DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, PARDUBICE

MATURITNÍ PROJEKT

CLISYNC

Jméno a příjmení:	*Miroslav BOŘEK*
Pro školní rok:	*2024/2025*
Třída:	*4.*
Obor:	*Informační technologie 18-20-M/01*
Téma práce:	Clisync: Vytvoření multiplatformní mobilní aplikace a implementace AI funkcí
Vedoucí práce:	Tomáš Pešek

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem projektu bude vytvořit mobilní aplikaci za pomocí frameworku Expo pro službu Clisync, aplikace bude vytvořena za pomoci technologie WebView a pokud bude potřeba budou implementovány i nativní obrazovky. Úkolem bude také optimalizovat web aby byla zajištěna správná funkčnost na mobilním zařízení. Dále bude zajištěna distribuce do appstorů (android, ios) a AI funkce pro aplikaci (souhrn textového dokumentu, přepsání zápisků do lepší a čitelnější podoby, text to speech a speech to text)

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

09.2024 - Nastavení aplikace:

- Nastavení projektu Expo.
- Vytvoření WebView

10.2024 – Web a průzkum AI:

- Optimalizace a úpravy webu
- Výzkum AI modelů nejlepší pro použití v aplikaci.

11. 2024 - Průzkum a implementace AI:

- Implementace AI v aplikaci a na serveru.

12.2024 – Opravy chyb a distribuce:

- Testování a oprava zjištěných chyb.
- Optimalizace výkonu a vylepšení UI/UX.
- Distribuce na AppStore (ios, android).

Prohlašuji, že jsem maturitní projekt vypracoval samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 31.3.2025

Poděkování

Děkuji Ing. Tomášovi Peškovi za studijní materiál a odborné vedení při zpracovávání maturitního projektu. Byl jste skvělým mentorem, vaše rady a nápady mi pomohly rozvinout projekt správným směrem. Dále děkuji Jakubovi Kluckému a Denisovi Matsenko za vzájemnou pomoc při řešení všech komplikací a funkcí aplikace. Nakonec bych chtěl poděkovat rodině a přátelům za morální podporu během celého projektu. Bez jejich podpory by to nebylo možné.

Anotace

Tato závěrečná práce přináší pohled do vývoje mobilní aplikace a implementace AI funkcí pro zjednodušení používání. Dokumentace je rozdělena do jednotlivých kapitol, kde jsou popsány postupy při vývoji takové aplikace. Jako výstup této práce je aplikace pro management klientů s názvem Clisync.

Klíčová slova

AI, Expo, React Native, NextJS, OpenAI

Obsah

1. Úvod

Cílem tohoto projektu je vytvořit mobilní rozhraní pro aplikaci Clisync, která pomůže psychologům zefektivnit jejich administrativní práci a organizaci poznámek. Tradiční způsoby zaznamenávání informací o klientech často nejsou jednotné, a mnoho psychologů uchovává data lokálně bez zálohování, což může vést k jejich ztrátě nebo neorganizovanosti. Tento projekt si klade za cíl nabídnout digitální řešení, které umožní snadnou správu zápisků a zároveň zajistí vysokou úroveň zabezpečení.

Dalším cílem je integrace umělé inteligence pro zpracování textu, která umožňuje sumarizaci zápisků, úpravy textu a jeho převod na řeč. Implementace AI se zaměřuje na praktické využití při práci s textem a je navržena tak, aby nenarušovala soukromí uživatelů. Z toho důvodu byla původně plánovaná implementace self-hosted AI řešení, které by minimalizovalo přenos citlivých údajů mimo kontrolované aplikační prostředí. Tento přístup byl zvolen s cílem zajistit maximální kontrolu nad zpracováním dat a snížit závislost na externích službách, což je klíčové pro důvěryhodnost aplikace v profesionálním prostředí zdravotní péče.

V teoretické části se závěrečná práce zaměřuje na výběr technologie pro mobilní aplikaci, kde jsou porovnány různé přístupy k vývoji a architektury aplikace. Součástí teoretické analýzy je také výběr AI modelu, přičemž jsou zhodnoceny dostupné možnosti, včetně hostovaných a self-hosted řešení, a zohledněna kritéria jako výkon, bezpečnost a kompatibilita. Nakonec se závěrečná práce věnuje samotné implementaci AI funkcí, kde jsou rozebrány způsoby integrace modelu do aplikace a jeho praktické využití pro zpracování textu.

2. Použité technologie ve společné části

Tato část pojednává o využitých technologiích ve společné části závěrečné práce, tzn. webové aplikaci, která se zároveň využívá i v mobilní aplikaci pomocí technologie webview.

2.1 Visual studio code

Visual Studio Code [1] je open-source editor pro úpravu kódu vyvinutý společností Microsoft, dostupný pro operační systémy Windows, mac OS a Linux. Jedná se o výkonný a flexibilní nástroj, který usnadňuje tvorbu, úpravu a správu kódu. Podporuje mnoho programovacích jazyků, čímž se stával univerzálním řešením. Jednou z jeho klíčových vlastností je IntelliSense, což je funkce automatického doplňování kódu, která nabízí chytré návrhy na základě analýzy projektu. Kromě toho obsahuje integrovaný terminál, který umožňuje spouštět příkazy přímo z editoru, a vestavěné nástroje pro ladění, které pomáhají identifikovat a řešit chyby přímo v kódu. Díky podpoře rozšíření lze přidávat nové funkce, například lintování, a podporu pro specifické technologie a frameworky. VS Code je rovněž dobře integrován s Gitem GitHubem, což usnadňuje spolupráci v týmu, která byla pro náš projekt potřeba.

