Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

BAKALÁŘSKÁ PRÁCE

# Vývoj aplikace pro zaznamenávání výsledků logopedické diagnostiky

Vypracovala: Eva Holakovská
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

# JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
## Ekonomická fakulta

Akademický rok: 2018/2019

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Eva HOLAKOVSKÁ** |
| Osobní číslo: | **E17441** |
| Studijní program: | **B6209 Systémové inženýrství a informatika** |
| Studijní obor: | **Ekonomická informatika** |
| Téma práce: | **Vývoj aplikace pro zaznamenávání výsledků logopedické diagnostiky** |
| Zadávající katedra: | **Katedra aplikované matematiky a informatiky** |

## Zásady pro vypracování

Cílem bakalářské je vytvořit aplikaci pro zaznamenávání výsledků logopedické diagnostiky. Data získaná během vyšetření budou zapsána do formuláře a následně uložena do databáze. Výsledky bude možné graficky prezentovat a sledovat pokroky klientů.

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Závěry a doporučení.

| | |
|---|---|
| Rozsah pracovní zprávy: | **40 – 50 stran** |
| Rozsah grafických prací: | **dle potřeby** |
| Forma zpracování bakalářské práce: | **tištěná** |

Seznam doporučené literatury:


1. Albahari, J. & Albahari, B. (2017). *C# 7.0 in a Nutshell.* Sebastopol, California (USA): O'Reilly Media.
2. Harrington, J. L. (2016). *Relational Database Design and Implementation.* Cambridge, MA, USA: Morgan Kaufmann.
3. Nagel, C. (2016). *Professional C# 6 and .NET Core 1.0.* Indianapolis, Indiana (USA): John Wiley & Sons.
4. Posadas, M. (2016). *Mastering C# and .NET Framework.* Birmingham, UK: Packt.
5. Price, M. J. (2017). *C# 7.1 and .NET Core 2.0: Modern Cross-Platform Development.* Birmingham, UK: Packt Publishing.


Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **15. ledna 2019**
Termín odevzdání bakalářské práce: **12. dubna 2020**

V Českých Budějovicích dne 15. března 2019

L.S.

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

**doc. Dr. Ing. Dagmar Škodová Parmová**
děkanka

**doc. RNDr. Jana Klicnarová, Ph.D.**
vedoucí katedry

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách, v platném znění, souhlasím se zveřejněním své bakalářské práce, a to v – nezkrácené podobě / v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb., o vysokých školách, v platném znění, zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Pacově dne 10. 4. 2021

……

Eva Holakovská

# Poděkování

Na tomto místě bych ráda věnovala poděkování vedoucímu mé bakalářské práce Mgr. Radimu Remešovi, který mi poskytl cenné rady a podporu během celého procesu tvorby této práce. Dále bych ráda poděkovala především své rodině, která mě během studií plně podporovala, a to i v dobách, kdy mě studijní povinnosti odvedly za hranice České republiky. V neposlední řadě děkuji svým přátelům, kteří mi jsou velkou oporou. Jmenovitě chci zmínit Violu Jónovou, která dala celé mé bakalářské práci úžasný přesah a Ladislavu Mackovou, která při mně vždy stála během společných studií na naší univerzitě.

# Obsah

1 Úvod a cíle práce ..... 11
2 Metodika ..... 12
3 Životní cyklus softwaru ..... 13
4 Prvotní plán a záměr ..... 15
4.1 Individuální aplikační software ..... 15
4.2 Typový aplikační software ..... 15
5 Analýza, definice a specifikace požadavků ..... 16
6 Návrh systému ..... 18
6.1 Návrh logické části aplikace ..... 18
6.2 Návrh databáze ..... 19
6.2.1 Normální formy ..... 20
7 Implementace kódu ..... 22
7.1 Framework .NET ..... 22
7.2 Windows Presentation Foundation ..... 22
7.3 XAML ..... 23
7.4 MVVM ..... 23
7.4.1 Model ..... 24
7.4.2 View Model ..... 24
7.4.3 View ..... 24
8 Uvedení aplikace do provozu ..... 25
8.1 Strategie přechodu ..... 25
9 Provoz aplikace ..... 27
10 Plánování a analýza požadavků ..... 28
11 Definice a specifikace požadavků ..... 29
11.1 Přijetí klienta do praxe ..... 29
11.1.1 Základní informace o klientovi ..... 29

11.1.2 Rodinná anamnéza ..... 29
11.1.3 Osobní anamnéza ..... 30
11.1.4 Zdravotní anamnéza ..... 30
11.1.5 Artikulace ..... 31
11.2 Diagnostická vyšetření ..... 31
11.2.1 Foneticko-fonologická rovina ..... 33
11.2.2 Lexikálně-sémantická rovina ..... 34
11.2.3 Morfologicko-syntaktická rovina ..... 35
11.2.4 Pragmatická rovina ..... 36
11.2.5 Motorika ..... 37
11.2.6 Ostatní ..... 38
12 Návrh aplikace ANDI ..... 40
12.1 Návrh uživatelského rozhraní ..... 40
12.2 Datový model ..... 41
12.2.1 Klient ..... 41
12.2.2 Rodinná anamnéza ..... 42
12.2.3 Osobní anamnéza ..... 42
12.2.4 Zdravotní anamnéza ..... 42
12.2.5 Artikulace ..... 42
12.2.6 Terapie ..... 42
12.2.7 Foneticko-fonologická rovina ..... 43
12.2.8 Lexikálně-sémantická rovina ..... 43
12.2.9 Morfologicko-syntaktická rovina ..... 43
12.2.10 Pragmatická rovina ..... 43
12.2.11 Motorika ..... 43
12.2.12 Ostatní ..... 43
13 Implementace aplikace ANDI ..... 44

13.1 Logo ..... 44
13.2 Uživatelské prostředí ..... 44
13.3 Vrstva ViewModel ..... 46
13.3.1 INotifyPropertyChanged ..... 47
13.3.2 ICommand ..... 47
13.4 Modely ..... 48
13.5 Úvodní obrazovka ..... 48
13.5.1 MainWindow ..... 48
13.5.2 MainVM ..... 50
13.6 Nový klient ..... 50
13.6.1 NewKlientControl ..... 50
13.6.2 NewKlientVM ..... 50
13.7 Osobní informace ..... 51
13.7.1 OsobniInfoControl ..... 51
13.7.2 OsobniInfoVM ..... 52
13.8 Terapie ..... 53
13.8.1 TerapieControl ..... 54
13.8.2 TerapieVM ..... 54
13.9 Grafické zobrazení výsledků terapie ..... 54
13.9.1 GrafControl ..... 55
13.9.2 GrafVM ..... 55
13.10 Pomocné třídy ..... 56
13.10.1 SelectedKlientSingleton ..... 56
13.10.2 SqliteDataAccess ..... 57
13.10.3 DataAccessHelpers ..... 57
14 Uvedení aplikace ANDI do provozu ..... 59
15 Závěr ..... 60

I. Summary and Key Words ..... 61

II. Seznam použitých zdrojů……..…….……..62

III. Seznam obrázků ..... 63

IV. Seznam ukázek kódu ..... 64

V. Seznam příloh ..... 65

VI. Přílohy..... 66

# 1 Úvod a cíle práce

Použití mluveného slova je jednou z nejdůležitějších lidských schopností. Slova nám dávají velkou moc, můžou podpořit, potěšit, pomoci, ale i ublížit a ranit. Logopedie je relativně mladý komplexní vědní obor, který se zabývá problémy v komunikaci. Současná logopedie se zabývá komunikačními schopnostmi, jejich narušením a následnou diagnostikou a terapií. Logopedie pomáhá odstraňovat problémy s komunikací všem věkovým kategoriím.

Každý logoped si průběhu své praxe vyvine vlastní systémy a postupy pro vedení terapie, diagnostiku klienta a záznam zjištěných informací. Tyto se liší nejen na základě osobních preferencí terapeuta, ale i oblasti logopedie a typu klientů.

Cílem této práce je zdokumentovat vývoj aplikace podle potřeb praxe Mgr. Violy Jónové, která funguje jako sdílený logoped pro několik různých zařízení a také se věnuje speciální logopedii v hiporehabilitaci v pedagogické a sociální praxi. Logopedie v hiporehabilitaci v pedagogické a sociální praxi je unikátní projekt střediska doporučené hiporehabilitace Horticon. Během této terapie dochází k zapojení mnoha faktorů a klientovi je k dispozici kromě logopedky i instruktor pro aktivity s využitím koní a asistent. Samotné prostředí statku a stájí, společně s kontaktem se zvířaty působí velice stimulativně.

Práce se soustředí nejen na vytvoření kódu, ale i procesy prvotního plánování, analýzy, definice a specifikace jednotlivých požadavků, návrhy dílčích částí a uvedení do praxe. Díky detailnímu rozboru jednotlivých procesů a postupů, může tato práce také sloužit případným budoucím uživatelům mimo prvotní logopedickou praxi.

Aplikace bude sloužit k uchovávání základních informací o klientech, jejich anamnézách a záznamech jednotlivých diagnostických vyšetření v rámci praxe logopeda. Z důvodu mladosti oboru i velké variability mezi samotnými logopedy a jejich postupy, je nabídka softwaru pro tyto účely malá. Díky tomu zde vzniká prostor pro tvorbu využitelného softwaru.

# 2 Metodika

Již při výběru tématu bakalářské práce jsem provedla zevrubný průzkum na trhu softwarových řešení pro logopedickou praxi. Oslovení logopedi, kteří zaznamenávali výsledky svých terapií pomocí speciálních softwarových řešení využívali části aplikačních řešení zpravidla v zdravotnických systémech. Prostor vyhrazený na logopedii byl ve většině případů pouze malou částí klientovy anamnézy. Ostatní oslovení využívali především vlastní hodnotící postupy a styly záznamu výsledků terapií. Častým řešením je používání vlastních tištěných šablon a jejich manuální vyplňování. Někteří oslovení také využívají textových editorů pro záznamy průběhu terapie a plné slovní hodnocení klientů. Tento průzkum mě utvrdil ve smysluplnosti vývoje aplikace pro použití v logopedické praxi.

Po výběru a ujasnění tématu práce jsem vycházela ze studia a analýzy odborné literatury. Znalosti jsem čerpala nejen z oblasti literatury o informačních systémech a vývoji počítačových aplikací, ale i literatury na téma logopedie jako vědy. Další nesmírně cennou studnicí informací byla sezení s Mgr. Violou Jónovou, během kterých jsem získala potřebné informace pro analýzu a návrh celého projektu.

Pro samotnou aplikaci jsem zvolila programovací jazyk C# a platformu .Net Framework. Program jsem napsala v aplikaci Visual Studio 2019 od firmy Microsoft. Grafické rozhraní jsem vytvořila pomocí knihovny WPF (Windows Presentation Foundation) opět od společnosti Microsoft s použitím značkovacího jazyka XAML. Pro tvorbu databáze využívám systému SQLite. Spojení mezi aplikací a řídící bází dat je zajištěno pomocí open source softwaru Dapper. Pro tvorbu grafů znázorňujících výsledky terapií je použita knihovna LiveCharts.

# 3 Životní cyklus softwaru

Vývoj životního cyklu softwaru (Software Development Life Cycle) je proces, který je využíván především na poli výpočetní a informační techniky pro produkci kvalitního softwaru, který bude uspokojovat zákazníkovi potřeby za určité množství zdrojů a času. (Khare, 27-Jun-2019)

Proces tvorby softwaru probíhá v normálních podmínkách za spolupráce tří hlavních stran:

- Zákazník – má na zavedení softwaru svůj zájem a financuje jeho vývoj;
- Dodavatel – tvůrce systému splňujícího zadané požadavky;
- Koncový uživatel – uživatel, který systém bude reálně používat.

Někteří koncoví uživatelé nechtějí zavedení nového systému z různých důvodů (nechuť se učit pracovat v novém systému, strach o práci, ...) dopustit. V tomto případě může dojít ke zpomalení vývoje například přidáváním nesmyslných požadavků či neobjektivní kritikou dodaného řešení. To může vést ke zpomalování projektu v krajním případě až k jeho neúspěchu. (Křena & Kočí, 2006)

Při vývoji také jednoznačně záleží na použitém modelu SDLC (Systems development life cycle). Každý z modelů představuje vlastní verzi procesu. Počet, pořadí, popřípadě opakování jednotlivých kroků či fází se model od modelu liší. Dva základní póly spektra modelů reprezentují vodopádový model a agilní model.

Vodopádový model popisuje vývoj života softwaru jako kaskádovitý sled jednotlivých fází. Pokud projekt přejde z jedné fáze do druhé již není možno se vracet a předcházející úkony či plány měnit. Veškeré kroky jsou pečlivě plánované a detailně dokumentované. Při vývoji se postupuje jasným směrem s předem stanovenými milníky. Výhodami jsou jednoduchost, srozumitelnost a důkladná dokumentace. Naopak nevýhodami je především nepřizpůsobivost a robustnost. Tento model může být vhodný pro malé, krátkodobé projekty. Nicméně v dnešní době, se zaměřením na rychlost a změnu, je model pro rozsáhlé projekty takřka nepoužitelný.

Na druhé straně spektra máme agilní modely. O agilním přístupu a modelech jsou napsány celé knihy. Pro účely této bakalářské práce pouze zmíním nejnutnější informace a rozdíly. Matt LeMay shrnuje odlišnosti agilního přístupu do čtyřech bodů:

- Jednotlivci a interakce převažují nad procesy a nástroji.
- Fungující software převažuje obsáhlou dokumentaci.
- Spolupráce se zákazníkem převažuje smluvní vyjednávání.
- Schopnost přizpůsobit se změně převažuje dodržování plánů. (LeMay, 2018)

Díky tomuto jsou agilní přístupy mnohem flexibilnější a v současné době se aktivně používají.

Během životního cyklu softwaru je potřeba sestavit detailní plán, specifikovat požadavky klienta, určit potřebné množství zdrojů a času, vytvořit návrhy (uživatelského prostředí, architektury systému, databáze atd.), implementovat kód, uvést systém do provozu, udržovat systém v provozu a ukončit činnost systému. Během ukončování činnosti většinou dochází k přestupu na nový systém a tento zase prochází celý cyklus od začátku.

# 4 Prvotní plán a záměr

První fází je sestavení základního plánu a určení záměru. Je potřeba ujasnit rozsah, hloubku a samotný obsah řešení. Tento krok můžeme také nazývat úvodní studií nebo studií proveditelnosti. Poté, co jsme definovali koncept (jaké funkce bude aplikace mít, kdo budou koncoví uživatelé, jaké budou hlavní rozdíly mezi novým a současným systémem atd.) je na čase vyhradit na projekt odpovídající množství personálních, finančních a časových zdrojů. Je sestaven řešitelský tým, který převezme zodpovědnost za získání daného řešení. V této prvotní fázi je potřeba zvážit možnosti naplnění konceptu s co nejmenšími náklady. Jedním z hlavních rozhodnutí je způsob získání nového softwaru. V tomto bodě se nám nabízí dvě základní alternativy.

