VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

MOBILNÍ APLIKACE PRO EVIDENCI ZDRAVOTNÍCH ZÁZNAMŮ

Bakalářská práce

Autor práce: Oliver Zlatuška

Vedoucí práce: Ing. Marek Musil

Jihlava 2026

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Oliver Zlatuška**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.

Název práce:	**Mobilní aplikace pro evidenci zdravotních záznamů**
Vedoucí práce:	Ing. Marek Musil
Cíl práce:	Cílem práce je vytvořit mobilní aplikaci pro osobní evidenci zdravotních záznamů pacienta. Aplikace bude využívat lokální databázi a bude určena pro evidenci osobních záznamů (léky, nemoci, návštěvy u lékaře). Aplikace umožní export dat pro lékaře nebo pro zálohu na (Google) cloud. Při tvorbě GUI budou zvážena specifika uživatele (pacienta). Dle možností budou implementovány funkce oznámení (notifikace).

Abstrakt

Tato práce se zabývá návrhem a implementací mobilní aplikace pro osobní evidenci zdravotních záznamů, vytvořené primárně pro operační systémy iOS a Android. Aplikace je vyvinuta pomocí frameworku React Native a platformy Expo, přičemž klade důraz na nezávislou „offline-first“ architekturu. Pro efektivní lokální ukládání dat o lécích, diagnózách a návštěvách lékaře je využita databáze SQLite v kombinaci s moderním nástrojem Drizzle ORM. Mezi klíčové funkcionality patří inteligentní správa domácí lékárničky s možností zpětné úpravy historie užívání, generování lékařských reportů ve formátu PDF a zabezpečení citlivých údajů pomocí biometrické autentizace. Uživatelské rozhraní bylo navrženo iterativně s důrazem na ergonomii a prevenci chyb. Výsledkem práce je plně funkční a bezpečná aplikace, která pacientům usnadňuje dlouhodobou správu jejich zdraví s možností budoucího rozšíření o cloudové zálohování.

Klíčová slova

Databáze; digitalizace zdravotnictví; eHealth; mHealth; mobilní aplikace; offline-first; osobní evidence; React Native; uživatelské rozhraní; zdravotní záznamy

Abstract

The aim of this bachelor thesis is the design and implementation of a mobile application for personal health record management, developed primarily for iOS and Android operating systems. The application is built using the React Native framework and the Expo platform, emphasizing an independent offline-first architecture. A local SQLite database, integrated with the modern Drizzle ORM tool, is utilized to efficiently store data regarding medications, diagnoses, and medical appointments. Key features include intelligent home pharmacy management with the ability to retroactively edit usage history, generating medical reports in PDF format, and securing sensitive data (e.g., through biometric authentication). The user interface was designed iteratively with a strong focus on ergonomics and user error prevention. The result of this thesis is a fully functional and secure application that facilitates long-term health management for patients, with the potential for future expansion to include cloud backups.

Keywords

Databases; healthcare digitalization; eHealth; mHealth; mobile apps; offline-first; personal records; React Native; user interface; health records

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 14. dubna 2026

…….

Podpis studenta/ky

Poděkování

Rád bych poděkoval panu Ing. Marku Musilovi za vedení mé bakalářské práce a za podnětné rady při jejím vypracování. Mé poděkování patří také rodině a přátelům za jejich trpělivost a podporu v průběhu celého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface (rozhraní pro programování aplikací)
ČSSZ	Česká správa sociálního zabezpečení
eHealth	Electronic Health (elektronické zdravotnictví)
EHR	Electronic Health Record (elektronický zdravotní záznam)
ER	Entity-Relationship (diagram datových entit)
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation
HL7	Health Level 7 (standard pro výměnu zdravotnických dat)
iOS	iPhone Operating System
IoT	Internet of Things
IZIP	Internetový zápisník zdravotní péče
JSON	JavaScript Object Notation
MDR	Medical Device Regulation (nařízení EU o zdravotnických prostředcích)
mHealth	Mobile Health (mobilní zdravotnictví)
NoSQL	Not Only SQL
NZIP	Národní zdravotnický informační portál
ORM	Object-Relational Mapping
PDF	Portable Document Format
PHR	Personal Health Record (osobní zdravotní záznam)
SDK	Software Development Kit
SQL	Structured Query Language
SÚKL	Státní ústav pro kontrolu léčiv
UI	User Interface (uživatelské rozhraní)
ÚOOÚ	Úřad pro ochranu osobních údajů
URL	Uniform Resource Locator
UX	User Experience (uživatelská zkušenost)
WHO	World Health Organization (Světová zdravotnická organizace)