2.2 Miro

Miro [2] je online platforma pro vizuální spolupráci, která umožňuje týmům pracovat společně v reálném čase nebo asynchronně. Je známá především jako digitální whiteboard, který nabízí širokou škálu nástrojů pro brainstorming, plánování, designování a řízení projektů. Uživatelé mohou vytvářet diagramy, mapy myšlenek, vývojové diagramy, wireframy a mnoho více. Díky intuitivnímu rozhraní a snadnému sdílení umožňuje spolupráci, a to i na dálku. Aplikace je využívána například vývojáři nebo designery. Její klíčovou vlastností je neomezené plátno, na kterém lze vizualizovat složité nápady, strukturovat informace a propojit různé formáty obsahu, což podporuje kreativitu a týmovou produktivitu.

2.3 React

React [3] je open-source JavaScriptová knihovna vyvinutá společností Meta pro tvorbu uživatelských rozhraní. Umožňuje jednoduše a efektivně vytvářet webové aplikace. Především jsem potřeboval nástroj, který umožní vytvářet uživatelské rozhraní efektivně, flexibilně a škálovatelně. React nabízí komponentový přístup, což umožňuje rozdělit aplikaci na menší, znovupoužitelné části, díky tomu je vývoj nejen rychlejší, ale i jednodušeji udržitelný přehledně.

Dalším důvodem je Virtuální DOM [4], který zajišťuje výkon aplikace. React provádí aktualizace uživatelského rozhraní pouze tam, kde došlo ke změnám, což je pro moji aplikaci, která zpracovává obsah dynamicky, jako například můj kalendář nebo seznam klientů. Deklarativní přístup Reactu pak usnadňuje definici toho, jak má uživatelské rozhraní vypadat v různých stavech, což je ideální pro práci s komplexními daty a různými scénáři interakce uživatelů.

Jeden z hlavních důvodů proč jsem využil react jsou react hooky Díky hookům, jako je useState nebo useEffect, jsem mohl pracovat se stavem a vedlejšími efekty aplikace jednoduše a bez nutnosti psát třídy, což zrychluje a zpříjemňuje vývoj.

2.4 NextJS

Next.js [5] je framework postavený nad Reactem, který rozšiřuje možnosti vývoje mé webové aplikace. Nabízí Server-Side Rendering (SSR) a Static Site Generation (SSG) [6], což vede ke zrychlení načítání stránek a lepší SEO optimalizaci. Například dynamické stránky, jako je seznam klientů nebo kalendář, mohou být vykresleny rychleji a vyhledávače je snadněji indexují. Automatický routing na základě struktury souborů v projektu zjednodušuje správu aplikací, zatímco API Routes umožňuje vytvářet backendové funkce přímo v aplikaci bez potřeby samostatného serveru.

Next.js také nabízí Image Optimization, která automaticky optimalizuje velikost a formát obrázků, čímž zlepšuje výkon aplikace a uživatelskou zkušenost, zejména na mobilních zařízeních. Kromě toho obsahuje Middleware, které umožňuje provádět operace nad dotazy a odpověďmi ještě předtím, než jsou zpracovány ve stránkách aplikace. To je užitečné například pro autentifikaci nebo přesměrování uživatelů na základě jejich stavu.

Další významnou funkcí Next.js je Incremental Static Regeneration (ISR), což umožňuje aktualizovat staticky generované stránky bez nutnosti rebuildu celé aplikace. Díky tomu mohou být statické stránky průběžně aktualizovány novými daty, aniž by bylo nutné čekat na kompletní nasazení nové verze webu.

Next.js podporuje také Edge Functions, které umožňují běh serverových funkcí přímo na okraji sítě (Edge Network), což vede k extrémně nízké latenci a rychlému zpracování požadavků. To je ideální například pro personalizaci obsahu nebo A/B testování.

Framework také nabízí Built-in Internationalization (i18n), což usnadňuje správu vícejazyčných aplikací bez nutnosti externích knihoven. Vývojářům poskytuje i Automatic Code Splitting, čímž minimalizuje velikost načtených JavaScriptových souborů a zlepšuje výkon.

Protože je postavený na Reactu, poskytuje všechny jeho výhody, včetně hooků, komponentního přístupu a Virtuálního DOMu, ale přidává pokročilé funkce bez nutnosti složité konfigurace. Navíc má built-in podporu pro TypeScript, což zjednodušuje vývoj a zlepšuje stabilitu kódu. Podporuje také moderní nástroje jako Webpack 5, SWC pro rychlou kompilaci a Rust-based minifikaci kódu.

Použití Next.js je ideální volbou, pokud hledáte robustní řešení pro vývoj rychlých, škálovatelných a optimalizovaných aplikací s vysokým výkonem a výbornou podporou pro SEO.

2.5 Server side rendering

Server-Side Rendering [6] přináší několik zásadních výhod, zejména pro webové aplikace, které vyžadují vysoký výkon, rychlé načítání obsahu a optimalizaci pro vyhledávače. SSR předrenderuje obsah na serveru, takže uživatel dostane kompletní HTML stránku ještě před načtením JavaScriptu. To znamená, že uživatelé uvidí obsah téměř okamžitě, což zlepšuje uživatelskou zkušenost. Je to zvláště výhodné pro aplikace s obsahem, který se často mění, jako jsou e-shopy, zpravodajské portály nebo dynamické tabulky dat. Dalším důvodem pro použití SSR je lepší SEO. Vyhledávače upřednostňují weby, které servírují plný obsah ve formě HTML, což usnadňuje indexaci a zlepšuje viditelnost webu. SSR také umožňuje generovat personalizované stránky přímo na serveru, například zobrazení uživatelsky specifických dat nebo přizpůsobení obsahu na základě lokalizace či zařízení. Další výhodou

je lepší výkon na pomalejších zařízeních, protože většina zátěže je přesunuta na server. SSR lze navíc kombinovat s dalšími strategiemi, jako je Static Site Generation pro statické stránky nebo Client-Side Rendering pro interaktivní prvky. Tato flexibilita umožňuje optimalizovat aplikaci pro různé scénáře, například pro real-time aktualizace obsahu, kdy server načte aktuální data při každém požadavku, a uživatel tak vždy dostane nejnovější verzi. SSR je vhodné, pokud potřebujete lepší SEO, pracujete s dynamickým obsahem, chcete snížit dobu načítání stránky nebo podporovat méně výkonná zařízení.