## 4.1 Individuální aplikační software

Aplikace vytvořená čistě na míru dle požadavků zadavatele projektu. Funkcionality jsou nastavené tak, aby přesně odpovídaly procesům, pro které jsou určeny. Nevýhodou tohoto přístupu je finanční i časová náročnost řešení. Výhodou je podpora specifických procesů a tím i podpora dosažení specifických cílů. Při pohledu na výhody a nevýhody je jasné, že tento nákladný individuální přístup není vhodný pro podporu pouze běžných standardizovaných procesů (např. e-mail, účetnictví). (Voříšek, 2015)

## 4.2 Typový aplikační software

Typový aplikační software je postaven na vývoji pro určitý typ podniku nebo odvětví. Aplikace splňuje obecné požadavky pro určitý model klientů. V některých případech je pak možno udělat drobné úpravy na přání klienta (napojení na jiné systémy, personalizované nastavení parametrů atd.) Výhodou pro klienta je nižší pořizovací cena a časová náročnost na zavedení. Celkové náklady na vývoj tohoto typu softwaru jsou sice vyšší, ale rozpustí se mezi více odběratelů. Doba nasazení je kratší, jelikož se do podniku zavádí již hotový produkt. Nevýhodou tohoto řešení je, že podnikový proces se musí přizpůsobit struktuře a možnostem programu. Na druhé straně tyto aplikace jsou většinou vyvíjeny ve spolupráci se špičkami v odvětví, takže méně rozvinuté podniky mohou získat ověřené postupy a část know-how lídrů trhu. (Voříšek, 2015)

# 5 Analýza, definice a specifikace požadavků

Pokud rozhodnutí padne na vývoj individuální aplikace, je potřeba, aby se řešitelské týmy stran dodavatele i odběratele pustily do detailních analýz požadavků. Dodavatelé se musí seznámit s cíli, kulturou a globální strategií zadavatele. Dalším vhodným krokem je analyzovat aplikace ostatních subjektů na stejném trhu, a to jak konkurentů, tak obchodních partnerů. V této fázi projektu je potřeba se již zaměřit na detailní dokumentaci a analýzu procesů, které má nový software podporovat.[1]

Zadavatelé by měli do této fáze vstupovat s vypracovanými seznamy požadavků. Pro definování požadavků nejsou dané pevné standardy. Je tedy nutné zajistit kvalitní komunikaci mezi týmy. Funkční požadavky popisují, co by měl systém dělat. Obecně je lze vyjádřit pomocí vět Systém umí (funkce)… Systém bude (funkce)… Nefunkční požadavky popisují vlastnosti systému, jako celek. Mezi nefunkční požadavky se můžou řadit nároky na bezpečnost, splňování legislativních rámců, uživatelskou přívětivost atd.

V průběhu analýzy se mohou požadavky měnit, jak zásahy ze stran řešitelského týmu dodavatele, tak zadavatele. Cílem je vytvořit dokument obsahující kompletní a konzistentní přehled požadavků, které odpovídají reálným potřebám.

Pokud již zadavatel informační systém používá, je také potřeba ho plně zanalyzovat a určit jeho výhody a nevýhody. Analýza může pomoci odhalit nadbytečné části (nepoužívané formuláře, data atd.) a naopak poukázat na úseky procesu, které nemají v systému oporu.

Analýza současného systému může také přinést cenné informace nejen o podporovaných procesech, ale i koncových uživatelích systému. Pokud máme identifikované procesy a osoby s nimi spjaté, můžeme používat informační systém pro měření a zvyšování výkonosti. Pro tento krok je klíčové definování vhodných metrik, stanovení výchozích a limitních hodnot. Data by měla být v průběžně předávána řídícím pracovníkům. Využití těchto měření může být zahrnuto do motivačních parametrů pracovníků. (Basl, J., & Blažíček, R. 2012)

[1] Tým pracující na straně dodavatele by měl získat obecný přehled v dané problematice. To napomůže pochopení daných procesů a předejde možným fatálním nerozuměním a chybám z nich vyplynulých.

Po pečlivé analýze a dokumentaci procesů, databází a aplikace jako takové je možné přejít do fáze návrhu.

# 6 Návrh systému

Dokumenty vzniklé během předchozí fáze analýz jsou základem pro tvoření návrhu. Návrh řešení aplikace detailně popisuje funkce aplikace, používaná data a procesy, které aplikace bude podporovat. Pokud během analýzy procesů byly odhaleny problémy a nedostatky, je nutné je opravit a upravit procesy do co nejjednodušší a efektivnější roviny.

Úpravy používaných procesů nepřináší pouze změny v oblasti aplikace jako takové, ale především ovlivňují práci koncových uživatelů. Příprava uživatelů na tyto změny je klíčová pro úspěch aplikace. Lidé se často nechtějí vzdávat svých zajetých postupů a režimů. Pokud není kooperace a komunikace klíčových uživatelů s dodavatelským týmem dostatečná, může zde dojít k bojkotu a neochotě nový systém používat.

Po ujednání návrhů na změny, popřípadě úpravy podporovaných procesů, je potřeba se zaměřit na návrhy bází používaných dat a koncového aplikačního řešení. Tyto návrhy se týkají samotné logiky aplikace, ale i technologických nároků.

## 6.1 Návrh logické části aplikace

První z fází návrhu logické části aplikace je zpracování požadavků vyjednaných ve fázi analýz. Funkce a funkcionality získávají strukturovanou formu zahrnující všechny důležité atributy, (tj. určení obsahu, definice vstupních a výstupních dat, postupy výpočtů, zahrnutí požadovaných legislativ do jednotlivých kroků…) Důležité je také připravit návrhy interních vazeb v programu samotném a externích vazeb na ostatní software, databáze a technologie. (Voříšek, 2015)

Dalším krokem je návrh vizuální strany aplikace. Příprava uživatelského rozhraní je klíčová. Koncoví uživatelé musí být schopni aplikaci plně využívat, ale jejich připomínky mohou být zapříčiněné pouze prvotní neznalostí a nezvykem. Komunikace mezi uživateli a vývojáři je opět klíčovým bodem. Je třeba zohlednit zařízení, na kterých program poběží a tomu přizpůsobit uživatelské prostření. Návrhy jednotlivých obrazovek pomohou při vizualizaci celého projektu. Dále je třeba připravit návrhy výstupních informací, jako jsou tištěné formuláře, standardizované texty, tiskové sestavy, výkazy atd.

## 6.2 Návrh databáze

Databáze je základním kamenem celého systému. Jakékoliv chyby, nekonzistence či nepřesnosti v návrhu se můžou přenést do struktury databáze a napáchat velké škody. Z tohoto důvodu se na správný návrh zaměřím detailněji.

Relační databáze jsou nejlepší volbou pro data, která jsou jasně strukturovaná a vztahy mezi nimi jsou dané. Proto nám relační datový model umožní jejich optimální uchování a následnou práci. Relační databáze je taková databáze, jejíž logická struktura je tvořena jen a pouze kolekcí relací.

Relační datový model představil světu v roce 1970 E. F. Codd a způsobil naprostou revoluci v tvorbě a struktuře databází. K popsání relačního modelu použil matematickou definici pojmu relace. (Máme-li množiny S1, S2… Sn (které nemusí být odlišné) pak R je relací na těchto množinách, pokud je podmnožinou kartézského součinu S1 × S2 ×… ×Sn) Pro zjednodušenou představu Codd ukazuje relaci jako pole uspořádaných relací n-tého stupně, splňujících následujících pět pravidel.

1. Každý řádek reprezentuje n-tici z R
2. Na pořadí řádků nezáleží
3. Všechny řádky se liší
4. Pořadí sloupců je významné a odpovídá pořadí S1, S2… Sn na kterých R definujeme
5. Význam sloupce je z části vyjádřen jeho pojmenováním

(Codd, 1970)

Dále jsou představeny pojmy primární klíč a cizí klíč. Primární klíč je unikátní sloupec nebo kombinace více sloupců sloužící pro jednoznačnou identifikaci každé jednotlivé řádky. Můžeme mít několik vhodných voleb pro primární klíč (tyto možnosti jsou tzv. kandidátní klíče), nicméně vždy může být zvolena pouze jedna, a ta se následně primárním klíčem nazývá. (Codd, 1970)

Pokud primární klíč jedné tabulky použijeme pro vytvoření reference (odkazu) v tabulce jiné, pak tento sloupec nazýváme cizím klíčem. Cizí klíče mohou na rozdíl od primárního klíče nabývat i hodnoty null. Nicméně musíme zajistit, aby byla dodržena referenční integrita. Pokud má cizí klíč nenulovou hodnotu, musí se shodovat s existujícím primárním klíčem v původní tabulce. Současné systémy řízení báze dat mají vyvinuté nástroje,

které odhalí případné narušení referenční integrity a nedovolí danou modifikaci dat. (Harrington, 22nd April 2016)

Další typ dělení klíčů je na klíče přirozené a umělé. Umělý klíč je číslo, které slouží čistě jako klíč a nemá žádný reálný význam. Oproti tomu přirozený klíč tvoří data, která by stejně byla součástí databáze a mají reálný odraz ve světě. Ačkoliv se může zdát použití přirozených klíčů lákavé, jsou zde rizika. Například větší riziko změny daného atributu a tím ohrožení integrity databáze, (např. používání názvu firmy nebo e-mailové adresy) nebo nedostupnost daných dat (např. použití českého rodného čísla způsobí komplikace při snaze zanést do databáze cizince…). Je potřeba mít na paměti, že reálný svět je vysoce dynamické prostředí a udržení konzistentního přepisu reálných skutečností do klíče databáze může přinést mnohonásobně více práce a problémů než užitku.

### 6.2.1 Normální formy

Kvalitu návrhu databáze podpoří dodržování šesti základnách normálních forem. Tyto formy mají kaskádovitý charakter a splnění vyšší formy je podmíněno naplněním požadavků všech forem předchozích.

*První normální forma* je splněna, pokud jsou data zaznamenána ve dvoudimenzionální tabulce a žádná z buněk neobsahuje opakující se skupiny dat. Jinými slovy: data v tabulce jsou atomická. *Druhá normální forma* říká, že každý neklíčový atribut je závislý na celém primárním klíči. *Třetí normální forma* vyžaduje, aby databáze neobsahovala žádné transitivní vztahy. To jsou vztahy typu atribut B je závislý na A a atribut C je závislý na B, tímto mezi atributem C a A vzniká nežádoucí tranzitivní vztah.

Tyto první tři normální normy bývají často dostačující pro zajištění kvalitního modelu. Boyceho–Coddova normální forma, čtvrtá normální forma a pátá normální forma se vztahují k více specifickým problémům, a to především u složených klíčů. Pokud relace nemá tato specifika a splňuje třetí normální formu, pak můžeme říct, že splňuje i pátou normální formu.

*Boyceho–Coddova normální forma* vyžaduje, aby všechny determinanty byly kandidátními klíči. (V případě, že existuje více kandidátních klíčů, přičemž všechny jsou tvořeny více než jedním sloupcem a mají jeden sloupec společný.) *Čtvrtá normální forma* zabraňuje vícehodnotovým závislostem. (Ty vznikají tím, že atribut A nabude určité

hodnoty a pak nezávislé atributy B a C nabývají pouze určitých omezených hodnot.) (Harrington, 22nd April 2016)

*Pátá normální forma* se zabývá situací, kdy složený primární klíč obsahuje párovou cyklickou závislost a tím tabulka opět obsahuje zbytečně opakující se hodnoty. Aby byla pátá normální forma splněna, musí být tabulky rozděleny tak, aby párové cyklické závislosti neobsahovaly a zároveň abychom po složení těchto dekomponovaných tabulek získali tabulku původní (nedošlo tedy ke ztrátě či příbytku dat). (Harrington, 22nd April 2016)

# 7 Implementace kódu

## 7.1 Framework .NET

Před příchodem .NET Frameworku se celý ekosystém programů okolo společnosti Microsoft pohyboval kolem několika málo základních jazyků např. Visual Basic a C++. Do chvíle nástupu .NET a změn, které přinášel se využívalo především modelu komponent zvaného COM (Component Object Model). Tento model měl své nedostatky, a proto v roce 1996 Microsoft vytvořil expertní tým softwarových inženýrů. Tento tým měl pod projektem Next Generation Windows Services (NGWS) vytvořit novou platformu oproštěnou od chyb a limitů COM. První výsledky snažení tohoto týmu byly do světa vypuštěny v roce 2000. Dne 13. února 2002 byla spuštěna první verze .NET framework. Od této chvíle jde tvorba .NET Framework ruku v ruce s novými verzemi vývojového prostředí Visual Studio. Tyto nástroje byly a jsou klíčové pro vývoj aplikací běžících na operačním systému Windows od společnosti Microsoft. (Posadas, 2016)

## 7.2 Windows Presentation Foundation

Windows Presentation Foundation (WPF) je knihovna tříd uživatelského rozhraní pro tvorbu aplikací pro systém Windows. WPF nabízí nástroje pro mnohem dynamičtější uživatelské rozhraní, než tomu bylo u jeho předchůdce Windows Forms (WinForms). Hlavní výhody WPF jsou:

1. Flexibilní rozložení objektů na obrazovce, bez nutnosti přesného umístění pomocí specifických souřadnic. Uživatelské rozhraní se může mnohem jednodušeji přizpůsobovat změnám velikosti okna, dynamickému obsahu či jazykovým verzím.
2. Možnost využívat bohatého množství písem a stylů. WPF přináší spoustu možností a vylepšení pro práci s textem. Delší texty mohou využívat stylizování do sloupců, zalamování textu či další pokročilé funkce pro zobrazování dokumentů, pro lepší přehlednost a navýšení uživatelského komfortu.
3. Podpora video a audio médií v aplikaci. WPF podporuje použití audiovizuálních médií, a to i několika současně. Aplikace také může být mnohem jednodušeji animovaná.
4. Příkazy (Commands) umožňují přístup k jedné funkcionalitě z vícero různých ovládacích prvků.

5. Styly a šablony jsou velikým pomocníkem při tvorbě uživatelského rozhraní. Používání stylů a šablon zajistí dodržení formátovacího standardu napříč celou aplikací. Změna stylu je pak jednoduchou úpravou a promítne se do všech prvků k němu přiřazených. (MacDonald, 2013)

## 7.3 XAML

Extensible Application Markup Language (XAML) je značkovací jazyk sloužící pro tvorbu instancí .NET objektů. Primární použití XAML v WPF projektech je tvorba uživatelského rozhraní. Soubory XAML definují podobu a umístění prvků (panely, tlačítka, textboxy…) na uživatelské obrazovce. Kód psaný v jazyce XAML se zpravidla nepíše ručně, ale využívá se nástrojů (Microsoft Visual Studio, Microsoft Expression Blend), které potřebný kód z části vygenerují. Při přidání XAML souboru do projektu dojde automaticky k vytvoření zdrojového kódu (tzv. code behind). Tento kód slouží k inicializaci všech popsaných objektů. Vývojář může tento kód rozšířit o vlastní metody.[2] Představení XAML jazyka velice usnadnilo kooperaci mezi návrháři uživatelského prostření a vývojáři samotné logiky aplikace, protože oba projekty mohou vznikat nezávisle na sobě.