Úvod

Digitalizace zdravotnictví nabývá stále většího významu a systémy pro správu zdravotních záznamů jsou dnes neodmyslitelnou součástí této transformace. Jednou z klíčových oblastí je možnost, aby pacienti měli kontrolu nad svými zdravotními údaji. Pacienti obecně nemají přístup k nástrojům, které by jim umožnily sledovat a uchovávat zdravotní záznamy, a to často vede k nepřehlednosti a ztrátě důležitých informací o jejich zdravotním stavu. Mnozí zapomínají na důležité detaily, jako jsou prodělané diagnózy, užívané léky nebo přesné termíny návštěv u lékaře.

Popsaný problém mě vedl k volbě tématu mé bakalářské práce. Z vlastní zkušenosti vím, jaké nepříjemnosti může způsobit nedostatek informací o vlastním zdravotním stavu. Jelikož jsem častěji nemocný, opakovaně jsem se dostával do situací, kdy jsem si nepamatoval, kdy přesně jsem byl u lékaře, které léky jsem bral, nebo zda jsem jejich medikaci využíval. V České republice nemají pacienti běžně přístup ke svým zdravotním záznamům, jak tomu je například u lékařů. To může vyvolávat nejistotu ohledně aktuálního zdravotního stavu. Vytvoření systému, který by poskytl uživatelům podobný přístup, by mohlo přispět k většímu klidu a kontrole nad zdravotní péčí.

Cílem bakalářské práce je proto vyvinout mobilní aplikaci pro osobní evidenci zdravotních záznamů. Tato aplikace nese název eZdraví. Aplikace bude využívat lokální databázi a umožní uživatelům evidovat důležité informace, jako jsou léky, které užívali, prodělané diagnózy a návštěvy u lékaře. Zároveň aplikace nabídne možnost exportu dat, například ve formátu PDF, což usnadní sdílení zdravotních záznamů s lékaři a zároveň poskytne možnost zálohování na cloudové platformy jako je například Google Cloud.

Pozornost budu věnovat také uživatelskému rozhraní (UI) a uživatelské zkušenosti (UX), které jsou ve zdravotnictví často podceňovány. Aplikace musí být navržena tak, aby byla snadno použitelná pro širokou škálu uživatelů, včetně těch starších nebo technicky méně zdatných. UI design v oblasti zdravotnictví je klíčový, protože musí být nejen intuitivní, ale také vizuálně přehledný a srozumitelný pro uživatele všech věkových kategorií. Díky tomu budou mít uživatelé lepší přehled o svých zdravotních záznamech a budou schopni snadněji dodržovat své léčebné plány.

Zabezpečení dat bude samozřejmě jedním z hlavních aspektů návrhu aplikace, protože práce s citlivými zdravotními údaji vyžaduje dodržování přísných standardů ochrany soukromí a bezpečnosti. Díky lokálnímu uložení dat bude aplikace přístupná i bez připojení k internetu, což zvýší její použitelnost. Do budoucna může být zvažována i možnost cloudového zálohování, které poskytne dodatečnou úroveň bezpečnosti a možnosti obnovy dat v případě ztráty nebo poškození zařízení.

Aplikace eZdraví umožňuje evidovat diagnózy s navazující dokumentací, spravovat léčebné režimy s automatickým odpočtem zásob domácí lékárničky a zaznamenávat návštěvy lékaře včetně fotografické dokumentace. Součástí je kalendářní přehled zdravotních záznamů, biometrické zabezpečení a export dokumentace do formátu PDF nebo JSON pro účely zálohy. Veškerá data jsou ukládána výhradně lokálně, což zajišťuje funkčnost bez připojení k internetu a plné soukromí zdravotních údajů.

1 Digitalizace zdravotnictví

Zdravotnictví patří k oblastem, kde digitální transformace probíhá s výrazným zpožděním za jinými odvětvími. Zatímco v bankovnictví, dopravě nebo vzdělávání má dnes běžný uživatel přístup ke svým datům a historii transakcí kdykoli a odkudkoli, v oblasti zdravotní péče platí opak: zdravotní záznamy pacienta jsou ve většině případů roztříštěny mezi jednotlivé poskytovatele péče, vedeny v papírové nebo proprietární digitální formě a pacient sám k nim zpravidla přístup nemá. Tato asymetrie informací, kdy lékař ví o pacientovi více než pacient sám, komplikuje kontinuitu péče, zvyšuje riziko chybných rozhodnutí při změně poskytovatele a staví pacienta do pasivní role příjemce péče namísto aktivního účastníka. Digitalizace zdravotnictví tento stav postupně mění, přičemž jejím dlouhodobým cílem je vytvořit prostředí, v němž jsou zdravotní data dostupná oprávněným subjektům v reálném čase a pacient nad nimi má kontrolu. (Ministr zdraví z. ú., 2024)

Současné trendy v digitalizaci zdravotnictví se soustředí do několika klíčových oblastí, které vzájemně ovlivňují nejen způsob poskytování péče, ale také roli samotného pacienta v tomto procesu.