2.6 Vercel

Vercel [7] je cloudová platforma pro nasazování a škálování moderních webových aplikací. Zaměřuje se na rychlost, jednoduchost a podporu vývojářů

Používání platformy Vercel přináší řadu výhod, které usnadňují vývoj a nasazení webových aplikací, zejména těch postavených na frameworku Next.js. Vercel je navržen tak, aby vývojáři mohli rychle nasazovat své projekty bez složité konfigurace serverů nebo infrastruktury. Jednou z hlavních výhod je možnost nastavení a automatického nasazení z GitHub repozitářů. Při každé změně kódu je aplikace automaticky nasazena na produkční i testovací prostředí. Dalším důvodem je vysoký výkon a optimalizace pro koncové uživatele. Vercel poskytuje globální síť CDN, která zajišťuje rychlé načítání stránek po celém světě. Platforma také nativně podporuje server-side rendering a static site generation, což je perfektní kombinace pro aplikace založené na Next.js. Navíc nabízí funkce jako edge middleware, které umožňují přizpůsobení obsahu přímo na okraji sítě, čímž se zlepšuje uživatelská zkušenost. Velkou výhodou je také jednoduchost a přívětivé rozhraní, které umožňuje monitorovat stav nasazení, výkon aplikace nebo správu domén. Vercel navíc podporuje týmovou spolupráci, což znamená, že více vývojářů může pracovat na jednom projektu efektivněji. Pro mě je Vercel ideálním nástrojem díky své jednoduchosti, výkonu a bezproblémové integraci s moderními webovými technologiemi.

2.7 Tailwind CSS

Tailwind CSS [8] je framework na stylování aplikací který používá utility přístup. Místo psaní vlastních tříd umožňuje používat předdefinované utility classes přímo v html. Tyto třídy reprezentují konkrétní stylové vlastnosti, jako jsou barvy, mezery, velikosti fontů, zarovnání

nebo stíny. Tailwind také obsahuje nástroj pro optimalizaci, který zajišťuje, že ve výsledném balíčku jsou zahrnuty pouze třídy, které reálně používáte, což zlepšuje výkon aplikace.

2.8 Daisy UI

Daisy UI [9] je plugin který poskytuje vlastní komponenty vytvořené pomocí frameworku Tailwind CSS tím zajišťuje možnost znovu použít jednotlivé komponenty a zároveň zachová možnost plně upravit přidáním vlastních Tailwind CSS tagů. Toto bylo klíčové, aby k tvorbě aplikace nebyl potřeba design, ale mohl jsem ji vytvořit sám bez bližších znalostí grafického designu.

3. Mobilní aplikace

3.1 Výběr technologie

Při vývoji mobilní aplikace je klíčové zvolit vhodnou technologii, která umožní efektivní vývoj, údržbu a rozšiřitelnost aplikace. Každý přístup má své výhody a nevýhody, které ovlivňují výkon, vývojovou náročnost a kompatibilitu s různými platformami. V této kapitole budou představeny možnosti mezi kterými se rozhodovalo a jejich porovnání.

3.1.1 Nativní aplikace

Nativní aplikace [10] jsou vyvíjeny specificky pro jednotlivé platformy, například pomocí **Swift** pro iOS a **Kotlin/Java** pro Android. Tento přístup zajišťuje nejlepší výkon, hlubokou integraci s hardwarem a optimalizované uživatelské prostředí.

Výhody:

- Nejvyšší výkon a plynulost
- Plná podpora nativních API a funkcí zařízení
- Možnost využití nejnovějších funkcí operačního systému

Nevýhody:

- Vyšší náklady na vývoj (nutnost dvou verzí aplikace)
- Dlouhá vývojová doba
- Náročnější údržba dvou kódových bází

3.1.2 React Native

React Native [11] je framework vyvinutý společností **Meta** (Facebook) v roce 2015. Je založen na **JavaScriptu a Reactu**, což umožňuje vývoj mobilních aplikací s využitím webových technologií. Kód je většinou sdílený mezi platformami, ale některé nativní prvky je nutné implementovat zvlášť.

Výhody:

- Možnost sdílení kódu mezi iOS a Androidem
- Široká komunita a podpora knihoven
- Snadná integrace s webovými technologiemi

Nevýhody:

- Nižší výkon oproti nativním aplikacím
- Složitější podpora některých nativních funkcí
- Čas od času problémy s kompatibilitou knihoven

3.1.3 Flutter

Flutter [12] je framework vyvinutý **Googlem** v roce 2017. Používá programovací jazyk **Dart** a vlastní vykreslovací engine, což umožňuje jednotný vzhled aplikací napříč platformami. Flutter generuje nativní kód, což zajišťuje lepší výkon než React Native.

Výhody:

- Vysoký výkon díky vlastním grafickým komponentám
- Stejný vzhled na všech platformách
- Rychlý vývoj díky Hot Reload

Nevýhody:

- Menší komunita než React Native
- Větší velikost aplikace
- Použití méně rozšířeného jazyka Dart

3.1.4 Expo

Expo [12] je framework postavený na **React Native**, který usnadňuje vývoj mobilních aplikací odstraněním složité konfigurace. Poskytuje předpřipravené API pro běžné funkce, jako jsou kamery, push notifikace a správa souborů.

Výhody:

- Rychlé nasazení a snadná údržba
- Možnost práce bez potřeby nativního vývojového prostředí
- Integrované nástroje pro testování a publikaci

Nevýhody:

- Omezenější přístup k některým nativním funkcím (nutnost od-ejectování)
- Menší flexibilita oproti čistému React Native
- Závislost na ekosystému Expo

Expo bylo zvoleno pro tento projekt z několika důvodů:

- **Zjednodušení vývoje** – umožňuje rychlé prototypování bez potřeby konfigurace nativního kódu
- **Podpora webview** – snadná integrace s webovou aplikací Clisync
- **Kompatibilita** – snadná distribuce aplikace přes Expo Go bez nutnosti sestavení APK/IPA při testování

3.2 Způsob Implementace

Při vývoji mobilní aplikace existuje několik přístupů k implementaci uživatelského rozhraní a logiky. Každý způsob má své výhody a nevýhody v závislosti na požadavcích na výkon, flexibilitu a vývojovou rychlost. V této části jsou popsány dva hlavní způsoby – klasická aplikace a WebView – a jejich porovnání z hlediska použitelnosti v projektu Clisync.

