcích prostředí, na jejichž eliminaci jsem se v rámci návrhu zaměřil. Inspirace pro návrh byla čerpána i z dalších zkoumaných anotačních aplikací.

Dále byly nastudovány relevantní principy návrhu dobře použitelného uživatelského prostředí a způsob testování použitelnosti s účastníky v rámci testovacích sezení, včetně způsobu hodnocení celkové použitelnosti účastníky po jejich sezeních.

Na základě srovnání vlastností moderních frameworků pro JavaScript a kaskádové styly byly vybrány ty nejvhodnější pro účely této práce. Následně byly vybrány knihovny třetích stran vhodné pro urychlení implementace navržených komponent.

Dle získaných znalostí byl proveden návrh funkcionality a drátěný model aplikace, jež se staly podkladem pro implementaci. Navrženou aplikaci se za použití zvolených technologií podařilo úspěšně implementovat v téměř plném rozsahu a propojit s novou verzí SpokenData API.

Aplikace byla otestována s uživateli dle nastudovaných postupů v rámci dvou kol o třech účastnících, čímž se podařilo ověřit použitelnost aplikace a získat řadu poznatků pro její další zlepšení. Výsledné použitelnostní skóre SUS z prvního kola testování je rovno 84,17 a z druhého kola 76,67, tato skóre jsou dle doporučené interpretace v rozsahu „přijatelné“ (70 a více). Aplikace byla také ve spolupráci s Ing. Josefem Žižkou nasazena na web SpokenData.

V budoucnu je třeba opravit zbylé problémy odhalené v rámci testování, implementovat správu přepisů uživatele a debugovat chyby implementace, zejména chyby v systému pohybu mezi kroky úprav přepisu. Vzhledem k nedostatkům aplikace není zcela připravena na nasazení do produkce, avšak má k tomu velmi blízko a po doladění detailů by se mohla stát nástupcem původní SpokenData aplikace. Jelikož byl o použití vyvinuté aplikace projeven zájem, moje práce na implementaci bude pravděpodobně po domluvě pokračovat.

Možná budoucí rozšíření zahrnují podporu anotace dvoukanálového zvuku, překryv značek sekcí textu, návod pro nové uživatele zabudovaný přímo do rozhraní a další.

Práce mi dala mnoho cenných znalostí a zkušeností s vývojem uživatelských prostředí, které zužitkuji zejména v profesním životě. Největším přínosem pro mě byla pravděpodobně zkušenost s Reactem a jeho knihovnami, který jsem si oblíbil a se kterým bych si nyní doká-

zal představit někdy pracovat i profesionálně. Vyvíjené prostředí bylo diametrálně odlišné od těch, na kterých jsem pracoval doposud, a podstatně jsem si tak rozšířil obzory.

Literatura

[1] Bassett, L. *Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON*. 1. vyd. O'Reilly Media, 2015. ISBN 1-4919-2945-6.

[2] Hurskiy, A. a Merenych, S. Front-end JavaScript Frameworks Showdown: Vue vs. React vs. Angular. *Clockwise Software* [online]. Září 2021 [cit. 2021-12-30]. Dostupné z: https://clockwise.software/blog/angular-vs-react-vs-vue/.

[3] Krug, S. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. 1. vyd. Pearson Education, 2009. Voices That Matter. ISBN 978-0-321-70284-5.

[4] Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3. vyd. New Riders Publishing, 2014. ISBN 0-321-96551-5.

[5] Rubanau, A. Bootstrap vs. Material-UI. Which One to Use for the Next Web App? *FlatLogic* [online]. Březen 2022 [cit. 2022-04-25]. Dostupné z: https://flatlogic.com/blog/bootstrap-vs-material-ui-which-one-to-use-for-the-next-web-app/.