## 7.4 MVVM

Model-View-View Model (MVVM) je vzor softwarové architektury představený Johnem Gossmanem v roce 2005 a v dnešní době je běžně využíván vývojáři WPF aplikací. Hlavní myšlenkou tohoto modelu je striktní rozdělení uživatelského rozhraní (User Interface) a aplikační logiky (business logic). Toho je dosaženo rozdělením aplikace do tří odlišných typů komponentů: Models, Views a Viewmodels. Vrstva Viewmodels se nachází mezi vrstvami Views a Models a zajišťuje mezi nimi oboustrannou komunikaci. Aby byl MVVM vzor dodržen, mezi komponenty Views a Models by se nemělo vyskytovat žádné přímé spojení. (Yuen, 2020)

Následování vzoru MVVM přináší mnoho výhod plynoucích z oddělení logické vrstvy, datové vrstvy i uživatelského rozhraní. V případě potřeby lze měnit uživatelské rozhraní nebo naopak využívat aplikační logiku v jiných projektech. Můžeme také jednoduše nahradit datovou vrstvu falešnou databází pro testovací účely, anebo veškeré reálné datové toky odříznout.

[2] Doplňování vlastních metod do code behind se zásadně nedoporučuje při využití architektury MVVM, jelikož dochází k přenesení části logiky aplikace do souborů obsahujících uživatelské rozhraní.

### 7.4.1 Model

Model je vrstva, která drží veškerá data. Datový model se zpravidla skládá z tříd, které kopírují strukturu příslušných tabulek v databázi. Veškeré komponenty v této vrstvě by měly být plně odděleny od uživatelského rozhraní.

### 7.4.2 View Model

Každý View Model poskytuje přiřazenému komponentu z vrstvy View veškerá potřebná data a funkcionality. Tato vrstva přebírá veškerá potřebná data z vrstvy Model a často dochází k úpravě formy dat z důvodu lepšího zobrazení v uživatelském rozhraní. View má se svým View-modelem obousměrné spojení, a to je založené především na data-binding a implementování INotifyPropertyChanged rozhraní.

### 7.4.3 View

View definuje rozložení a vzhled uživatelského prostředí. Aktivně komunikuje s vrstvou View-model, přes kterou dochází k získání potřebných dat a funkcionalit z nejhlubší vrstvy Model. Jedině tato vrstva přichází do přímého kontaktu s uživatelem aplikace.

# 8 Uvedení aplikace do provozu

Před samotným zavedením aplikace do provozu je potřeba udělat přípravné práce. Plán migrace a postupu zavedení do provozu zahrnuje specifikaci a časový harmonogram pro samotný přesun na nový software (popřípadě přesun z analogového procesu na digitální) a migraci dat s tím spojenou. Tento plán zahrnuje jednotlivé kroky, termíny a strategii migrace.

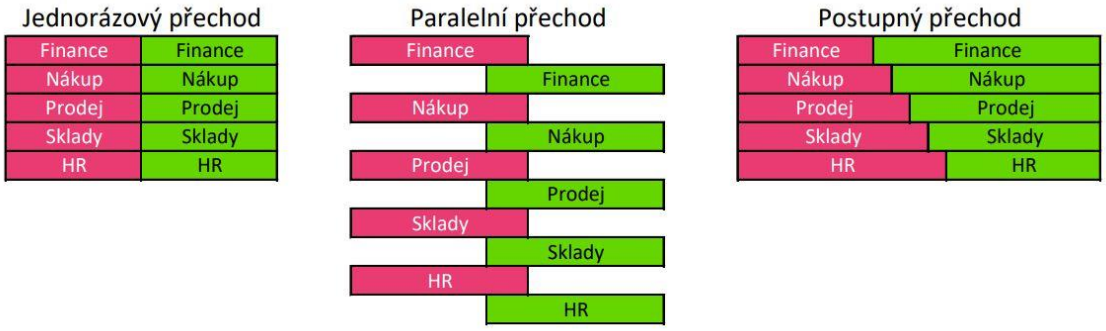
## 8.1 Strategie přechodu

Jednorázový přechod je strategie, při které dojde k ukončení veškerých funkcí starého softwaru a okamžitého přesunu na software nový.

Při postupném přechodu se nahrazuje používání starého systému u jednotlivých úloh systémem novým. Například jako první migruje do nového systému účetnictví. Po úspěšném přesunu se nový systém začne používat i pro správu skladů atd. Proces migrace končí ve chvíli, kdy nový systém podporuje všechny procesy.

Paralelním přechodem nazveme situaci, kdy po určitou dobu běží oba systémy současně a poté se starý software odstaví.

*Obrázek 1 Integrace IS*




Zdroj: Vlastní zpracování (2021)

Při volbě strategie je vždy potřeba zvážit rizika a vhodnost pro konkrétní situaci. Při postupném a paralelním přechodu zatěžujeme uživatele používáním obou systémů současně. Na druhou stranu při jednorázovém přesunu riskujeme neočekávané technické výpadky na počátku nového provozu a zvýšené riziko lidské chyby z neznalosti.

Před uvedením do provozu je nutné zkontrolovat technologickou infrastrukturu (servery, klientské stanice atd.) a provést nutné upgrady a instalace. Dále musíme zajistit migraci dat, a to buďto konverzí původních databází nebo manuální tvorbou databází nových. V neposlední řadě musíme připravit samotné koncové uživatele. Ti by měli být řádně proškoleni a seznámeni s organizačními změnami, které nový systém přináší.

Fáze uvedení aplikace do provozu je zpravidla ukončena předávacím řízením. Během úspěšného přesunu na dodaný software se potvrdí, že došlo k naplnění požadovaných funkcionalit a provozních charakteristik aplikace. Formálním ukončením této fáze bývá podepsání akceptačních protokolů.

# 9 Provoz aplikace

Během provozu aplikace je potřeba monitorovat její výkony a řešit výpadky a poruchy. Monitorování je zaměřeno na sledování vytížení, charakter chyb a poruch a způsob jejich nápravy. Výsledkem monitorování jsou provozní statistiky, které jsou důležitým zdrojem pro plánování úprav, změn a dalšího rozvoje aplikace. Zvláštní význam má péče o databáze (přístupová práva, zálohování, archivace).

Dále je během provozu nutné poskytovat podpůrný servis a konzultační služby uživatelům aplikace. Kontaktní místo pro uživatele bývá zpravidla nazýváno jako helpdesk (v rozšířenějším pojetí service desk). Zde může uživatel řešit problémy s aplikací, získat odpovědi na svoje otázky i nahlásit požadavky, a to zpravidla skrze variaci komunikačních kanálů (telefon, e-mail, webový formulář…).

Rozvoj a optimalizace aplikace je přirozeným krokem v životním cyklu. Prvotním krokem do této fáze je analýza nových požadavků v tzv. změnovém řízení. Během změnových řízení dochází k úpravám a rozšířením funkcionality aplikace. Tyto zásahy s sebou nesou dopady a rizika, a proto je potřeba mít stanovena základní pravidla.

- Kdo může požadavky na změny formulovat.
- Jaké jsou formální nároky na návrh změn. (Např. Struktura dokumentu)
- Místo a způsob evidence nových požadavků.
- Kdo je zodpovědný za posouzení a vyhodnocení požadavků.
- Způsob nacenění změn v projektu.
- Informování navrhovatele o vyhovění nebo zamítnutí požadavku.

(Voříšek, 2015)

Pokud je rozhodnuto, že úprava na základě požadavku proběhne, je potřeba rozlišit, zda jde o dílčí úpravu realizovatelnou v podobě běžné údržby, nebo rozsáhlý zásah vyžadující specifikaci nového projektu. Pokud jsou úpravy natolik rozsáhlé, že se řídící pracovníci rozhodnou pro zadání nového projektu, životní cyklus aplikace se uzavírá. Pro nový projekt se připraví plán a záměr a celý cyklus začne od počátku.

# 10 Plánování a analýza požadavků

Logopedická diagnostika je velice flexibilní pojem, a i její pojetí se různí s každým jednotlivým logopedem. Pokud již logoped zvládl ve své praxi ukotvit postup, jakým diagnostiku provádí musel i zvolit:

- Jak bude svá zjištění a výsledky zaznamenávat.
- Kam je bude zaznamenávat.
- Jak je bude uchovávat.
- Jak se bude vracet k historickým datům, která dozajista potřebuje pro pozorování posunů či propadů klientů.

Mgr. Viola Jónová funguje jako sdílený logoped pro několik různých zařízení a také se věnuje speciální logopedii v hiporehabilitaci. Pečlivé vedení záznamů dat v papírové podobě, jejich zakládání, přesuny a archivace jsou logisticky náročné. Proto digitalizace přinese velkou úlevu, přehlednost dat a několikanásobně zlepší přístup k historickým datům. Další výhodou, kterou software přinese, bude vizualizace naměřených hodnot.

Aplikace bude určená pro stolní počítače a notebooky s operačním systémem Windows 10 a vyšší. Předpokládá se pouze jeden uživatel. Aplikace by měla být schopná uchovávat osobní informace o klientech a záznamy jednotlivých terapií v rámci logopedické praxe. Tyto záznamy by měly být editovatelné. Klienti by měli být rozděleni na aktivní a neaktivní klienty. Hodnoty zaznamenané během terapií by měly mít vizuální výstup.

Naše aplikace nemá za cíl kompletně odstranit papírovou stopu vznikající během vyšetření, ale především zajistit shromažďování strukturovaných dat a jejich přístupnost.

# 11 Definice a specifikace požadavků

## 11.1 Přijetí klienta do praxe

Logoped při příjmu nového klienta potřebuje zaznamenat jeho základní informace a anamnézy, které by mohly odhalit či nastínit původ jeho problémů. Následuje rozpracovaný souhrn dat, která je potřeba před zahájením terapie a diagnostiky získat. Po několika společných schůzkách, jsem vypracovala hrubý návrh formuláře, do kterého lze tyto informace zapsat. Tento návrh můžete vidět jako přílohu 1.

### 11.1.1 Základní informace o klientovi

Prvním krokem je záznam obecných základních informací:

- Jméno a příjmení
- Datum narození
- Datum přijetí do péče – datum, kdy logoped přijal klienta do péče
- Datum odchodu z péče – datum, kdy klient přestává využívat logopedovi péče
- Telefonní číslo, e-mail – v případě dětského klienta jsou uvedeny kontaktní informace na zodpovědného zákonného zástupce
- Diagnóza – diagnóza stanovená jinými specialisty
- Zájmy – znalost zájmů usnadňuje navázání záživné komunikace, získání náklonosti klienta (popřípadě jeho doprovodu) a lze jí taky využít k motivaci během terapie
- Poznámka – prostor na doplňující informace

### 11.1.2 Rodinná anamnéza

Do kategorie rodinná anamnéza patří:

- Jména, příjmení a profese obou rodičů
- Rodina – záznam struktury rodiny (úplná rodina, neúplná rodina, druhotně vzniklá rodina)
- Počet dětí a pořadí – počet dětí v rodině a pořadí ve kterém je klient vůči svým sourozencům (řazeno věkem)
- NKS v rodině – pokud se v rodině vyskytují narušené komunikační schopnosti může to mít souvislost s problémy klienta
- Multilingvismus, jazyky a jazyková situace – v případě, že klient žije v prostředí, kde je užíváno více než jednoho jazyka je potřeba získat informace o jazycích

a jazykové situaci (např. v domácnosti je používán pouze cizí jazyk a čeština přichází pouze z vnějšího okolí)

### 11.1.3 Osobní anamnéza

V osobní anamnéze sledujeme především ranná stádia klientova života.

- Prenatální období – problémy a komplikace, které se vyskytly před porodem
- Perinatální období – problémy a komplikace, které se vyskytly během porodu
- Postnatální období – problémy a komplikace, které se vyskytly po porodu
- Kojení – zda a jak dlouho byl klient kojen + pokud se vyskytly komplikace
- Lahev – zda a jak dlouho byla klientovi podávána kojenecká lahev + pokud se vyskytly komplikace
- Dudlík – zda, popřípadě jak dlouho byl klientovi podáván dudlík
- Motorický vývoj (otáčení/lezení/sed/chůze) – v které fázi se klient nachází, popřípadě průběh motorického vývoje
- Příjem potravy – způsob příjmu potravy nebo specifické odchylky v této oblasti
- Polykání – správnost procesu polykání
- Spánek, Enuresis nocturna – spánkový režim, noční pomočování jeho frekvence a okolnosti
- Hlas – kvalita hlasu a zvláštnosti během jeho tvorby
- Čelist a zuby – správná fyziologie čelisti a zubů + případné patologie
- Předškolní zařízení
- Škola

### 11.1.4 Zdravotní anamnéza

Ze zdravotní anamnézy logopeda zajímají tyto údaje:

- Úrazy – vážná zranění a úrazy
- Operace, narkóza – provedené operace a použití narkózy během
- Nemocnost – často se opakující nebo chronické nemoci
- Medikace – pravidelně užívané medikamenty
- Specialisté – jiní specialisté, které klient navštěvuje
- Logopedie – předchozí či ostatní logopedičtí pracovníci pracující s klientem

### 11.1.5 Artikulace

Artikulace = vytváření hlásek mluvidly

Artikulace bude mít možnost krátkého slovního hodnocení. Tabulka pro záznam o úrovni artikulace při vstupu do péče je součást karty klienta.

Sledování úrovně výslovnosti hlásek. Seznam hlásek s věkem, kdy by dítě mělo být schopno dané hlásky tvořit.

- J (1-2,5 let)
- K, G (2,6 – 3,5 let)
- H, CH (2,6 – 3 let)
- V, F (2,6 – 3,5 let)
- P, B, M (1-2,5 let)
- T, D, N (3,5 – 4 let)
- Ť, Ď, Ň (3,5 – 4,5 let)
- BĚ, PĚ, VĚ, MĚ (3,6 – 4,5 let)
- Č, Š, Ž (4,5 – 5,5 let)
- L (4,5 – 5 let)
- C, S, Z (5,3 – 6,5 let)
- Diferenciace Č, Š, Ž, C, S, Z (6 –7 let)
- R (5,5 – 6,5 let)
- Ř (6 – 7 let)

## 11.2 Diagnostická vyšetření

Cíle diagnostiky narušené komunikační schopnosti jsou definovány odborníky napříč oborem různě. Viktor Lechta mezi hlavní cíle řadí především:

- určení narušení a identifikace druhu
- zjištění příčiny
- možné následky (prognóza)
- stanovení, zda je narušení:
    - trvalé, či přechodné
    - vrozené (vada řeči), nebo získané (porucha řeči)

- zjištění, je-li NKS dominující klinickému obrazu, či je symptomem poškození, nemoci nebo narušení
- stanovení, či si klient problém uvědomuje, nebo neuvědomuje určení stupně narušení (Lechta, 2003)

Průběh vyšetření je vhodné si zaznamenávat pomocí audiovizuálních technologií a následně si ho projít vícekrát pro přesné určení hodnot velkého množství diagnostických markerů. Velké množství zjišťovaných informací klade vysoké nároky na pozorovací schopnosti diagnostika.

Pro kompletní diagnostické vyšetření jsme s Mgr. Jónovou vytvořily formulář sestávající se z šesti tematicky sdružených celků z prvků logopedické diagnosticky a přehledu klientovi artikulace.