3.2.1 Klasická aplikace

Klasická mobilní aplikace znamená, že veškerá funkcionalita, uživatelské rozhraní a logika běží přímo v rámci nativního nebo hybridního kódu aplikace. Data mohou být získávána přes API, ale samotné vykreslování, navigace i interakce se uživatelem probíhají v prostředí mobilní aplikace.

Použití:

- Nativní aplikace (Swift pro iOS, Kotlin/Java pro Android)
- Hybridní aplikace (React Native, Flutter)
- Aplikace, které vyžadují hlubokou integraci s hardwarem zařízení (např. kamera, Bluetooth)

Výhody:

- Maximální výkon a optimalizace pro konkrétní platformu
- Možnost přímé interakce s nativními API a hardwarem zařízení
- Lepší uživatelský zážitek díky hladší animaci a plynulejší odezvě

Nevýhody:

- Náročnější vývoj a údržba – nutnost spravovat zvlášť verze pro iOS a Android, pokud není použit hybridní framework
- Delší vývojový cyklus a vyšší náklady
- Složitější publikace aktualizací – změny často vyžadují nový build a schválení na App Store nebo Google Play

3.2.2 WebView

WebView [13] je komponenta, která umožňuje zobrazování webových stránek uvnitř mobilní aplikace. Aplikace tedy funguje jako obal (shell) pro webovou aplikaci, která běží v prostředí mobilního prohlížeče integrovaného do aplikace. Tato technologie byla vybrána pro projekt Clisync, protože umožňuje sdílení kódu mezi webovou a mobilní verzí aplikace a usnadňuje vývoj a údržbu.

Použití:

- Mobilní aplikace zobrazující existující webovou platformu
- PWA (Progressive Web Apps) v mobilním prostředí
- Aplikace, které nevyžadují složité nativní interakce

Výhody:

- **Rychlejší vývoj** – aplikace využívá existující webovou platformu, není nutné vytvářet samostatné UI pro mobilní zařízení
- **Jednotný kód pro více platforem** – změny ve webové aplikaci se automaticky projeví i v mobilní aplikaci
- **Jednodušší aktualizace** – většinu změn lze provádět na straně serveru, bez nutnosti aktualizace aplikace v App Store nebo Google Play
- **Snížení nákladů na vývoj** – odpadá nutnost vytvářet samostatnou nativní aplikaci

Nevýhody:

- **Nižší výkon oproti nativním aplikacím** – WebView není tak optimalizované jako nativní UI komponenty
- **Omezený přístup k nativním funkcím** – některé pokročilé funkce zařízení mohou vyžadovat speciální mosty mezi WebView a nativním kódem
- **Závislost na dostupnosti internetu** – pokud webová aplikace není navržena pro offline režim, nemusí být aplikace funkční bez připojení

Pro tento projekt byla zvolena technologie WebView, protože umožňuje efektivní vývoj s minimální duplikací kódu mezi webovou a mobilní verzí aplikace Clisync. Díky frameworku Expo je implementace WebView jednoduchá a umožňuje budoucí rozšiřování funkcionality podle potřeb uživatelů.

3.3 App Store a Google Play

Publikace mobilní aplikace do oficiálních obchodů App Store (iOS) a Google Play (Android) je proces, který podléhá přísným pravidlům a schvalovacím procesům. Přestože distribuce aplikace přes tyto platformy poskytuje největší dosah k uživatelům, může být tento proces komplikovaný a časově náročný.

3.3.1 Požadavky na App Store (Apple)

Apple má velmi přísná pravidla pro aplikace distribuované přes App Store [14]. Aplikace musí splňovat jak technické, tak obsahové požadavky, jinak může být zamítnuta. Mezi hlavní problémy při vydávání patří:

- **Nedostatečná funkcionalita** – Apple zamítá aplikace, které považuje za neúplné, příliš jednoduché nebo sloužící pouze jako obal pro webovou stránku. WebView aplikace jsou často označovány jako „minimálně užitečné".
- **Omezené používání WebView** – Aplikace, které se spoléhají na WebView bez přidané nativní funkcionality, mohou být odmítnuty. Apple preferuje aplikace s plnohodnotným nativním zážitkem.
- **Chybějící metody přihlášení** – Apple vyžaduje, aby aplikace podporovaly „Sign in with Apple", pokud nabízejí jiné metody přihlášení (Google, Facebook).
- **Platební systém** – Aplikace nabízející placené služby nebo předplatné musí používat Apple In-App Purchases, což znamená vyšší poplatky (až 30 %).
- **Bezpečnostní a výkonové požadavky** – Aplikace musí být stabilní, nesmí nadměrně využívat systémové prostředky a musí odpovídat bezpečnostním standardům.

3.3.2 Požadavky na Play Store (Google)

Google Play je obecně méně přísný než App Store [15], ale stále má určité požadavky, které je třeba splnit. Hlavní problémy při schvalování aplikace zahrnují:

- **Kvalita aplikace** – Google kontroluje, zda aplikace funguje bez chyb a neobsahuje nekompletní nebo testovací funkce.
- **Obsahové zásady** – Aplikace nesmí obsahovat zakázaný obsah (např. dezinformace, podvodné chování, citlivá data uživatelů bez souhlasu).
- **Použití WebView** – Podobně jako Apple, Google může odmítnout aplikace, které pouze zobrazují webovou stránku bez přidané hodnoty.
- **Zabezpečení a ochrana osobních údajů** – Aplikace musí dodržovat zásady ochrany soukromí, jako je správné nakládání s uživatelskými daty a požadování pouze nezbytných oprávnění.