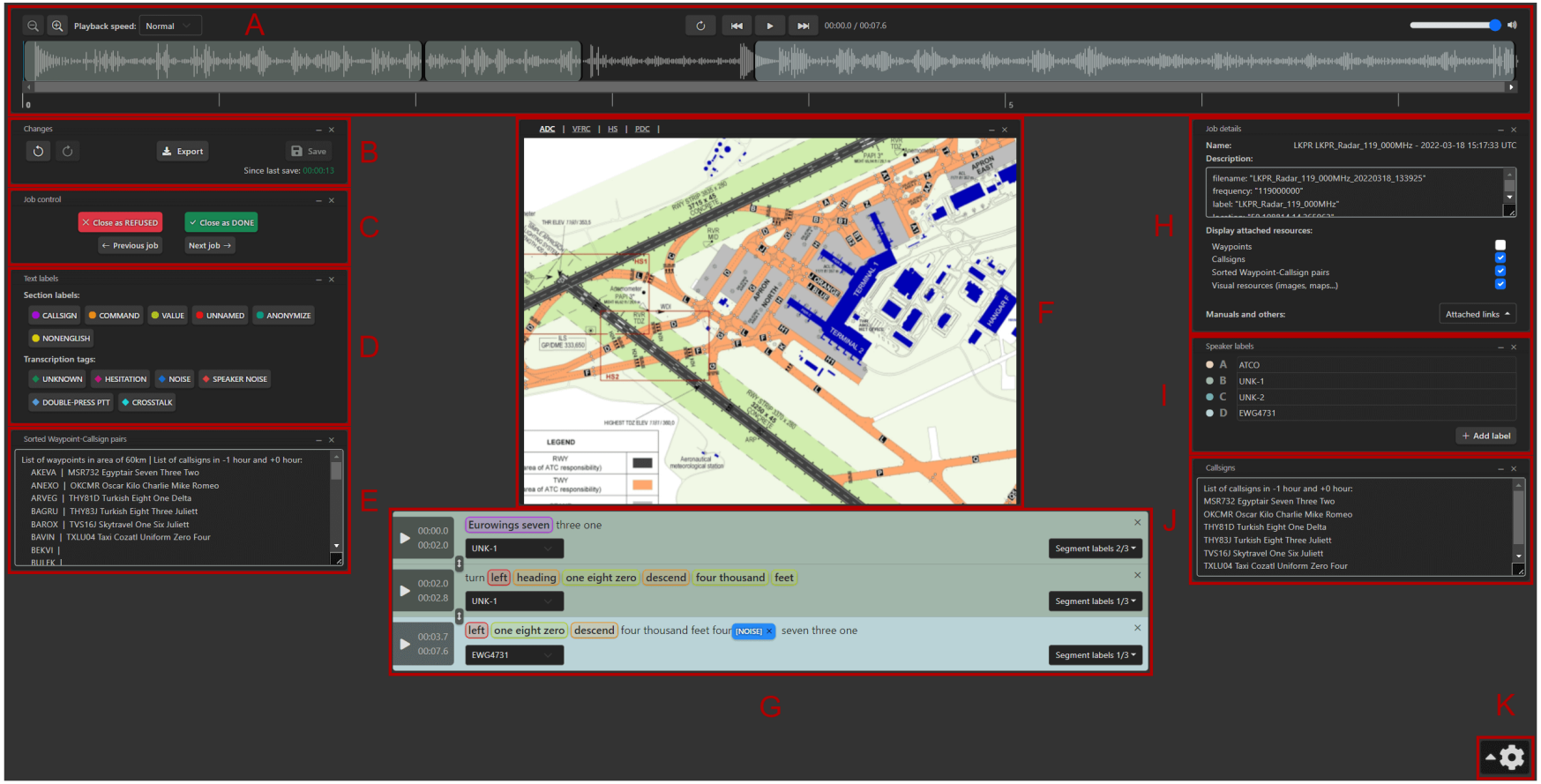
[6] Tidwell, J. *Designing Interfaces: Patterns for Effective Interaction Design*. 2. vyd. O'Reilly Media, 2005. ISBN 0-596-55517-2.

[7] Tullis, T. a Albert, W. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. 2. vyd. Amsterdam: Morgan Kaufmann, 2013. Morgan Kaufmann Series in Interactive Technologies. ISBN 978-0-12-415781-1.

Příloha A

Finální podoba implementovaného prostředí

Obrázek A.1: Implementované prostředí s otevřeným oknem nastavení: **A** – přehrávač, **B** – modul Změny, **C** – modul správy přepisů, **D** – textové značky, **E** – segmenty mluvčích, **F** – modul nastavení, **F1** – nastavení klávesových zkratek, **F2** – nastavení rozložení, **F3** – ostatní nastavení, **F4** – tlačítko pro otevření modulu nastavení, **G** – detaily přepisu, **H** – značky mluvčích



Obrázek A.2: Implementované prostředí s několika otevřenými přílohami: **A** – přehrávač, **B** – modul Změny, **C** – modul správy přepisů, **D** – textové značky, **E** – textová příloha (seřazené páry waypoint-callsign), **F** – obrázkové přílohy, **G** – segmenty mluvčích, **H** – detaily přepisu, **I** – značky mluvčích, **J** - textová příloha (callsigny), **K** – tlačítko pro otevření modulu nastavení

Příloha B

Testovací protokol

Úvodní instrukce

Dobrý den, moje jméno je Jan Plhal a budu vás provázet tímto testovacím sezením. Před tím, než začneme, bych vás seznámil s pár informacemi o průběhu testování, které budu číst, abych nic nevynechal.