Pro první tři roviny jsme zvolily pětistupňový systém hodnocení.

1) Absolutně nezvládá
2) Téměř nezvládá
3) Zvládá s výraznou nápovědou
4) Zvládá s dopomocí
5) Zvládá naprosto samostatně

Pro zbytek oblastí bude hodnocení probíhat pouze třístupňově.

1) Výrazné narušení
2) Mírné odchylky
3) Čisté a bez problémů

Prvotní návrh formuláře pro záznam logopedické diagnostiky najdete v příloze 2.

Průběh terapií se bude uchovávat v sekci terapie, kde je každá terapie označena názvem a datem. V jednotlivých záznamech budou uchovány informace o progresu či propadu ve sledovaných veličinách. Pro případné informace nad rámec hodnotící škály bude možno ke každému záznamu terapie připojit slovní poznámku.

### 11.2.1 Foneticko-fonologická rovina

Zabývá se zvukovou stránkou řeči. V praxi se logoped zabývá výslovností a sluchovým rozlišováním.

Fonetika – obor zabývající se se fyzikální charakteristikou řečových zvuků; podává popis základních i vyšších zvukových jednotek řeči (např. hlásky, slabiky), popis jejich kombinací a změn. (Dvořák, 2001)

Fonologie – jazykovědní obor, zkoumající využívání a fungování zvukových signálů v jazyce; studuje soustavu zvukových prostředků jazyka a jeho systém fonémů; zkoumá funkce hlásek vzhledem k vyšším složkám jazyka (jejich schopnosti rozlišovat slova svou přítomností či záměnou) (Dvořák, 2001)

- Sluchová analýza
    - Schopnost klienta rozkládat slova na hlásky. (příklad zkoušející: PES a dítě hláskuje P E S)
- Sluchová syntéza
    - Schopnost klienta skládat slova z jednotlivých hlásek. Učívá se u dětí ve věku 5-8 let. U starších při podezření na percepční poruchu sluchu. (Př. M Á by klient měl určit, jako slovo má.)
- Znělost/ neznělost
    - U souhlásek znělých (b, v, d, ď, z, ž, g, m, n, ň, j, l, r) při artikulaci zaznívá šum i tón, hlasivky se chvějí.
    - Při artikulaci neznělých souhlásek (p, f, t, ť, s, š, k, c, č) vzniká jen šum.
    - Logoped sluchem hodnotí přítomnost znělosti.
- Tvrdost/měkkost
    - Z pravopisného hlediska se dělí souhlásky na měkké, tvrdé a obojetné. Po měkkých souhláskách (ž, š, č, ř, c, j, ď, ť, ň) píšeme měkké i. Po tvrdých souhláskách (h, ch, r, d, t, n) píšeme tvrdé y.
    - Logoped kontroluje správné vyslovování tvrdosti a měkkosti souhlásek.
    - (tílko x týlko)
- První hláska ve slově
    - Klient umí určit počáteční hlásku zadaného slova (kolem pátého roku věku)
    - (kolem čtvrtého roku by měl být schopný určit první slabiku)

- Poslední hláska ve slově
  - Klient v předškolním věku by měl určit konečnou hlásku zadaného slova. Souhláska na konci slova je lehčeji rozpoznatelná.
- Délka vokálu

### 11.2.2 Lexikálně-sémantická rovina

Řeší především slovní zásobu a užití jazyka. V praxi logoped sleduje šíři slovní zásoby (aktivní i pasivní) a úroveň porozumění.

Lexikální – týkající se slovní zásoby nebo významu slov

Sémantika – nauka o významu slov jazykových jednotek; význam slov je lexikální (slovníkový), gramatický (vyjadřující vztahy mezi plnovýznamovými slovy s gramatickým výrazem nebo gramatickým morfémem) a referenční (k čemu odkazují) (Dvořák, 2001)

- Popis obrázku
  - Při popisu obrázku se sleduje slovní zásoba a fluence, vlastní produkce, pozorovací schopnost, vnímání detailů, oblast zájmů, porozumění a schopnost reagovat na otázky.
- Pojmenování
  - Pozorování slovní zásoby, rychlost reakce, rozsah (slovo, slovní spojení, jednoduché věty, rozvité věty)
- Porozumění
  - Kvalita a úroveň porozumění: Otázka, obrázek, pojem. Snížená schopnost porozumění může indikovat například poruchu sluchu, vývojovou dysfázii nebo lehké mentální postižení.
- Serialita + vyprávění
  - Schopnost dítěte vyprávět děj a sestavit časovou posloupnost. Sleduje se možnost vyjádřit časovou a dějovou linii verbálně i neverbálně (seřazení série obrázků).

- Nadřazené a podřazené pojmy
    - Logoped sleduje úroveň slovní zásoby, schopnosti doplnit seznamy, vytvořit vlastní výčty, třídit slova do kategorií, vypozorovat společné znaky. (U dětí většinou podloženo obrazovými materiály.)
    - Hyponymie – vztah jednotek nižší sémantické roviny (červená, modrá) k jednotkám roviny vyšší (barva)
- Antonyma
    - Protiklady. Znalost protikladů či jejich tvorby jedním z ukazatelů kvality slovní zásoby. Při využití podkladových materiálů lze sledovat i úroveň porozumění.
    - Antonymie – vztah dvou lexémů, které mají opačný význam
- Synonyma
    - Slova stejného významu. Slovní zásoba. Práce se synonymy se využívá především při aktivních dynamických cvičení.
    - Synonymie – více výrazů pro vyjádření jednoho významu
- Homonyma
    - Slova stejně znějící, ale mající jiný význam. Úroveň slovní zásoby. Odlehčovací úkol cílící na rozšíření slovní zásoby a uvědomění si možností jazyka.
    - Homonymie – vztah mezi slovy, která stejně znějí, ale mají různý význam

### 11.2.3 Morfologicko-syntaktická rovina

Řeší gramatiku, stavbu slova a stavbu věty. Logoped sleduje, jak klient zvládá užití gramatiky v praxi.

Morfologie – nauka o tvarových vlastnostech rostlin, živočichů a člověka (jazykověda zkoumá a popisuje vnitřní výstavbu slova z morfémů (morfém je nejmenší nedělitelná jednotka jazyka)) (Dvořák, 2001)

Syntagmatika – gramaticky správné tvoření vět, souvětí

- Skloňování
    - Skloňování je tvoření různých tvarů podle pádu a čísla. Při skloňování rozlišujeme v slově kmen a koncovku. Skloňujeme všechna jména.
    - Logoped pozoruje úroveň gramatiky.

- Časování
    - Časování je tvoření tvarů, které vyjadřují různé osoby a různé číslo pomocí osobních koncovek. Časování zahrnuje všechny tvary slovesa (i neurčité).
    - Logoped pozoruje úroveň gramatiky.
- Určování rodů
    - Mluvnický rod je mluvnická kategorie, kterou rozeznáváme u podstatných jmen, přídavných jmen, některých zájmen a číslovek.
    - Diagnostik pozoruje klientovo vnímání sebe sama a rodů živých bytostí. Následně i vnímání možností jazyka a gramatiky.
- Užívání slovních druhů (se;si)
    - Diagnostik hodnotí výskyt slovních druhů v mluvě klienta. Užívání slovních druhů a používání zvratných zájmen se/si. Diagnostik hodnotí i správnost užití zvratných zájmen.
- Tvorba vět
    - Sledování schopnosti tvořit věty podle vývojové úrovně (Dvouleté děti by měly být schopny tvořit dvouslovné věty atd.)
- Počítání
- Barvy

### 11.2.4 Pragmatická rovina

Užití řeči v praxi. Logoped sleduje schopnost klienta komunikovat, vést rozhovor a používat neverbální komunikaci.

Pragmatika – zkoumání řečových aktů jako úkonů, kterými se dosahuje jistých záměrů

- Dialog
    - Dialog – rozhovor (rozmluva) zpravidla mezi dvěma osobami
    - Sledování schopnosti klienta vést dialog a jeho úrovně. Pochopení výměny rolí, zvládnutí komunikačních strategií (neskáče do řeči, vykání, pozdravení, poděkování, schopnost udržet dějovou linku).
- Zrakový kontakt
    - Schopnost udržení zrakového kontaktu. (Udržuje, neudržuje, délka)
- Neverbální komunikace
    - Diagnostik přejímá zpětnou vazbu klienta na prožívanou situaci. Schopnost rozpoznání rozpoložení a čtení emocí výrazně napomáhá průběhu

terapie. Včasné odhalení nepříjemných pocitů klienta může zabránit znechucení či strachu z dalších setkání.

- Zrcadlení, napodobení
- Popisem neverbální komunikace lze rozvíjet slovní zásobu (Vidím, že se mračíš. Co se stalo?)

- Hra
  - Zvláštnosti ve hře mohou indikovat hlubší problémy.
  - Diagnostik může skrze hru navázat kontakt, navodit uvolnění a motivovat dítě.

### 11.2.5 Motorika

Logoped sleduje motoriky, jejichž úroveň úzce souvisí s vývojem řeči. Problémy v této rovině může vyřešit spolupráce s jinými odborníky (fyzioterapeut, ergoterapeut).

Motorika – soubor pohybových činností lidského organismu řízených nervovým systémem a uskutečňovaných kosterním svalstvem. Skládá se z pohybů spontánních, reflexních, volných a expresivních. (Dvořák, 2001)

- Hrubá motorika
  - Pohyb velkých svalových skupin. Skákání, běhání, jízda na kole.
- Jemná motorika
  - Motorika prstů a koordinace oko ruka. Stěžejním bodem je špetkovitý úchop. Navlékání korálků, třídění drobných předmětů, aktivity s modelínou
- Úchop
  - Sledování patologií psacího náčiní. (Př. svírání tužky v pěsti u pětiletého dítěte.)
- Přítlak
  - Sledování kvality přítlaku. Patologie bývá většinou příliš silný přítlak.
- Úroveň kresby
  - Kresba je jedním z důležitých diagnostických psychologických nástrojů. Pomáhá při navázání kontaktu terapeuta s klientem a určení správné vývojové úrovně.

- Motorika mluvidel
    - Obratnost artikulačních orgánů. Zdravá motorika mluvidel zahrnuje vyvážený poměr aktivace a relaxace orofaciálních svalů. Logoped zkoumá například funkčnost rtů, jazyka a zapojení měkkého patra, čelisti a zubů. Je to jeden z předpokladů správné výslovnosti.

### 11.2.6 Ostatní

- Respirace
    - Sledování odchylek v dýchání (rytmus, dýchání s otevřenými ústy, typy dýchání, kvalita respirace)
- Fonace
    - Fonace – účast hlasu při mluvení; způsob může být měkký (bez tlaku vydechovaného vzdušného proudu na hlasivky) nebo tvrdý (při němž jsou hlasivky pevně sevřené a výdechový proud musí prorazit jejich uzávěr)
    - Tvorba hlasu
    - Logoped sleduje odchylky, při problémech může doporučit návštěvu na foniatrii, kde lze provést hlubší a detailnější vyšetření.
    - Může být indikátorem závažného postižení nebo onemocnění.
- Rezonance
    - Rezonance (ozvuk, souznění) – kmitáním hlasivek vzniká zvuk nepříliš libý, příjemné dotváření lidského vzniká až rezonancí (nazvučením) v přilehlých dutinách (ústní, nosní, hrtanu a hltanu)
    - Logoped zkoumá správný poměr orality a nazality. Tzv. rezonanční vyváženost.
    - Nosovost je rezonance dutiny nosní. Oralita je rezonance dutiny ústní.
    - Logoped se snaží odhalit případné poruchy rezonance, které mohou narušovat komunikační schopnosti. Narušení je silné především u akustického vjemu, a proto může být pozorováno i laikem (např. huhňavost při ucpaném nosu, nebo při rozštěpu orofacionální oblasti a jiných deformací obličeje.)
- Diadochokinéza
    - Diadochokinéza – schopnost vykonávat rychle po sobě pohyby v opačném smyslu

- Schopnost vykonávat střídavé pohyby. Může být motorická – př. Střídání pozic dlaně nebo artikulační, střídání hlásek př. rychlé vyslovování PTK.

- Spolupráce
  - Schopnost dítěte kooperovat během terapie. Respekt z terapeuta, zhoršené chování v přítomnosti rodiče, odchylky a narušené chování.
- Fluence (plynulost)
  - Schopnost produkovat za určitou dobu co nejvíce odpovědí na různé stimuly (např. vymyslet co nejvíce slov začínajících na určitou hlásku apod.)
  - Psycholingvistický termín vyjadřující plynulost slovního (myšlenkového) projevu
  - Poruchami fluence jsou:
    - Koktavost: prolungace delší zaseknutí na jedné hlásce a repetice několikanásobné opakování jedné hlásky, slabiky nebo mluvního celku
    - Breptavost – rychlá nesrozumitelná mluva.
      - Vývojová patologická odchylka plynulosti mluvy. Mezi příznaky je obvykle překotnost tempa řeči (akcelerace intraverbální i interverbální), jehož důsledkem je vynechání slabik. Dochází i k deformaci obsahu – vybočení od tématu, vynechání podstatných informací, absence sémantické soudržnosti apod.

# 12 Návrh aplikace ANDI

## 12.1 Návrh uživatelského rozhraní

Uživatelské rozhraní umožní uživateli na úvodní obrazovce vyhledávat mezi stávajícími klienty, neaktivními klienty a založením nového klienta. V případě založení nového klienta se otevře nová obrazovka s příslušným formulářem. Do toho bude nutné zaznamenat obecné osobní a kontaktní informace a podle potřeby i rodinnou, osobní a zdravotní anamnézu. Další položkou bude záznam o artikulaci klienta pro jednotlivé hlásky, jejich skupiny, popřípadě jejich rozlišování.

Při vyhledávání stávajícího klienta uživatel proklikem na jméno ze seznamu otevře novou obrazovku s kartou klienta, která nese osobní informace, terapie a grafické vyhodnocení. Mezi jednotlivými kartami bude lze přepínat pomocí tlačítek menu umístěných v horní části obrazovky.

Karta osobní informace bude zobrazovat především data z tabulky klient a k ní navázaných tabulek Rodinná anamnéza, Osobní anamnéza, Zdravotní anamnéza a Artikulace. Data lze doplnit či upravit pomocí tlačítka horní části obrazovky. Pokud se klient nachází v seznamu klientů, kteří opustili logopedickou praxi bude mít uživatel možnost odstranit klienta. Po stisknutí tohoto tlačítka bude uživatel upozorněn na chystající se odstranění klienta a veškerých jeho terapií ze sytému. Uživatel může své rozhodnutí potvrdit nebo zvrátit.

Karta Terapie zobrazí uživateli tlačítko pro tvorbu nového záznamu terapie a seznam zaznamenaných terapií, a to v pořadí v jakém byly zaznamenány do systému. Zobrazená tabulka obsahuje datum a název terapie. Po prokliknutí na danou terapii se otevře obrazovka s přehledem dané terapie. V horní části se nachází tlačítka pro možnost úpravy či doplnění dat, návratu na přehled klientových terapií a kompletního odstranění záznamu terapie.