3.3.3 Důvody neúspěšné publikace aplikace

V případě této aplikace nebylo možné vydání na App Store ani Google Play kvůli nesplnění některých klíčových požadavků:

- **WebView aplikace bez nativní funkcionality** – Hlavním důvodem zamítnutí je, že aplikace funguje jako pouhý obal pro webovou aplikaci Clisync bez dalších nativních funkcí. To nesplňuje požadavky App Store ani Google Play.
- **Nedostatek přidané hodnoty** – Obchody vyžadují, aby aplikace nabízela něco navíc oproti pouhé webové verzi. To může být například offline režim, notifikace, integrace s nativními funkcemi zařízení apod.
- **Chybějící podpora pro nativní přihlášení** – Apple vyžaduje podporu „Sign in with Apple“, což aplikace zatím neimplementuje.
- **Omezené testování a stabilita** – Vydání aplikace vyžaduje důkladné testování na různých zařízeních a verzích operačních systémů, což zatím nebylo plně dokončeno.

3.3.4 Alternativní způsoby distribuce

Vzhledem k problémům s publikací do oficiálních obchodů lze zvážit alternativní způsoby distribuce aplikace:

- **Expo Go** – Aplikace může být testována a používána přes Expo Go, což umožňuje jednoduchou distribuci bez nutnosti schvalování.
- **PWA (Progressive Web App)** – Webová aplikace Clisync může být upravena na PWA, což umožní uživatelům ji přidat na domovskou obrazovku bez nutnosti stahování z obchodu.
- **Přímá distribuce APK (Android)** – Uživatelé Androidu si mohou aplikaci stáhnout a nainstalovat přímo z webové stránky.
- **Enterprise distribuce (iOS)** – Pro iOS lze využít podnikové certifikáty pro distribuci mimo App Store.

4. AI ve webové aplikace

V rámci projektu Clisync byla implementována umělá inteligence (AI), která slouží k úpravě textů zápisů psychologů. AI zajišťuje efektivní zpracování textových informací, které jsou následně upravovány a vylepšovány podle specifických potřeb uživatelů. V této části dokumentace jsou popsány klíčové aspekty výběru modelu, poskytovatelé AI a možnosti implementace, včetně bezpečnostních opatření.

4.1 Výběr modelu

Při výběru modelu pro AI bylo třeba zvážit několik faktorů [16], které ovlivňují jeho výkonnost, přístupnost a integraci do aplikace Clisync. Důležité je nejen vybrat správný typ modelu, ale také rozhodnout o jeho poskytovateli, metodě nasazení a zajištění bezpečnosti.

4.1.1 Důležitá kritéria

Při výběru modelu pro úpravy textu byly zohledněny následující kritéria:

- **Přesnost a kvalita generovaných textů** – Model musí být schopen správně analyzovat a upravit zápisy, přičemž by měl dodržovat správnou gramatiku a zachovat význam textu.
- **Rychlost a výkon** – Pro efektivní používání v reálném čase je nezbytné, aby model generoval texty rychle, bez výrazných prodlev.
- **Možnosti přizpůsobení** – Model by měl umožnit přizpůsobení specifickým potřebám aplikace Clisync, zejména v oblasti terminologie používané psychology.
- **Náklady na používání** – Finanční náklady na použití modelu mohou být rozhodujícím faktorem, zejména pokud se aplikace používá ve větším měřítku.
- **Kompatibilita s API** – Model by měl být snadno integrovatelný s webovou aplikací běžící ve WebView, což zajišťuje jednoduchou komunikaci mezi aplikací a AI.

4.1.2 Poskytovatelé a možnosti

Existuje několik poskytovatelů AI, kteří nabízejí modely pro úpravu textu, generování přirozeného jazyka a další pokročilé funkce. Mezi hlavní možnosti patří:

- **OpenAI** – Poskytovatel, který nabízí pokročilé modely pro generování textu jako GPT-4, které jsou schopné nejen opravovat gramatiku, ale také přeformulovat a vylepšit texty na základě specifických požadavků.
- **Google Cloud AI** – Nabízí různé modely pro analýzu a generování textu, které lze přizpůsobit specifickým potřebám aplikace, včetně podpory pro různé jazyky.
- **Microsoft Azure Cognitive Services** – Poskytuje nástroje pro zpracování textu, které zahrnují opravy gramatiky, analýzu sentimentu a další funkce, které mohou být užitečné pro aplikaci zaměřenou na zpracování zápisů.

Každý z těchto poskytovatelů nabízí různé úrovně přizpůsobení, nákladů a podporovaných funkcí, což ovlivňuje rozhodování o tom, který model bude pro aplikaci Clisync nejvhodnější.

4.2 Self-hosting

Další možností je **self-hosting** AI modelu [17], což znamená nasazení modelu na vlastním serveru. Tento přístup má několik výhod:

- **Kontrola nad daty** – Umožňuje plnou kontrolu nad tím, jak jsou data zpracována a ukládána, což je důležité pro zajištění ochrany soukromí uživatelů.
- **Flexibilita** – Self-hosting může poskytnout větší flexibilitu v oblasti přizpůsobení modelu specifickým potřebám aplikace.
- **Nižší dlouhodobé náklady** – Při větším objemu zpracovávaných dat mohou být náklady na používání externího poskytovatele AI vyšší než náklady na provoz vlastního serveru.