Elektronická zdravotnická dokumentace (EHR — Electronic Health Records)

Elektronické zdravotní záznamy umožňují komplexní digitální evidenci zdravotní historie pacienta a jejich sdílení mezi různými poskytovateli péče. Jejich klíčovou přidanou hodnotou oproti papírové dokumentaci je okamžitá dostupnost dat v místě ošetření, eliminace rizika ztráty nebo nečitelnosti záznamu a možnost integrace s laboratorními systémy a diagnostickými přístroji. V ideálním stavu tvoří EHR centrální datové úložiště, z nějž čerpají všichni lékaři pacienta bez ohledu na jejich geografickou polohu nebo institucionální příslušnost. V České republice je však dostupnost těchto záznamů pro pacienty omezena. Elektronicky jsou plošně dostupné převážně pouze eRecepty a eNeschopenky, zatímco komplexní klinická dokumentace zůstává v dispozici jednotlivých zdravotnických zařízení. (Atherton, 2011)

Telemedicína

Telemedicína zahrnuje poskytování zdravotní péče na dálku prostřednictvím digitálních komunikačních technologií — vzdálené konzultace s lékaři, monitoring chronicky nemocných pacientů v domácím prostředí nebo přenos diagnostických dat. Ve Finsku a Estonsku, které jsou v digitalizaci zdravotnictví v evropském kontextu nejdále, jsou rozvíjeny dálkové monitorovací nástroje propojené s mobilními zdravotnickými přístroji a systémy Internetu věcí (IoT). Telemedicína snižuje zátěž ambulancí, zvyšuje dostupnost péče pro obyvatele vzdálených oblastí a umožňuje průběžné sledování zdravotního stavu bez nutnosti fyzické návštěvy lékaře. (Národní telemedicínské centrum Fakultní nemocnice Olomouc, 2023)

Umělá inteligence ve zdravotnictví

Umělá inteligence nachází uplatnění v analýze medicínských obrazových dat, prediktivní diagnostice a personalizované medicíně. Algoritmy strojového učení jsou schopny identifikovat patologické nálezy v rentgenových snímcích nebo histologických preparátech s přesností srovnatelnou s odborníky, a to ve zlomku času potřebného pro ruční analýzu. AI rovněž nachází uplatnění v predikci zdravotních komplikací u chronicky nemocných pacientů. Systém může na

základě trendů v datech pacienta upozornit na blížící se zhoršení stavu dříve, než se projeví klinickými příznaky. (Po medině, 2024)

1.1 Historický vývoj elektronizace zdravotních záznamů

První pokusy o zavedení počítačů do zdravotnictví sahají do 60. let 20. století, kdy velké nemocnice začaly pořizovat výpočetní techniku primárně pro administrativní účely: fakturace, evidence pacientů a správa zásob. Klinická dokumentace tehdy zůstávala papírová; počítače sloužily ekonomice provozu nemocnice, nikoli péči o pacienta. (Atherton, 2011)

Průlom v přístupu ke klinické dokumentaci přinesl americký lékař Lawrence Weed, který na přelomu 60. a 70. let na University of Vermont vyvíjel systém PROMIS (Problem-Oriented Medical Information System). Weed zavedl strukturovaný způsob záznamu zdravotního stavu pacienta organizovaný kolem konkrétních zdravotních problémů — přístup, který se stal základem pro moderní elektronické zdravotní záznamy a jehož principy jsou využívány dodnes. (National Library of Medicine, 2019) V průběhu 70. a 80. let se EHR systémy začaly rozšiřovat zejména ve velkých nemocnicích, avšak jejich vzájemná interoperabilita zůstávala minimální. Systémy jednotlivých zařízení spolu typicky nekomunikovaly a přenos dat pacienta mezi institucemi byl obtížný.