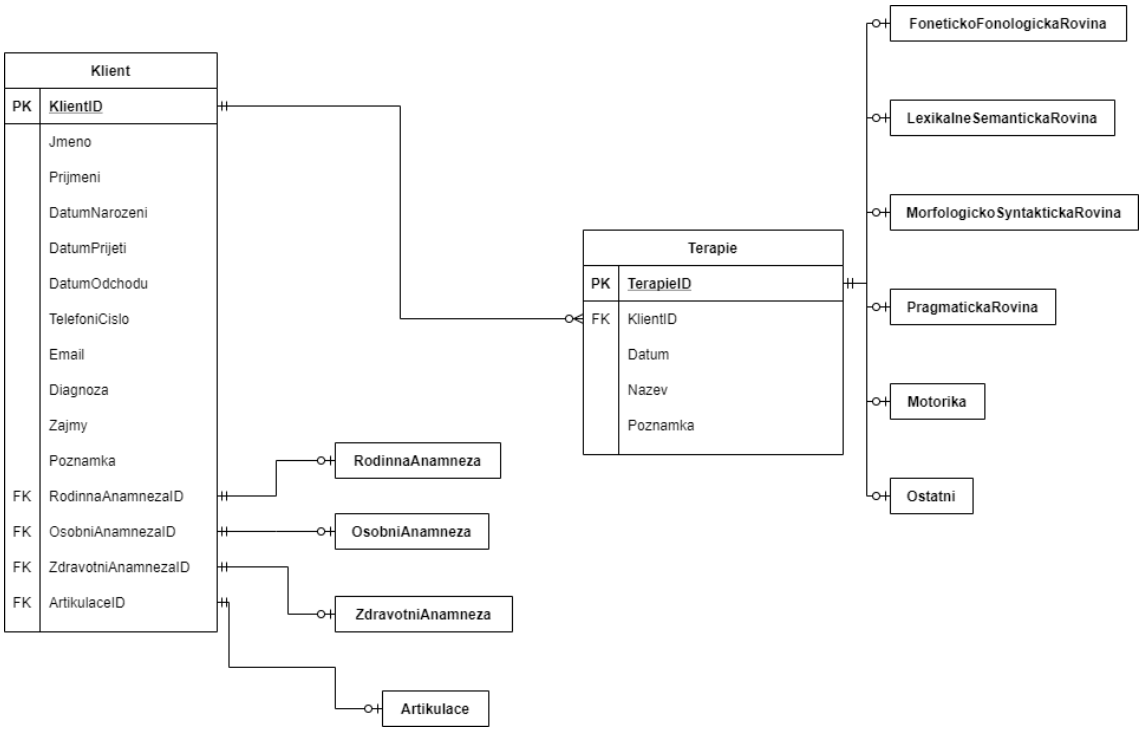
Formulář pro záznam terapie bude rozdělen do šesti menších částí, které reprezentují jednotlivé roviny. V pravé horní části obrazovky se zobrazuje přehled informací o artikulaci klienta (pokud by logoped do aplikace zaznamenával v reálném čase tato část mu pomůže při cílení na potřebné problémové hlásky a jejich skupiny) a v dolní části je prostor pro zanechání slovního komentáře k terapii.

Karta graf zobrazí uživateli obrazovku s šesticí grafů. Každý graf se bude zaměřovat na jednu z rovin terapie. Na ose x budou všechny terapie a na ose y průměrné bodové hodnocení z dané roviny během dané terapie. Tato šestice grafů pak pomůže logopedovi přehledně vidět klientův posun v rámci jednotlivých rovin v průběhu času.

## 12.2 Datový model

Z předchozích analýz jasně vyplynulo, že v aplikaci budou vystupovat dvě hlavní entity. První entitou je klient, ke kterému se váže velké množství osobních, kontaktních a diagnostických informací. Druhá entita je terapie, ta musí být přiřazena ke klientovi a sdružuje informace o tématu terapie, datumu konání, naměřených hodnotách a případné poznámky. Jelikož jsou tyto entity příliš komplikované, rozdělila jsem je na menší části. Finální databáze se skládá z 12 tabulek a rozdělení dat probíhalo na základě příslušnosti dat k logickému celku, viz následující obrázek 2.

*Obrázek 2 ER diagram databáze ANDI*




Zdroj: Vlastní zpracování (2020)

### 12.2.1 Klient

Tabulka Klient drží informace o jménu, příjmení, datu narození, datu přijetí do praxe, datu opuštění praxe, diagnóze, zájmech klienta, a kontaktních informací v podobě

e-mailové adresy a telefonního čísla. Dále tabulka obsahuje čtyři cizí klíče z tabulek RodinnaAnamneza, OsobniAnamneza, ZdravotniAnamneza a Artikulace.

Spojením těchto pěti tabulek dostáváme detailní obraz klienta, jeho anamnéz a přehled o stavu jeho řeči ve vztahu k jednotlivým hláskám.

### 12.2.2 Rodinná anamnéza

Tabulka RodinnaAnamneza udržuje informace o jménech, příjmeních a zaměstnáních obou rodičů. Dalšími poli jsou typ rodiny, ve které klient vyrůstá, počet dětí v rodině, pořadí narození vůči sourozencům (možnost uvést vlastní i nevlastní sourozence), popis narušených komunikačních v rodině (klient může přejímat návyky od svého okolí nebo mít vrozené predispozice), jazyky (toto pole slouží nejen pro zápis jazyků které dítě ovládá, ale i k popisu celkové jazykové situace v přímém okolí klienta).

### 12.2.3 Osobní anamnéza

Tabulka OsobniAnamneza sdružuje informace především z raných fází klientova života. V této tabulce nalezneme sloupce prenatální ob., perinatální ob., postnatální ob, kojení, lahev, dudlík, motorický vývoj, příjem potravy, polykání, spánek, enuresis nocturna, hlas, čelist a zuby společně se sloupci o navštěvovaných vzdělávacích institucích (předškolní zařízení, škola).

### 12.2.4 Zdravotní anamnéza

Tabulka ZdravotniAnamneza slouží pro záznam zdravotních indispozic, které mohou mít vliv a klientovu řeč a komunikaci a specialistů, kteří v dané problematice s klientem spolupracují. Tato tabulka obsahuje sloupce úrazy, operace, nemocnost, medikace, specialisté a logopedie.

### 12.2.5 Artikulace

Tabulka Artikulace obsahuje sloupce pro jednotlivé hlásky, skupiny a diferenciace, jak byly uvedené ve specifikaci požadavků.

### 12.2.6 Terapie

Tabulka Terapie je stěžejní pro propojení klienta s jeho diagnostickými daty. Obsahuje sloupce KlientId (cizí klíč z tabulky Klient), název terapie, datum, poznámka.

Množství všech možných diagnostických ukazatelů je příliš velké, a proto jsme již při specifikaci požadavků vytvořily šest podskupin (rovin), do kterých lze ukazatele rozdě-

lit. Každá z těchto rovin je v databázi zastoupena vlastní tabulkou. Tyto tabulky obsahují sloupec TerapieId jako cizí klíč.

### 12.2.7 Foneticko-fonologická rovina

Tabulka FonetickoFonologickaRovina obsahuje sloupce pro bodové hodnocení v oblastech Sluchová analýza, Sluchová syntéza, Znělost, Tvrdost/Měkkost, První hláska, Poslední hláska, Délka vokálu.

### 12.2.8 Lexikálně-sémantická rovina

Tabulka LexikalneSemantickaRovina uchovává body získané v oblastech Popis obrázku, Pojmenováni, Porozumění, Serialita vyprávění, Nad/Podřazené pojmy, Antonyma, Synonyma, Homonyma.

### 12.2.9 Morfologicko-syntaktická rovina

Tabulka MorfologickoSyntaktickaRovina drží bodové hodnoty pro Skloňovaní, Časování, Určování rodu, Užívání slovních druhů, Tvorba vět, Počítaní, Barvy.

### 12.2.10 Pragmatická rovina

Tabulka PragmatickaRovina má pouze čtyři diagnostické sloupce a to Dialog, Zrakový kontakt, Neverbální komunikace a Hra.

### 12.2.11 Motorika

Tabulka Motorika slouží pro uchování dat Jemná motorika, Hrubá motorika, Úchop, Přítlak, Úroveň kresby a Motorika mluvidel.

### 12.2.12 Ostatní

Tabulka Ostatni slouží pro diagnostická měření, která typově nezapadala do ani jedné z předchozích kategorií. Jedná se o hodnocení Respirace, Fonace, Resonance, Diadochokinéza, Spolupráce, Fluence.

# 13 Implementace aplikace ANDI

Řešení aplikace ANDI se skládá ze dvou projektů. Prvním je Windows Aplikace `Andi`, tento projekt obsahuje veškerý XAML kód uživatelského prostředí a pokrývá vrstvu View z MVVM architektury. Druhým projektem je knihovna tříd `AndiLibrary`, která obsahuje veškerou logiku aplikace a napojení této logiky na uživatelské prostředí. Tento projekt obsahuje všechny soubory vrstev ViewModel a Model včetně několika pomocných tříd.

V následujících podkapitolách se nachází popis samotného kódu a jeho ukázky. Jelikož je aplikace poměrně rozsáhlá popsala jsem pouze klíčové části. Veškeré zdrojové soubory celé funkční aplikace se nachází na přiloženém datovém disku.

## 13.1 Logo

Logo programu (viz obrázek 3) se skládá z názvu ANDI (zkratové slovo utvořené ze slov anamnéza a diagnostika) a obrázku úsměvu s vyplazeným jazykem. Grafická úprava loga byla zpracována Dis. Lucií Daňkovou.

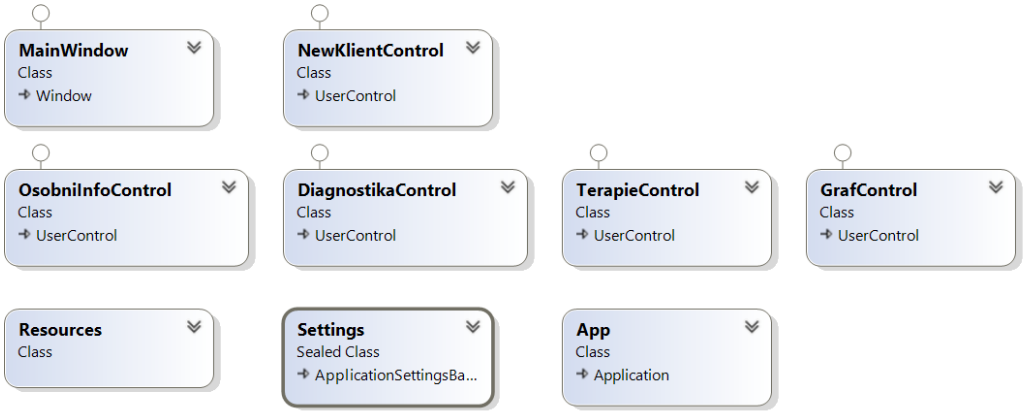
*Obrázek 3 Logo ANDI*


Zdroj: Lucie Daňková (2020)

## 13.2 Uživatelské prostředí

Uživatelské prostředí aplikace ANDI je tvořeno hlavním oknem `MainWindow`, sadou tříd typu user control a automaticky vygenerovanými třídami `App`, `Settings` a `Resources`, viz následující obrázek 4 diagram tříd UI.

*Obrázek 4 Diagram tříd UI*




Zdroj: Vlastní zpracování (2021)

Stylizace jednotlivých prvků v celé aplikaci probíhá podle stylů, které jsem definovala v třídě `App.xaml`. Díky používání stylů lze dynamicky měnit vlastnosti všech objektů daného stylu pomocí jednoduché úpravy. V této třídě jsem také definovala datové šablony (`DataTemplate`), které pomohou propojení mezi `UserControls` ve vrstvě View a třídami vrstvy ViewModel. Viz ukázka kódu 1.

*Ukázka kódu 1 DataTemplate*

```
<DataTemplate DataType="{x:Type viewmodels:OsobniInfoVM}">
    <views:OsobniInfoControl/>
</DataTemplate>

<DataTemplate DataType="{x:Type viewmodels:TerapieVM}">
    <views:TerapieControl/>
</DataTemplate>

<DataTemplate DataType="{x:Type viewmodels:NewKlientVM}">
    <views:NewKlientControl/>
</DataTemplate>

<DataTemplate DataType="{x:Type viewmodels:GrafVM}">
    <views:GrafControl/>
</DataTemplate>
```

Zdroj: Vlastní zpracování (2021)

Veškerá komunikace mezi prvky uživatelského rozhraní a vrstvou ViewModel probíhá díky procesu DataBinding. Díky těmto datovým vazbám jsou uživatelské prvky schopny zobrazovat správná data a hodnoty. Uživatel skrze tyto vazby vysílá svoje požadavky a ty jsou logikou aplikace zpracovávány. V následující ukázce kódu 2 můžete vidět napojení prvku `ListView` na `ObservableCollection<KlientModel> AktivniKlienti` z třídy `MainVM`. Díky této datové vazbě se každý klient z kolekce zobrazuje v seznamu ve tvaru příjmení + jméno.