4.2.1 Otevřené AI modely

Pro self-hosting AI modelu je ideální platforma **Hugging Face** [18], která nabízí širokou škálu před trénovaných modelů pro různé úkoly strojového učení. Pro výběr správného modelu pro práci s textem je nutné zhodnotit mnoho aspektů:

- **Účel modelu** – Určit, jaký konkrétní úkol má model vykonávat (např. sumarizace textu, generování textu, analýza sentimentu, oprava gramatiky apod.).
- **Jazyková podpora** – Zkontrolovat, zda model podporuje jazyk(y), ve kterých bude aplikace fungovat (např. čeština, angličtina).
- **Velikost modelu** – Vybrat model s vhodnou velikostí, která odpovídá požadavkům na výkon a paměťové nároky aplikace. Větší modely obvykle poskytují lepší přesnost, ale vyžadují více výpočetních prostředků. Velikost modelu je definovaná celkovým počtem parametrů pomocí kterách byl natrénován. Menší modely se pohybují ve stovkách milionů, větší kompetentnější v řádů stovek miliard parametrů.
- **Přesnost a kvalita výsledků** – Posoudit, jak kvalitní a přesné jsou výsledky modelu při provádění požadovaných úkolů (např. jak dobře model generuje nebo opravuje text).
- **Licenční podmínky** – Zajistit, že licenční podmínky modelu dovolují jeho použití v komerční aplikaci, pokud bude aplikace určena pro širokou veřejnost nebo platící uživatele.
- **Efektivita využití zdrojů** – Posoudit, jak náročný je model na výpočetní zdroje, jako je CPU, RAM, a další hardware, aby byl provoz na serveru nebo mobilním zařízení udržitelný.

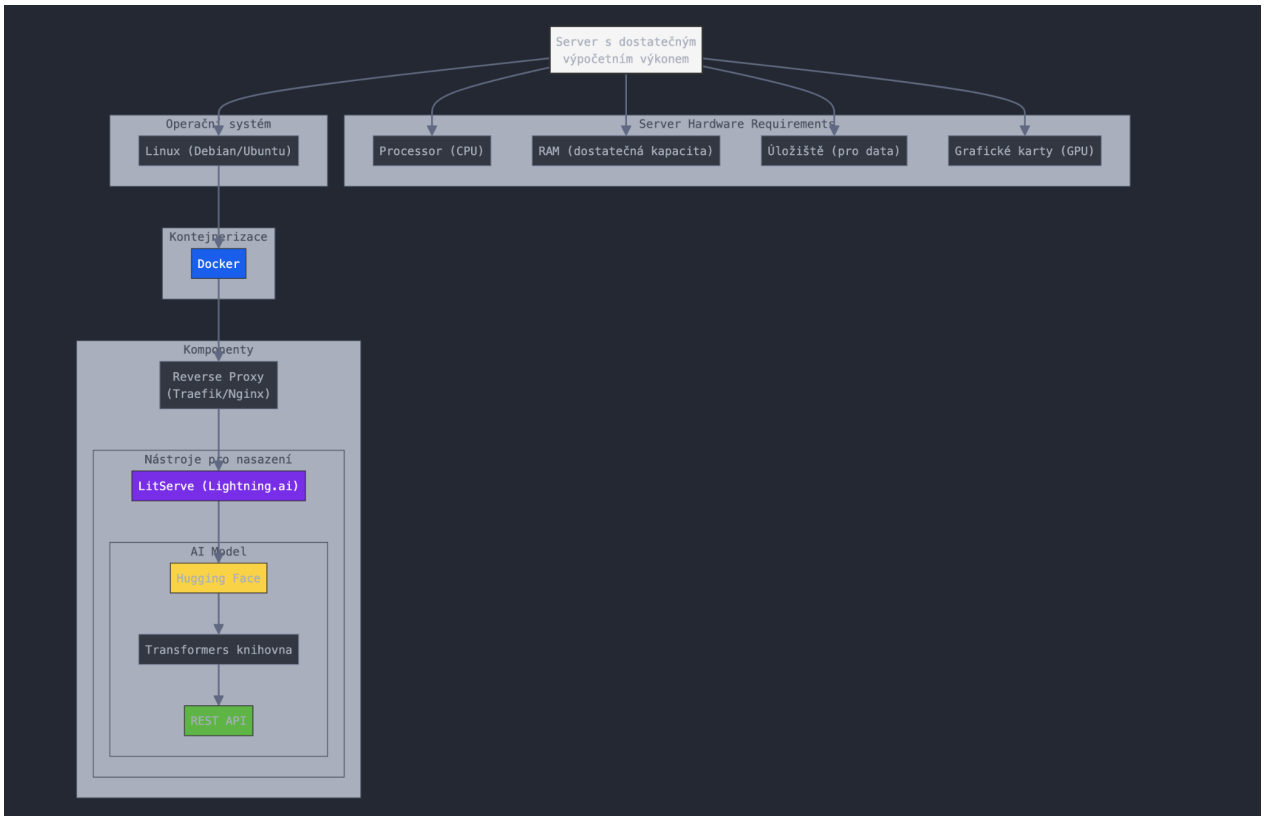
4.2.2 Infrastruktura

Nasazení AI modelu na vlastní hardware představuje náročný úkol, který vyžaduje pečlivé plánování a dostatečně silnou infrastrukturu. Celkové nasazení modelu je velmi komplexní, neboť zahrnuje nejen výběr vhodného hardware, ale i konfiguraci serverů a správu softwarového prostředí. Pro efektivní fungování AI modelů je nezbytné zajistit dostatečný výpočetní výkon. To zahrnuje silný processor, dostatek RAM, protože modely jsou nahrávány do paměti, aby mohly být okamžitě použity, a v případě náročnějších modelů je třeba také velkého úložiště pro ukládání dat. Kromě toho, pro některé modely je nutné vybavení s grafickými kartami (GPU), které zajišťují rychlé zpracování a zvyšují výkon při trénování a inferenci modelů.

Jednodušší varianta nasazení modelu může vypadat následovně: na serveru s dostatečným hardwarem běží linuxový operační systém, jako je například Debian nebo Ubuntu, který je

základem pro nasazení aplikace. Na tomto serveru je spuštěn Docker, což umožňuje snadnou správu a izolaci jednotlivých komponent. V rámci Dockeru běží reverse proxy server (např. Traefik nebo Nginx), který se stará o směrování požadavků mezi různými službami, a následně samotný kontejner, který obsahuje instanci AI modelu. Pro hostování AI modelu z platformy **Hugging Face** [18] lze využít jejich knihovnu **Transformers**, která umožňuje snadné nasazení před trénovaných modelů a jejich integraci do aplikace.