Pracuji na webové aplikaci, kterou s vaší pomocí testuji, abych zjistil, jestli funguje tak, jak by měla. Testuji pouze aplikaci, ne vás samotného, takže se nemusíte bát, že uděláte nějakou chybu.

Chtěl bych vás požádat, abyste při používání aplikace co nejvíce přemýšlel nahlas: abyste říkal, na co se zrovna díváte, co se snažíte udělat a nad čím zrovna přemýšlíte. Nemusíte se obávat, že mě vaším názorem urazíte, cílem tohoto testování je zlepšení aplikace a chtěl bych slyšet vaše upřímné názory.

Pokud byste v průběhu testování měl nějaký dotaz, pravděpodobně jej nebudu moci zodpovědět, protože je potřeba, abyste pracoval samostatně, jako kdyby s vámi nikdo nebyl. Pokud však na konci testování budete ještě mít nějaké dotazy, rád je potom zodpovím.

S vaším svolením bude toto sezení nahráváno a nahrávka bude následně použita pouze mnou pro případné dohledání informací, které jsem si nestihl poznamenat. To je pro úvod všechno, máte nějaké dotazy?

Úkoly

- Otevřete si anotační manuál.
- Snižte rychlost přehrávání a opravte přepis segmentu.
- Označte segment.
- Označte sekci textu.
- Vložte značku mezi text.
- Vytvořte nový segment.
- Přidejte značku mluvčího a označte jím segment.
- Otevřete si přílohu.

- Zavřete modul a následně jej opět otevřete.
- Přenastavte si klávesovou zkratku.

Scénáře

- Před započetím práce si chcete nastudovat správný postup anotace. Najděte a otevřete si zdroj s návodem pro anotaci.
- Chcete opravit přepis druhého segmentu, ale mluvčí hovoří příliš rychle. Zpomalte si nahrávku a přepis segmentu následně opravte.
- Druhý segment se vám nyní zdá správně přepsaný. Označte náležitě celý segment.
- Po kontrole značek textu v třetím segmentu jste zjistil, že část textu „zero six zero“ je špatně označena a má být označena jako hodnota. Opravte danou značku.
- Po přehrání druhého segmentu jste zjistil, že mezi slovy „information“ a „VFR“ je v nahrávce zaváhání mluvčího. Označte tak dané místo.
- Po přehrání konce nahrávky jste zjistil, že bezprostředně za posledním segmentem má být ještě jeden segment s rozsahem až do konce nahrávky. Mluvčí tohoto segmentu je odlišný od těch existujících a má mít značku „UNK-3“. Přidejte daný segment a označte jej danou značkou.
- V průběhu anotace jste narazil na sekci textu, o které si myslíte, že je callsign, ale nejste si jistý. Otevřete si seznam callsignů a libovolně si jej napozicujte.
- Při práci jste se rozhodl, že modul „job control“ momentálně nepotřebujete a zavřel jste ho. Po chvíli tento modul opět potřebujete a chcete si jej otevřít. Zavřete daný modul a opět jej otevřete.
- V průběhu práce jste zjistil, že vám nevyhovuje klávesová zkratka pro uložení. Změňte klávesovou zkratku pro uložení na kombinaci „Ctrl/Cmd + Q“.

Příloha C

Obsah přiloženého média

Obsah přiloženého média je následující:

```
/
├── annotation-service
├── plakat.png
├── video.mp4
├── readme.txt
├── src-latex
└── technicka-zprava.pdf
```

Složka `annotation-service` obsahuje zdrojový kód aplikace, soubor `plakat.png` je plakát formátu A2 prezentující práci, soubor `video.mp4` je krátké video prezentující výsledky práce, soubor `readme.txt` obsahuje návod pro spuštění aplikace a další informace, složka `src-latex` obsahuje zdrojový kód technické zprávy práce a soubor `technicka-zprava.pdf` je přeložená technická zpráva ve formátu PDF.