*Ukázka kódu 2 DataBinding*

```
<ListView ItemsSource="{Binding AktivniKlienti}"
    SelectedValue="{Binding SelectedKlient}" Style="{StaticResource listViewStyle}">
    <ListView.ItemTemplate>
        <DataTemplate>
            <Grid >
                <TextBlock>
                    <Run Text="{Binding Prijmeni}"/>
                    <Run Text="{Binding Jmeno}"/>
                </TextBlock>
            </Grid>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

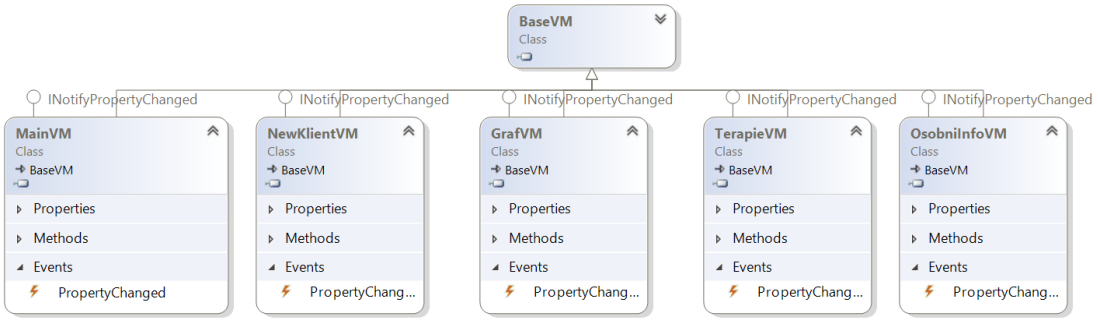
Zdroj: Vlastní zpracování (2021)

## 13.3 Vrstva ViewModel

Aplikace obsahuje celkem šest VM tříd implementujících rozhraní `INotifyPropertyChanged` a devět tříd implementujících rozhraní `ICommand`. Tyto třídy jsou jádrem celé aplikace.

Jak je vidno z následujícího obrázku 5, BaseVM je rodičovskou třídou pro všechny ostatní třídy vrstvy ViewModel. Díky této dědičnosti je zajištěno fungování commandu `UpdateViewCommand`.

*Obrázek 5 Class diagram tříd ViewModel*




Zdroj: Vlastní zpracování (2021)

### 13.3.1 INotifyPropertyChanged

`INotifyPropertyChanged` je rozhraní sloužící pro udržení aktuálnosti mezi objekty grafického rozhraní a objekty k nim navázaných. Například, aby změna vlastnosti třídy byla reflektována uživatelským rozhraním. K tomu, aby bylo rozhraní implementováno, je potřeba deklarovat `PropertyChanged` událost a `OnPropertyChanged` metodu (viz ukázka kódu 3). Tato metoda se posléze volá, když chceme vlastnost aktualizovat.

*Ukázka kódu 3 INotifyPropertyChanged*

```
public event PropertyChangedEventHandler PropertyChanged;

8 references
private void OnPropertyChanged(string propertyName)
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}
```

Zdroj: Vlastní zpracování (2021)

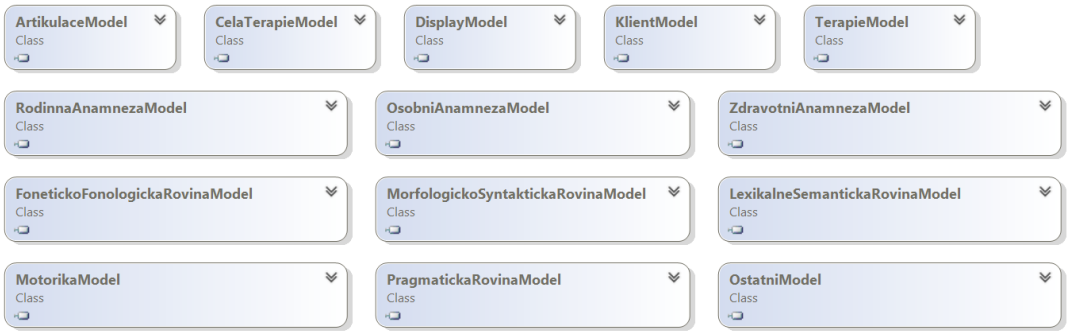
### 13.3.2 ICommand

`ICommand` rozhraní je použito u tříd, které slouží k zadávání příkazů na základě chování prvků uživatelského rozhraní (nejčastěji prvku `Button`). Metody podmíněné rozhraním `ICommand` `CanExecute(Object)` (zajišťuje spustitelnost příkazu za aktuálního stavu) a `Execute(Object)` (samotná metoda volaná po spuštění příkazu). Další podmínkou tohoto rozhraní je událost (event) `CanExecuteChanged`, která se vyvolá v okamžiku, kdy dojde ke změnám, které by mohly ovlivnit spuštění daného příkazu.

## 13.4 Modely

Aplikace ANDI má čtrnáct modelových tříd, viz následující obrázek 6. `ArtikulaceModel`, `FonetickoFonologickaRovinaModel`, `KlientModel`, `LexikalneSemantickaRovinaModel`, `MorfologickoSyntaktickaRovinaModel`, `MotorikaModel`, `OsobniAnamnezaModel`, `OstatniModel`, `PragmatickaRovinaModel`, `RodinnaAnamnezaModel`, `TerapieModel` a `ZdravotniAnamnezaModel` kopírují tabulky databáze (`KlientModel` má navíc vlastnost `CeleJmeno` zpřístupněnou pouze pro čtení. `DisplayModel` slouží při nastavování vlastnosti `Visibility` u prvků uživatelského rozhraní. `CelaTerapieModel` při své inicializaci vytvoří instance tříd všech šesti rovin.

*Obrázek 6 Model třídy*




Zdroj: Vlastní zpracování (2021)

## 13.5 Úvodní obrazovka

Na úvodní obrazovce si uživatel může zobrazit konkrétního klienta ze seznamů aktivních a neaktivních klientů, přepnout se do režimu tvorby nového klienta nebo vyhledávat pomocí vyhledávacího pole. Vizuální podoba úvodní obrazovky viz příloha 3.

### 13.5.1 MainWindow

Okno `MainWindow` obsahuje všechny uživatelské prvky úvodní obrazovky podle předchozího návrhu, plus `Menu` a `ContentControl`, které mají vlastnost `visibility` nastavenou na hodnotu `colappsed` dokud není vybrán klient nebo zvolena možnost vytvoření nového klienta. Menu slouží k přepínání mezi ostatními obrazovkami aplikace a objekt `ContentControl` poté obsahuje konkrétní objekt typu `UserControl`. Přepínání mezi jednotlivými obrazovkami zajišťuje command `UpdateViewCommand`,

který pomocí předávaného parametru inicializuje příslušný ViewModel. Následuje ukázka kódu 4 pro metodu Execute daného commandu.

*Ukázka kódu 4 UpdateViewCommand*

```
8 references
public void Execute(object parameter)
{
    if (parameter.ToString() == "OsobniInfo")
    {
        viewmodel.SelectedViewModel = new OsobniInfoVM();
    }
    else if (parameter.ToString() == "Terapie")
    {
        viewmodel.SelectedViewModel = new TerapieVM();
    }
    else if (parameter.ToString() == "Graf")
    {
        viewmodel.SelectedViewModel = new GrafVM();
    }
    else if (parameter.ToString() == "NewKlient")
    {
        viewmodel.DisplayNewClientForm();
        viewmodel.SelectedViewModel = new NewKlientVM();
    }
    else if (parameter.ToString() == "Zpet")
    {
        viewmodel.SelectedKlient = null;
        viewmodel.SelectedViewModel = null;
        viewmodel.PopulateKlientLists();
    }
}
```

Zdroj: Vlastní zpracování (2021)

Dalším důležitým commandem spojujícím `MainWindow` a `MainVM` je `SearchCommand`. Tento command je spuštěn po kliknutí na tlačítko Hledat vedle vyhledávacího pole, v případě, že pole není prázdné. Následuje ukázka kódu 5 pro metody CanExecute a Execute.

*Ukázka kódu 5 SearchCommand*

```
8 references
public bool CanExecute(object parameter)
{
    string query = parameter as string;

    if (string.IsNullOrWhiteSpace(query))
        return false;
    return true;
}

8 references
public void Execute(object parameter)
{
    VM.MakeQuery();
}
```

Zdroj: Vlastní zpracování (2021)

### 13.5.2 MainVM

ViewModel MainVM obsahuje devět vlastností a šest metod. Metody začínající slovem `Display` slouží k zobrazování a skrývání uživatelských prvků podle potřeby. Metoda `PopulateKlientLists` v následující ukázce kódu 6 naplní kolekce `AktivniKlienti` a `NeaktivniKlienti` údaji z databáze. Metoda `MakeQuery` je postavena na stejném principu s upraveným sql příkazem za použití vlastnosti `Query`.

*Ukázka kódu 6 PopulateKlientLists*

```
2 references
public void PopulateKlientLists()
{
    AktivniKlienti.Clear();
    NeaktivniKlienti.Clear();
    string sql = "select * from Klient order by Prijmeni";
    var clientList = SqliteDataAccess.LoadData<KlientModel>(sql, new Dictionary<string, object>());
    foreach (KlientModel klient in clientList)
    {
        if (klient.DatumOdchodu == null | klient.DatumOdchodu == "")
        {
            AktivniKlienti.Add(klient);
        }
        else
        {
            NeaktivniKlienti.Add(klient);
        }
    }
}
```

Zdroj: Vlastní zpracování (2021)

## 13.6 Nový klient

Obrazovka Nový klient zobrazuje prázdný formulář pro založení karty klienta. Viz příloha 4.

### 13.6.1 NewKlientControl

Formulář se sestává z objektů typu `TextBlock` a `TextBox`. Pokud uživatel vyplní základní osobní informace o novém klientovi může data uložit do databáze pomocí tlačítka Uložit do databáze navázaného na `SaveNewKlientCommand`. Pokud se uživatel rozmyslí a data ukládat nechce, klikem na položku menu Zpět na hlavní stranu se vrátí na úvodní obrazovku.

### 13.6.2 NewKlientVM

Nejpodstatnější metodou tohoto ViewModelu je `SaveFullKlient`. Tato metoda zavolá verifikační metodu `NotNullKlient` a pokud je vrácena hodnota true uloží data do databáze pomocí metod Insert. Následující ukázka kódu je pro metodu `InsertAr-`

`tikulace`, která ukládá data do tabulky Artikulace. Pokud `NotNullKlient` vrátí false je uživateli oznámeno, že musí formulář vyplnit před uložením.

*Ukázka kódu 7 InsertArtikulace*

```
1 reference
public void InsertArtikulace()
{
    string sql = "insert into Artikulace (J, PBM, L, KG, TDN, CSZ, CCH, TDNMekce, R, " +
        "VF, CSZMekce, RHacek, BePeVeMe, CSZRozdil)" +
          "values (@J, @PBM, @L, @KG, @TDN, @CSZ, @CCH, @TDNMekce, @R, " +
          "@VF, @CSZMekce, @RHacek, @BePeVeMe, @CSZRozdil)";

    Dictionary<string, object> parameters = new Dictionary<string, object>
        {
            {"@J", Artikulace.J },
            {"@PBM", Artikulace.PBM},
            {"@L", Artikulace.L},
            {"@KG", Artikulace.KG},
            {"@TDN", Artikulace.TDN},
            {"@CSZ", Artikulace.CSZ},
            {"@CCH", Artikulace.CCH },
            {"@TDNMekce", Artikulace.TDNMekce},
            {"@R", Artikulace.R},
            {"@VF", Artikulace.VF},
            {"@CSZMekce", Artikulace.CSZMekce},
            {"@RHacek", Artikulace.RHacek},
            {"@BePeVeMe", Artikulace.BePeVeMe},
            {"@CSZRozdil", Artikulace.CSZRozdil}
        };
    SqliteDataAccess.SaveData(sql, parameters);
    GetLastArtikulaceIDIntoNewClient();
}
```

Zdroj: Vlastní zpracování (2021)

## 13.7 Osobní informace

Tato obrazovka (viz příloha 5) obsahuje formulář s tzv. kartou klienta. Můžeme zde najít obecné osobní informace o klientovi, kontaktní informace, rodinnou anamnézu, osobní anamnézu, zdravotní anamnézu a přehled o artikulaci klienta. V případě, že klient je již neaktivní (vyplněné datum odchodu) má uživatel možnost klienta a veškeré záznamy o jeho osobě vymazat.

### 13.7.1 OsobniInfoControl

Pomocí tlačítka napojeného na command `EditOsobniInfoCommand` lze přepínat mezi režimem editace a prohlížení. Viz následující ukázka kódu 8. V režimu edit jsou informace zobrazené v objektech typu `TextBox` a uživatel je může libovolně přepisovat. V režimu notEdit uživatel vidí informace z příslušných tabulek v objektech typu `TextBlock`.

*Ukázka kódu 8 EditOsobniInfoCommand*

```
8 references
public void Execute(object parameter)
{
    if (OsobniInfo.EnableEdit == true)
    {
        OsobniInfo.UpdateOsobniInfo();
        OsobniInfo.NotEnableEditDisplay();
        OsobniInfo.EnableEdit = false;
        OsobniInfo.ButtonEditText = "Upravit";
    }
    else if (OsobniInfo.EnableEdit == false)
    {
        OsobniInfo.EnableEditDisplay();
        OsobniInfo.EnableEdit = true;
        OsobniInfo.ButtonEditText = "Uložit";
    }
}
```

Zdroj: Vlastní zpracování (2021)

### 13.7.2 OsobniInfoVM

O získání všech potřebných dat z databáze se stará metoda `GetFullAnamneza`. Tato metoda postupně volá Get metody pro všechny potřebné tabulky. Následuje ukázka jedné z těchto metod `GetZdravotniAnamneza`. Viz ukázka kódu 9.

*Ukázka kódu 9 GetZdravotniAnamneza*

```
1 reference
public void GetZdravotniAnamneza()
{
    string sql = $"select * from ZdravotniAnamneza where Id=\"{selectedKlient.ZdravotniAnamnezaId}\" ";
    var anamnezaList = SqliteDataAccess.LoadData<ZdravotniAnamnezaModel>(sql, new Dictionary<string, object>());
    if (anamnezaList.Count == 0)
    {
        ZdravotniAnamneza = new ZdravotniAnamnezaModel();

    }
    else
    {
        zdravotniAnamneza = anamnezaList[0];
    }

}
```

Zdroj: Vlastní zpracování (2021)

V případě, že uživatel se přepnul do režimu editace a rozhodl se změny uložit volá se metoda `UpdateOsobniInfo`. Tato metoda aktualizuje data v databázi. Jako ukázku jsem opět uvedla metodu zdravotní anamnézy, `UpdateZdravotniAnamneza`. Viz ukázka kódu 10.

*Ukázka kódu 10 UpdateZdravotniAnamneza*

```
1 reference
public void UpdateZdravotniAnamneza()
{
    string updatesql = $"UPDATE ZdravotniAnamneza SET Urazy=@Urazy, Operace=@Operace, Nemocnost=@Nemocnost, " +
        $"Medikace=@Medikace, Specialiste=@Specialiste, Logopedie=@Logopedie where Id=\"{selectedKlient.ZdravotniAnamnezaId}\" ";
    Dictionary<string, object> parameters = new Dictionary<string, object>
        {
            {"@Urazy", ZdravotniAnamneza.Urazy},
            {"@Operace", ZdravotniAnamneza.Operace},
            {"@Nemocnost", ZdravotniAnamneza.Nemocnost},
            {"@Medikace", ZdravotniAnamneza.Medikace},
            {"@Specialiste", ZdravotniAnamneza.Specialiste},
            {"@Logopedie", ZdravotniAnamneza.Logopedie}

        };
    SqliteDataAccess.SaveData(updatesql, parameters);

}
```

Zdroj: Vlastní zpracování (2021)

Pokud je klient již neaktivní, lze pomocí tlačítka smazat veškerá data, která se k dané osobě vztahují. Toto tlačítko je napojené na command `DeleteKlientCommand`, který si od uživatele vyžádá potvrzení skrze `MessageBox` a následně spustí metodu `DeleteKlient` z následující ukázky kódu 11.

*Ukázka kódu 11 DeleteKlient*

```
1 reference
public void DeleteKlient()
{
    DeleteTerapieKlienta();
    string sql = $"delete from OsobniAnamneza where Id=\"{selectedKlient.OsobniAnamnezaId}\" ";
    SqliteDataAccess.DeleteData(sql);
    sql = $"delete from RodinnaAnamneza where Id=\"{selectedKlient.RodinnaAnamnezaId}\" ";
    SqliteDataAccess.DeleteData(sql);
    sql = $"delete from ZdravotniAnamneza where Id=\"{selectedKlient.ZdravotniAnamnezaId}\" ";
    SqliteDataAccess.DeleteData(sql);
    sql = $"delete from Artikulace where Id=\"{selectedKlient.ArtikulaceId}\" ";
    SqliteDataAccess.DeleteData(sql);
    sql = $"delete from Klient where Id=\"{selectedKlient.Id}\" ";
    SqliteDataAccess.DeleteData(sql);