Pro efektivní nasazení, správu a škálování AI modelů v rámci kontejnerů je možné využít open-source projekt **LitServe** od **Lightning.ai** [19], který je optimalizován pro rychlou implementaci a škálovatelnost modelů v produkčním prostředí. **LitServe** umožňuje automatizované nasazení modelů v Docker kontejnerech a jejich správu, čímž výrazně usnadňuje nasazení a umožňuje efektivní škálování podle požadavků na výkon. Tento přístup zajišťuje nejen flexibilitu a snadnou správu, ale i efektivní využití výpočetních zdrojů. **LitServe poskytuje REST API bez složité konfigurace**, což výrazně zjednodušuje integraci a komunikaci s AI modelem v aplikacích.



Obrázek 1 Diagram infrastruktury

4.3 Bezpečnost

Bezpečnost je klíčovým faktorem při práci s AI modely, zejména v případě citlivých informací, jakými jsou zápisy psychologů. Hlavní bezpečnostní aspekty zahrnují:

- **Šifrování dat** – Všechna data přenášená mezi klientem a AI serverem by měla být šifrována, aby se zajistila ochrana před neoprávněným přístupem.
- **Zabezpečení uživatelských informací** – Infrastruktura by měla být navržena tak, aby se nezpracovávaly citlivé osobní údaje bez souhlasu uživatele.
- **Ochrana před zneužitím** – Je nutné implementovat mechanismy, které zamezí zneužití modelu pro generování nevhodného nebo škodlivého obsahu.
- **Soulad s regulacemi** – Je třeba zajistit, aby použití AI modelu splňoval všechny právní požadavky týkající se ochrany osobních údajů, jako je GDPR v EU nebo HIPAA ve Spojených státech.

Bezpečnostní opatření je nezbytné pečlivě naplánovat a implementovat, aby byla aplikace v souladu s nejvyššími standardy ochrany dat.

4.4 Implementace

Tato část dokumentace popisuje, jak byly implementovány různé AI funkce v aplikaci Clisync, které psychologům usnadňují správu a práci s jejich zápisy. Funkcionality zahrnují souhrn dokumentu a přepis zápisků. Převod textu na řeč (Text to Speech) a převod řeči na text (Speech to Text) zatím nejsou implementovány.

V aplikaci je využito OpenAI API, jelikož vlastního self-hostingu AI modelu nebylo možné dosáhnout. Implementované funkce tedy slouží jako showcase možností, které umělá inteligence nabízí v kontextu práce s textovými záznamy psychologů.

4.4.1 Souhrn dokumentu

Souhrn dokumentu v aplikaci Clisync slouží k automatickému zkrácení a zvýraznění klíčových informací v zápisech psychologa. Tato funkce využívá OpenAI API pro analýzu textu a generování stručného přehledu hlavních bodů.

Výhody implementace:

- Rychlé získání přehledu o obsahu delších zápisů
- Zlepšení organizace a archivace poznámek
- Možnost zpětného nahlédnutí bez nutnosti číst celý text

Souhrn dokumentu je navržen tak, aby nezasahoval do původního textu, ale nabídl přehledné shrnutí na základě AI analýzy.

4.4.2 Přepis zápisků

Přepis zápisků je klíčovou funkcí aplikace, která umožňuje psychologům upravovat a přeformulovávat své poznámky. OpenAI API provádí analýzu textu a nabízí možnosti stylistických úprav, korekcí gramatiky či formátování textu.

Hlavní vlastnosti přepisu:

- Oprava pravopisných a gramatických chyb
- Možnost přeformulování vět pro lepší srozumitelnost
- Automatické formátování textu pro přehlednější strukturu

Přepis je plně integrován do sekce zápisů v aplikaci Clisync, kde mohou psychologové své poznámky snadno editovat a vylepšovat.

4.4.3 Text to speech

Funkce **Text to Speech** by umožňovala psychologům nechat si přečíst obsah zápisu nahlas. Tato funkcionalita je užitečná zejména při revizi poznámek nebo při práci na více úkolech současně.

Výhody implementace:

- Možnost poslechu textu místo jeho čtení
- Zvýšení dostupnosti pro uživatele s poruchami čtení
- Lepší soustředění na obsah zápisů

Tato funkcionalita může být zvážena pro budoucí verze aplikace, pokud bude poptávka ze strany uživatelů.

4.4.4 Speech to text

Funkce **Speech to Text**, která by umožňovala převádět mluvený projev na psaný text, **není v aplikaci implementována**. I když by mohla být užitečná pro rychlé zaznamenání poznámek, nebyla zatím zahrnuta kvůli technickým a uživatelským požadavkům.

Potenciální výhody jejího budoucího nasazení:

- Možnost hlasového diktování poznámek
- Rychlejší zaznamenávání informací během konzultací
- Eliminace potřeby manuálního psaní zápisů

Tato funkcionalita může být zvážena pro budoucí verze aplikace, pokud bude poptávka ze strany uživatelů.

5. Závěr

Cílem tohoto projektu bylo vytvořit mobilní aplikaci Clisync, která by pomohla psychologům zefektivnit jejich administrativní práci a organizaci poznámek. Během vývoje bylo dosaženo několika klíčových výsledků.

Podařilo se vytvořit funkční webovou aplikaci využívající moderní technologie jako React a Next.js, která nabízí intuitivní rozhraní pro správu zápisků a organizaci práce psychologů. Webová aplikace byla úspěšně implementována do mobilního prostředí pomocí technologie WebView s využitím frameworku Expo, což umožnilo sdílení kódu mezi platformami a zefektivnilo vývoj.

V rámci projektu byly implementovány dvě základní AI funkcionality – souhrn dokumentu a přepis zápisků prostřednictvím OpenAI API. Tyto funkce demonstrují potenciál umělé inteligence v oblasti zpracování a optimalizace textových záznamů psychologů. Původní záměr self-hostingu AI modelu bohužel nemohl být realizován kvůli technickým a infrastrukturním omezením, což vedlo k využití externího API jako kompromisního řešení.