}
```

Zdroj: Vlastní zpracování (2021)

## 13.8 Terapie

Úvodní obrazovka v kartě Terapie (viz příloha 6) je `ListView` terapií klienta v pořadí, v jakém byly zadány do databáze a tlačítko Nová terapie. Přes toto tlačítko se dostaneme na obrazovku prázdného formuláře pro záznam terapie. V pravém horním rohu je zobrazený přehled klientovi artikulace, takže pokud logoped aplikaci využívá v reálném čase, může sloužit i jako rychlá reference. Na záznamy konkrétních terapií se lze podívat proklikem ze seznamu terapií (viz příloha 7).

### 13.8.1 TerapieControl

`TerapieControl` může zobrazovat tři různé obrazovky. Zobrazení skupin prvků tvořících jednu obrazovku je řízeno vlastností `Display` ViewModelu `TerapieVM`. Při prokliku na detailním záznam terapie je možné přepínat mezi režimy úpravy a prohlížení. Také je možné pomocí tlačítka v pravém horním rohu odstranit celý záznam terapie.

### 13.8.2 TerapieVM

`TerapieVM` se skládá z poměrně mnoha vlastností a metod. Důležitou část tvoří metody pracující s databází (`GetRoviny`, `SaveNewTerapie`, `UpdateTerapie`, `DeleteTerapie`) a také metody pro validaci zadaných hodnot podle nastavených principů softwaru `NotNullValidate`. V následující ukázce kódu 12 je metoda `SaveNewTerapie`, která využívá `Insert` a zároveň `NotNullValidate` metody pro všechny roviny. Tato metoda se volá až po ověření správnosti vyplnění informací pro tabulku Terapie, takže tato validace je ošetřena v jiné části kódu.

*Ukázka kódu 12 SaveNewTerapie*

```
1 reference
public void SaveNewTerapie()
{
    InsertTerapie();
    if (NotNullValidateFonetickoFonologickaRovina()) { InsertFonetickoFonologickaRovina(); }
    if (NotNullValidateLexikalneSemantickaRovina()) { InsertLexikalneSemantickaRovina(); }
    if (NotNullValidateMorfologickoSyntaktickaRovina()) { InsertMorfologickoSyntaktickaRovina(); }
    if (NotNullValidatePragmatickaRovina()) { InsertPragmatickaRovina(); }
    if (NotNullValidateMotorika()) { InsertMotorika(); }
    if (NotNullValidateOstatni()) { InsertOstatni(); }

}
```

Zdroj: Vlastní zpracování (2021)

## 13.9 Grafické zobrazení výsledků terapie

Obrazovka s grafickým výstupem obsahuje šest sloupcových grafů (jeden pro každou z rovin). Viz příloha 8. Sloupce reprezentují průměrné bodové hodnocení (na ose y) získané během terapie (na ose x) v rámci dané roviny. Díky těmto grafům získá logoped jasný přehled, na které roviny se v průběhu času soustředí nejvíce a zda se klient zlepšuje či horší. Tento výstup může také sloužit rodičům klientů pro lepší vizualizaci progresu.

### 13.9.1 GrafControl

Pro tvorbu grafů je v `GrafControl` použitý nuget package LiveCharts. Celá obrazovka se sestává z šesti objektů typu `CartesianChart`. Zobrazované hodnoty na osách x a y jsou zajištěné pomocí DataBinding z `GrafVM`.

### 13.9.2 GrafVM

Zajištění všech potřebných dat pro zobrazení grafů je dosažené pomocí série get metod. Tyto jsou zastřešené v metodě `GetAllData`. Tato metoda postupně volá ostatní get metody pro jednotlivé roviny a plní kolekce `ChartValues` získanými daty. Ve dvou následujících ukázkách kódu 13 a 14 je metoda `GetDataPragmatickaRovina` a detail zabaleného foreach cyklu.

*Ukázka kódu 13 GetDataPragmatickaRovina*

```
1 reference
public void GetDataPragmatickaRovina()
{
    PrumeryPragmatickaRovina = new ChartValues<double>();
    string sql = $"SELECT PragmatickaRovina.Id, TerapieId, Dialog, ZrakovyKontakt, NeverbalniKomunikace, " +
        $"Hra FROM PragmatickaRovina INNER JOIN Terapie ON PragmatickaRovina.TerapieId = Terapie.Id " +
        $"WHERE Terapie.KlientId=\"{SelectedKlientSingleton.Instance.Klient.Id}\" ORDER BY TerapieId ";
    var PragmatickaRovinaList = SqliteDataAccess.LoadData<PragmatickaRovinaModel>(sql, new Dictionary<string, object>());

    if (PragmatickaRovinaList.Count != 0)
    {
        int i = 0;
        PragmatickaRovinaModel PR = PragmatickaRovinaList[i];
        foreach (var terapie in TerapieList)...
    }
    else
    {
        foreach (string Jmeno in TerapieJmena)
        {
            PrumeryPragmatickaRovina.Add(0);
        }
    }
}
```

Zdroj: Vlastní zpracování (2021)

*Ukázka kódu 14 GetDataPragmatickaRovina detail cyklu foreach*

```
foreach (var terapie in TerapieList)
{
    if (terapie.Id == PR.TerapieId && i < PragmatickaRovinaList.Count)
    {
        int suma = 0;
        int notNullFields = 0;

        if (PR.Dialog != 0) { suma += PR.Dialog; notNullFields++; }
        if (PR.ZrakovyKontakt != 0) { suma += PR.ZrakovyKontakt; notNullFields++; }
        if (PR.NeverbalniKomunikace != 0) { suma += PR.NeverbalniKomunikace; notNullFields++; }
        if (PR.Hra != 0) { suma += PR.Hra; notNullFields++; }

        if (notNullFields == 0) { notNullFields = 1; }

        double prumer = (double)suma / notNullFields;
        PrumeryPragmatickaRovina.Add(prumer);
        i++;
        if (i < PragmatickaRovinaList.Count) { PR = PragmatickaRovinaList[i]; }
    }
    else   {  PrumeryPragmatickaRovina.Add(0); }
}
```

Zdroj: Vlastní zpracování (2021)

## 13.10 Pomocné třídy

Pomocné třídy slouží pro komunikaci s databází a přenos informace jaký klient je právě zvolen napříč ViewModely.

### 13.10.1 SelectedKlientSingleton

Aby byla zachována informace o vybraném klientovi napříč ViewModely vytvořila jsem singleton třídu `SelectedKlientSingleton`, která má podle návrhového vzoru singleton vždy maximálně jednu instanci. Kód k této metodě je v ukázce kódu 15.

*Ukázka kódu 15 SelectedKlientSingleton*

```
16 references
public sealed class SelectedKlientSingleton
{
    1 reference
    private SelectedKlientSingleton()
    {

    }

    private static readonly Lazy<SelectedKlientSingleton> instance = new Lazy<SelectedKlientSingleton>(() => new SelectedKlientSingleton());
    11 references
    public static SelectedKlientSingleton Instance
    {
        get
        {
            return instance.Value;
        }
    }

    12 references
    public KlientModel Klient { get; set; }
}
```

Zdroj: Vlastní zpracování (2021)

### 13.10.2 SqliteDataAccess

Pro přístup k databázi jsem vytvořila třídu `SqliteDataAccess`, která obsahuje metody: `LoadData`, `SaveData`, `DeleteData` a `ToDynamicParameters`. Všechny čtyři metody jsou viditelné v následujících dvou ukázkách kódu 16 a 17.

*Ukázka kódu 16 LoadData, SaveData*

```
33 references
public static List<T> LoadData<T>(string sqlStatement, Dictionary<string, object> parameters, string connectionName = "Default")
{
    DynamicParameters p = parameters.ToDynamicParameters(); //Using dapper

    using (IDbConnection cnn = new SQLiteConnection(DataAccessHelpers.LoadConnectionString(connectionName)))
    {
        var rows = cnn.Query<T>(sqlStatement, p);
        return rows.ToList();
    }
}

24 references
public static void SaveData(string sqlStatement, Dictionary<string, object> parameters, string connectionName = "Default")
{


    DynamicParameters p = parameters.ToDynamicParameters();

    using (IDbConnection cnn = new SQLiteConnection(DataAccessHelpers.LoadConnectionString(connectionName)))
    {
        cnn.Execute(sqlStatement, p);
    }
}
```

Zdroj: Vlastní zpracování (2021)

*Ukázka kódu 17 DeleteData, ToDynamicParameters*

```
19 references
public static void DeleteData(string sqlStatement, string connectionName = "Default")
{
    using (IDbConnection cnn = new SQLiteConnection(DataAccessHelpers.LoadConnectionString(connectionName)))
    {
        cnn.Execute(sqlStatement);
    }
}

2 references
private static DynamicParameters ToDynamicParameters(this Dictionary<string, object> p)
{
    DynamicParameters output = new DynamicParameters();
    p.ToList().ForEach(x => output.Add(x.Key, x.Value));
    return output;

}
```

Zdroj: Vlastní zpracování (2021)

### 13.10.3 DataAccessHelpers

DataAccessHelpers třída (viz ukázka kódu 18) slouží pro připojení k databázi. Pokud není předaný parametr metoda automaticky předá default. `ConnectionString`

s názvem `Default` je zapsaný v souboru `App.config`. Zde je také potřeba po přesunu zdrojových souborů (na jiné zařízení nebo v rámci disku) nakonfigurovat správnou cestu k databázi.

*Ukázka kódu 18 DataAccessHelpers*