Otázka self-hostingu AI modelů se ukázala jako komplexní problém vyžadující důkladnou analýzu. Přestože self-hosting nabízí významné výhody v podobě lepší kontroly nad daty, větší flexibility a potenciálně nižších dlouhodobých nákladů, jeho implementace představuje značnou technickou výzvu. V průběhu projektu bylo zjištěno, že nasazení vlastních AI modelů vyžaduje nejen důkladný výběr vhodného modelu z platformy jako je Hugging Face, ale také robustní infrastrukturu s dostatečným výpočetním výkonem, RAM a úložištěm. Realizace takového řešení by vyžadovala pokročilou konfiguraci serverů, Docker kontejnerů a správu softwarového prostředí, což přesahovalo dostupné zdroje projektu.

Během vývoje bylo zjištěno, že publikace WebView aplikace do oficiálních obchodů App Store a Google Play čelí významným překážkám, především kvůli požadavkům na nativní funkcionalitu a přidanou hodnotu oproti webové verzi. Jako alternativa byla zvolena distribuce prostřednictvím Expo Go a možnost přímé instalace APK pro Android.

Přestože některé plánované funkce jako Text to Speech a Speech to Text nebyly implementovány, projekt poskytuje solidní základ, který může být v budoucnu rozšířen. Aplikace v současné podobě splňuje základní cíl – usnadnit psychologům správu jejich zápisků a organizaci práce prostřednictvím digitálního řešení.

Budoucí vývoj by mohl zahrnovat implementaci offline režimu, přidání nativních funkcí pro lepší integraci s mobilními zařízeními, rozšíření AI funkcionalit o další nástroje pro práci s textem a potenciálně i vlastní hosting AI modelů s důrazem na bezpečnost a ochranu citlivých dat.

Literatura

[1] HELLER, Martin. *What is Visual Studio Code? Microsoft's extensible code editor*. Online. InfoWorld. 2022. Dostupné z: https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html. [cit. 2025-03-30].

[2] MIRO. *What is Miro? Get to know our innovation workspace*. Online. Miro. Dostupné z: https://miro.com/what-is-miro/. [cit. 2025-03-30].

[3] GEEKSFORGEEKS. *React Introduction*. Online. Geeksforgeeks. 2025, 2025-01-28. Dostupné z: https://www.geeksforgeeks.org/reactjs-introduction/. [cit. 2025-03-30].

[4] MATÉU.SH. *What is the Virtual DOM in React?* Online. Geeksforgeeks. 2024. Dostupné z: https://www.freecodecamp.org/news/what-is-the-virtual-dom-in-react/. [cit. 2025-03-30].

[5] WIKIPEDIA. *Next.js*. Online. Wikipedia. 2025, 2025-03-25. Dostupné z: https://en.wikipedia.org/wiki/Next.js. [cit. 2025-03-30].

[6] CEZAR, Felipe. *Understanding All Types of Page Rendering in Next.js*. Online. dev.to. 2024. Dostupné z: https://dev.to/felipecezar01/understanding-all-types-of-page-rendering-in-nextjs-1fbi. [cit. 2025-03-30].

[7] AHMED, Asim. *What is Vercel and Why You Should Use It?* Online. Fishtank. 2023. Dostupné z: https://www.getfishtank.com/insights/what-is-vercel. [cit. 2025-03-30].

[8] GEEKSFORGEEKS. *Introduction to Tailwind CSS*. Online. Geeksforgeeks. 2024, 2024-10-07. Dostupné z: https://www.geeksforgeeks.org/introduction-to-tailwind-css/. [cit. 2025-03-30].

[9] DAISYUI. *Introduction*. Online. DaisyUI. 2024. Dostupné z: https://daisyui.com/docs/intro/. [cit. 2025-03-30].

[10] MIGHTY. *What Is a Native App?* Online. Mighty. 2024. Dostupné z: https://www.mightynetworks.com/resources/native-app. [cit. 2025-03-30].

[11] WIKIPEDIA. *React Native*. Online. Wikipedia. 2015, 2025-01-29. Dostupné z: https://en.wikipedia.org/wiki/React_Native. [cit. 2025-03-30].

[12] GOSS, Thomas. *Flutter vs Expo*. Online. MobiLoud. 2024. Dostupné z: https://www.mobiloud.com/blog/flutter-vs-expo. [cit. 2025-03-30].

[13] BUCK, Andrew. *Native Apps vs Webview Apps - What's the Best Choice for Your Business?* Online. MobiLoud. 2025. Dostupné z: https://www.mobiloud.com/blog/native-app-vs-webview-app. [cit. 2025-03-30].

[14] APPLE. *App Review Guidelines*. Online. Apple Developer. 2024, 2024-09-13. Dostupné z: https://developer.apple.com/app-store/review/guidelines/. [cit. 2025-03-30].

[15] CITRUSBITS. *Difference Between AppStore vs Google Play Store*. Online. CitrusBits. 2017. Dostupné z: https://citrusbits.com/difference-between-app-store-vs-google-play-store/. [cit. 2025-03-30].

[16] SPHINX. *How to Choose AI Model for Your Project?* Online. Medium. 2024. Dostupné z: https://medium.com/@sphinxshivraj/how-to-choose-ai-model-for-your-project-9ef5316dfda0. [cit. 2025-03-31].

[17] NADIY. *What Are Self-Hosted AI Solutions?* Online. Lizard Global. 2024. Dostupné z: https://www.lizard.global/blog/what-are-self-hosted-ai-solutions-benefits-implementation. [cit. 2025-03-31].

[18] HUGGINGFACE. *Hugging Face*. Online. 2023. Dostupné z: https://huggingface.co/. [cit. 2025-03-31].

[19] LIGHTNING AI. *LitServe*. Online. 2024. Dostupné z: https://lightning.ai/docs/litserve/home. [cit. 2025-03-31].

Seznam obrázků, grafů, tabulek