```
3 references
public static class DataAccessHelpers
{
    3 references
    internal static string LoadConnectionString(string id = "Default")
    {
        return ConfigurationManager.ConnectionStrings[id].ConnectionString;
    }
}
```

Zdroj: Vlastní zpracování (2021)

# 14 Uvedení aplikace ANDI do provozu

Během vývoje byl postup projektu komunikován s Mgr. Jónovou jakožto konečným uživatelem aplikace. Po dokončení implementace následovalo měsíční testovací období, kdy aplikace procházela drobnými úpravami (především uživatelského rozhraní) a laděním chyb. Během tohoto měsíce došlo k několika schůzkám, kde byla aplikace předvedena a Mgr. Jónová byla zaškolena.

Během těchto schůzek byla zpracována data, která byla během let 2019 a 2020 zaznamenána do vzorových formulářů (přílohy 1 a 2). Postupné zavedení práce s těmito formuláři do praxe ulehčilo přesun ke kvantifikovanému hodnocení během terapie, které je požadováno pro použití vyvinutého softwaru. Díky těmto dokumentům také vznikla databáze historických dat, takže hned v prvních týdnech používání již můžeme vidět využití grafické části aplikace.

Finální release verze softwaru pro záznam logopedické diagnostiky ANDI je přiložena na datovém disku.

V od března 2021je aplikace již aktivně využívána.

# 15 Závěr

Lidské slovo je neuvěřitelně mocným nástrojem a zároveň jedním ze základních kamenů lidské komunikace. Logopedie přináší pomoc a řešení těm, kteří mají v této klíčové oblasti problémy. Vytvořená aplikace ANDI má za cíl ulehčit logopedům práci se záznamem logopedické diagnostiky a archivování historických dat. Grafická vizualizace progresu klienta může také sloužit jako motivační nástroj.

Teoretická část této práce se zaměřuje na životní cyklus informačního systému a popis jeho jednotlivých částí. Čtenář je seznámen s postupem analýz a specifikací potřebných pro rozhodnutí, zda projekt vůbec spustit. Dále je popsán postup tvorby návrhů jednotlivých částí aplikace a teoretický úvod do tvorby WPF aplikace na platformě .Net Framework a s využitím XAML značkovacího jazyka a MVVM architektury. Poslední dvě kapitoly teoretické části se zabývají uvedením systému do provozu a následné údržby.

Praktická část práce je zaměřena na vývoj aplikace pro záznam logopedické diagnostiky, podle potřeb praxe Mgr. Violy Jónové. Nejdříve jsem musela proniknout do tajů logopedie a během několika schůzek se detailně seznámit s prací logopeda a požadavky na logopedickou diagnostiku. Společnými silami jsme se slečnou Jónovou vypracovaly seznam položek logopedické diagnostiky a požadavky na kartu klienta pro její osobní praxi. Na základě tohoto seznamu jsem vypracovala vzorové záznamové formuláře, které byly používány jako příprava před přesunem na digitalizovaný systém. Na základě těchto příprav jsem v průběhu roku 2020 naprogramovala WPF aplikaci a na jaře 2021 byla aplikace uvedena do provozu.

Tento projekt mě nesmírně obohatil, rozšířil mi obzory i v oblastech vzdálených mému oboru studia. Aplikace ANDI byla prvním velkým projektem, který jsem sama naprogramovala a moje schopnosti v této oblasti byly podrobeny mnoha zkouškám. Během práce jsem si ujasnila mé budoucí pracovní zaměření a získala mnoho cenných zkušeností ohledně dlouhodobé spolupráce.

Celkově projekt hodnotím jako úspěch a pokud se aplikace v následujících měsících osvědčí, nabídnu její použití i jiným logopedickým praxím, které o projekt ANDI projevily zájem.

# I. Summary and Key Words

Communication, especially speech, is one of the most important skills that people have. The main goal of my bachelor thesis is development of an application suitable for storing speech therapy results.

The first part of the paper is about software life cycle and theoretical step by step approach for software development.

The second part of the thesis describes speech therapy sessions, data coming out of examination by speech therapist and how process those datasets. It also speaks about development of an application called ANDI, which serves to speech therapists. This software helps store data about clients and their sessions. Program can also be used for long-term observation of client progress. Data are insert through electronic forms and outputs are presented in multiple charts.

Application was created for usage in professional practise of Mgr. Viola Jónová. Programming language C# and .NET technologies were used for development.

Key words: software development, speech therapy, C# .NET

# II. Seznam použitých zdrojů


Basl, J., & Blažíček, R. (2012). *Podnikové informační systémy: podnik v informační společnosti* (3., aktualizované a doplněné vydání). Grada Publishing.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications Of The Acm*, 13(6), 377-387. https://doi.org/362384.362685

Dvořák, J. (2001). *Logopedický slovník* (2. upr. a rozš. vyd). Žďár nad Sázavou: Logopedické centrum.

Harrington, J. L. (22nd April 2016). *Relational Database Design and Implementation* (Fourth Edition). Morgan Kaufmann.

Khare, S. (27-Jun-2019). *Learn Software Development Life Cycle: A series of planned activities to develop or alter the Software Products* (Kindle Edition). Amazon Media EU. https://read.amazon.co.uk/?asin=B07TLYG1JX

Křena, B., Kočí, R. (2006). *Úvod do softwarového inženýrství: IUS (Studijní opora)*. Brno: FIT VUT.

Lechta, V. (2003). *Diagnostika narušené komunikační schopnosti*. Praha: Portál.

LeMay, M. (2018). *Agile for Everybody: Creating Fast, Flexible, and Customer-First Organizations*. O'Reilly Media.

MacDonald, M. (2013). *Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5* (Fourth Edition). Apress.

Posadas, M. (2016). *Mastering C# and .NET Framework: Deep dive into C# and .NET architecture to build efficient, powerful applications*. Packt Publishing.

Voříšek, J. (2015). *Principy a modely řízení podnikové informatiky* (Vydání druhé). Oeconomica, nakladatelství VŠE.

Yuen, S. (2020). Mastering Windows Presentation Foundation: Build responsive UIs for desktop applications with WPF (Second Edition). Packt Publishing.

# III. Seznam obrázků

Obrázek 1 Integrace IS ..... 25
Obrázek 2 ER diagram databáze ANDI ..... 41
Obrázek 3 Logo ANDI ..... 44
Obrázek 4 Diagram tříd UI ..... 45
Obrázek 5 Class diagram tříd ViewModel ..... 47
Obrázek 6 Model třídy ..... 48

# IV. Seznam ukázek kódu

Ukázka kódu 1 DataTemplate ..... 45
Ukázka kódu 2 DataBinding ..... 46
Ukázka kódu 3 INotifyPropertyChanged ..... 47
Ukázka kódu 4 UpdateViewCommand ..... 49
Ukázka kódu 5 SearchCommand ..... 49
Ukázka kódu 6 PopulateKlientLists ..... 50
Ukázka kódu 7 InsertArtikulace ..... 51
Ukázka kódu 8 EditOsobniInfoCommand ..... 52
Ukázka kódu 9 GetZdravotniAnamneza ..... 52
Ukázka kódu 10 UpdateZdravotniAnamneza ..... 53
Ukázka kódu 11 DeleteKlient ..... 53
Ukázka kódu 12 SaveNewTerapie ..... 54
Ukázka kódu 13 GetDataPragmatickaRovina ..... 55
Ukázka kódu 14 GetDataPragmatickaRovina detail cyklu foreach ..... 56
Ukázka kódu 15 SelectedKlientSingleton ..... 56
Ukázka kódu 16 LoadData, SaveData ..... 57
Ukázka kódu 17 DeleteData, ToDynamicParameters ..... 57
Ukázka kódu 18 DataAccessHelpers ..... 58

# V. Seznam příloh

Příloha 1 Formulář klient ..... 66
Příloha 2 Formulář Logopedická diagnostika ..... 67
Příloha 3 Úvodní obrazovka ..... 68
Příloha 4 Obrazovka – Nový klient ..... 68
Příloha 5 Obrazovka – Osobní informace ..... 69
Příloha 6 Obrazovka – Terapie seznam ..... 69
Příloha 7 Obrazovka – Terapie detail ..... 70
Příloha 8 Obrazovka – Grafy ..... 70

# VI. Přílohy

*Příloha 1 Formulář klient*

| | | | |
|---|---|---|---|
| Jméno | | Příjmení | |
| Datum narození | | Přijetí do praxe | |
| Telefonní číslo | | Email | |
| Diagnóza | | | |
| Zájmy | | | |
| Poznámka | | | |

## Rodinná anamnéza

| | | | | | |
|---|---|---|---|---|---|
| Jméno | | Příjmení | | Povolání | |
| Jméno | | Příjmení | | Povolání | |
| Rodina | | Počet dětí | | Pořadí | |
| NKS v rodině | | | | | |
| Jazyková situace | | | | | |

## Osobní anamnéza

| | | | | | |
|---|---|---|---|---|---|
| Prenatální ob. | | Perinatální ob. | | Postnatální ob. | |
| Kojení | | Lahev | | Dudlík | |
| Motorický vývoj (otáčení/lezení/sed/chůze) | | | | | |
| Příjem potravy | | | Polykání | | |
| Spánek | | | Enuresis nocturna | | |
| Zuby | | | Čelist | | |
| Předškolní zař. | | | Škola | | |

## Zdravotní anamnéza

| | | | |
|---|---|---|---|
| Úrazy | | Operace, narkóza | |
| Nemocnost | | Medikace | |
| Specialisté | | Logopedie | |

Zdroj: Vlastní zpracování (2019)

*Příloha 2 Formulář Logopedická diagnostika*

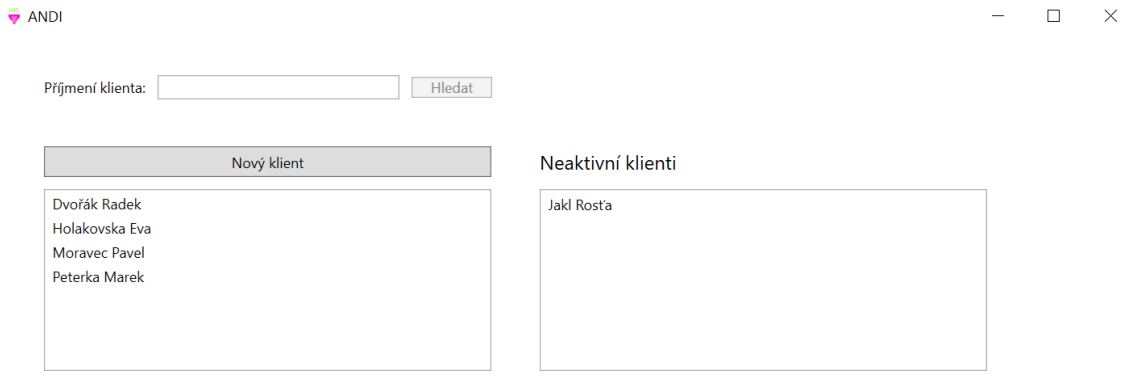
| | | | |
|---|---|---|---|
| Jméno | | Příjmení | |
| Datum narození | | Datum sezení | |
| Telefonní číslo | | Email | |
| Diagnóza | | | |
| Zájmy | | | |
| Poznámka | | | |

| Artikulace (doporučený věk) | | | | | |
|---|---|---|---|---|---|
| J (1 – 2,5) | | P, B, M (1 – 2,5) | | L (4,5 – 5) | |
| K, G (2,6 – 4) | | T, D, N (3,5 – 4) | | C, S, Z (5,3 – 6,5) | |
| H, CH (2,6 – 4) | | Ť, Ď, Ň (3,5 – 4,5) | | R (5,5 – 6,5) | |
| V, F (2,6 – 3,5) | | Č, Š, Ž (4,5 – 5,5) | | Ř (6 –7) | |
| BĚ, PĚ, VĚ, MĚ (3,6 – 4,5) | | | Č, Š, Ž, C, S, Z (6 – 7) | | |

| FONETICKO – FONOLOGICKÁ ROVINA | | LEXIKÁLNĚ – SÉMANTICKÁ ROVINA | | MORFOLOGICKO – SYNTAKTICKÁ ROVINA | |
|---|---|---|---|---|---|
| Sluchová analýza | | Popis obrázku | | Skloňování | |
| Sluchová syntéza | | Pojmenování | | Časování | |
| Znělost/ neznělost | | Porozumění | | Určování rodů | |
| Tvrdost/měkkost | | Serialita + vyprávění | | Užívání slovních druhů (se;si) | |
| První hláska ve slově | | Nad/Podřazené pojmy | | Tvorba vět | |
| Poslední hláska ve slově | | Antonyma | | Počítání | |
| Délka vokálu | | Synonyma | | Barvy | |
| | | Homonyma | | | |
| PRAGMATICKÁ ROVINA | | MOTORIKA | | OSTATNÍ | |
| Dialog | | Hrubá motorika | | Respirace | |
| Zrakový kontakt | | Jemná motorika | | Fonace | |
| Neverbální komunikace | | Úchop | | Rezonance | |
| Hra | | Přítlak | | Diadochokineza | |
| | | Úroveň kresby | | Spolupráce | |
| | | Motorika mluvidel | | Fluence | |
| | | | | Pozornost | |
| | | | | | |

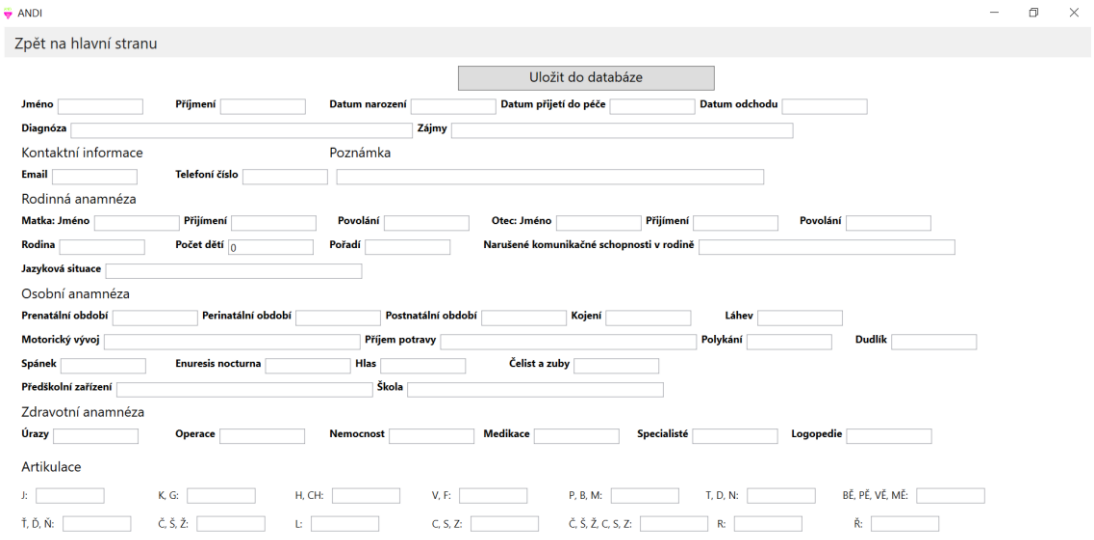
Zdroj: Vlastní zpracování (2019)

## *Příloha 3 Úvodní obrazovka*



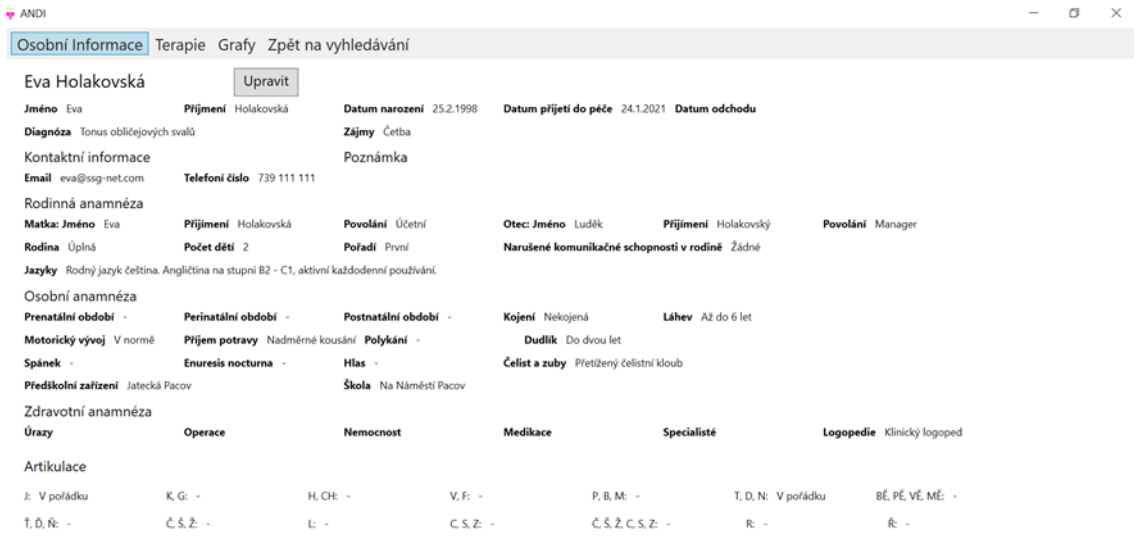

Zdroj: Vlastní zpracování (2021)

## *Příloha 4 Obrazovka – Nový klient*



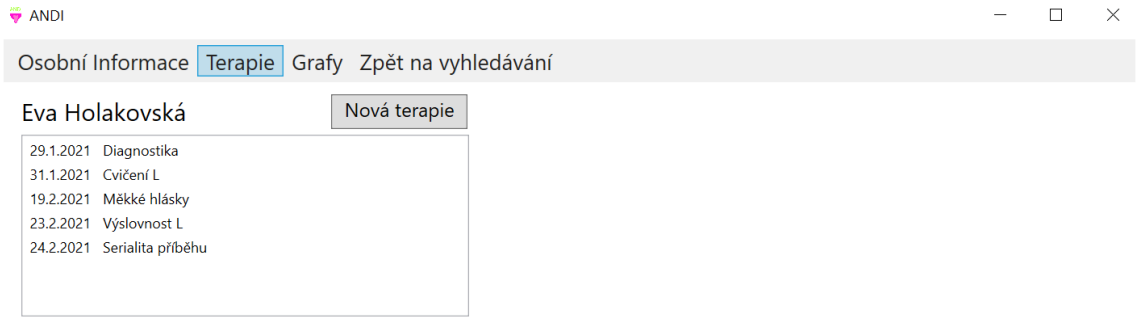

Zdroj: Vlastní zpracování (2021)

*Příloha 5 Obrazovka – Osobní informace*




Zdroj: Vlastní zpracování (2021)

*Příloha 6 Obrazovka – Terapie seznam*

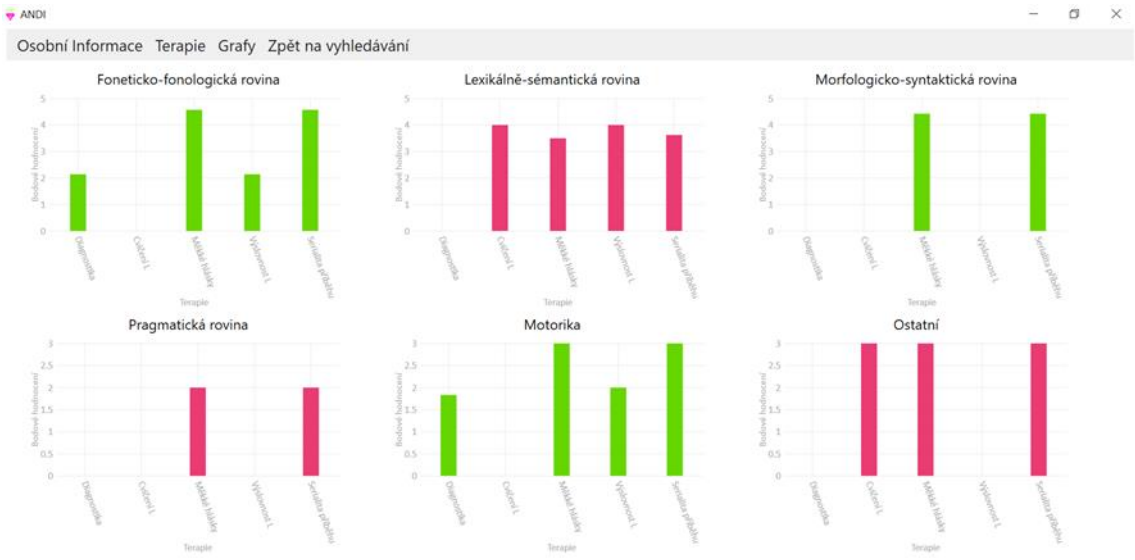



Zdroj: Vlastní zpracování (2021)

## *Příloha 7 Obrazovka – Terapie detail*


Zdroj: Vlastní zpracování (2021)

## *Příloha 8 Obrazovka – Grafy*




Zdroj: Vlastní zpracování (2021)