Devadesátá léta a masový nástup internetu otevřely novou etapu: diskusi o osobních zdravotních záznamech spravovaných samotným pacientem. Tento koncept, označovaný zkratkou PHR (Personal Health Record), představuje zdravotní dokumentaci, jejíž správu a kontrolu nad sdílením dat drží pacient nikoli poskytovatel péče. (Terrell, 2024) PHR se stal základem pro celou generaci aplikací a portálů, které se snaží přenést kontrolu nad zdravotními daty ze zdravotnických institucí na samotné pacienty, a právě z tohoto konceptu vychází i aplikace eZdraví vytvářená v rámci této bakalářské práce.

1.2 Současný stav v České republice

V České republice probíhá digitalizace zdravotnictví po dlouhá léta výrazně pomalejším tempem než v zemích západní a severní Evropy. Tento stav je dlouhodobě terčem kritiky odborné veřejnosti. Odborníci upozorňují, že Česko v digitální transformaci péče zaostává, a varují před přetrvávající roztříštěností systémů. Tento deficit vedl až k veřejným výzvám expertů, aby se politická reprezentace k systematické nápravě a dokončení digitalizace zdravotnictví jasně zavázala ve svých programech (Ministr zdraví z. ú., 2025a).

Zásadním pokusem o vytvoření centralizovaného úložiště zdravotních dat dostupného jak lékařům, tak pacientům bylo spuštění systému IZIP (Internetový přístup ke zdravotním informacím pacienta) v roce 2001. IZIP představoval webový portál, do nějž lékaři zaznamenávali zdravotní události a pacient měl k těmto záznamům přístup prostřednictvím internetového prohlížeče. Projekt byl svého druhu průkopnický. V době svého spuštění neměl v Evropě obdobu. Přestože se k systému připojily stovky zdravotnických zařízení a tisíce pacientů, projekt byl v roce 2012 ukončen. Jako příčiny neúspěchu jsou uváděny nízká motivace lékařů k aktivnímu plnění systému, technické problémy s integracemi, a v neposlední řadě otázky financování a odpovědnosti za správu citlivých zdravotních dat. Přesto IZIP doložil jak společenskou

poptávku po takovém řešení, tak technickou realizovatelnost myšlenky pacientsky přístupného zdravotního záznamu. (Economia, a.s., 2015)

Po ukončení IZIP se elektronizace zdravotnictví v ČR soustředila na dílčí agendy. Nejvýznamnějším výsledkem je systém eReceptu, který od roku 2018 nahradil papírové recepty. (Státní ústav pro kontrolu léčiv (SÚKL), 2017) V roce 2020 byla zavedena elektronická neschopenka a postupně se rozvíjí propojení mezi nemocničními informačními systémy. (Česká správa sociálního zabezpečení (ČSSZ), 2020) Ministerstvo zdravotnictví představilo Národní strategii elektronického zdravotnictví 2025–2035, jejímž cílem je vybudovat sdílenou zdravotnickou infrastrukturu umožňující výměnu dat mezi poskytovateli péče. (Ministerstvo zdravotnictví ČR, 2025) Od ledna 2026 vstoupila v platnost novela zákona o elektronizaci zdravotnictví, která zavádí konkrétní nástroje tohoto směřování: elektronickou žádanku (e-žádanku) umožňující přímé předání žádosti lékaře specialistovi, sdílený zdravotní záznam zpřístupňující základní informace o pacientovi (alergie, krevní skupina, preventivní prohlídky) a registr oprávnění pro správu souhlasů k přístupu ke zdravotnickým údajům. (Ministr zdraví z. ú., 2025b) Přístup pacienta ke komplexní vlastní zdravotní dokumentaci nicméně stále není standardně zajištěn. Pacient zpravidla disponuje pouze tím, co si sám zaznamenal, nebo dokumenty v papírové formě, které obdržel od lékaře.

Právě tato mezera — absence nástroje, který by pacientovi umožnil vést vlastní průběžnou zdravotní agendu nezávisle na tom, u kterého lékaře je léčen, tvoří motivaci pro vznik aplikace eZdraví.

1.3 Výzvy digitalizace

Navzdory technologickému pokroku čelí digitalizace zdravotnictví řadě systémových výzev, jejichž řešení není primárně technické, ale organizační, legislativní a kulturní. Pochopení těchto výzev je relevantní i pro vývoj pacientsky orientovaných aplikací, neboť přímo ovlivňují, jaká data jsou v prostředí dostupná a jaká bezpečnostní a právní omezení musí aplikace respektovat.

Bezpečnost a ochrana osobních údajů

Zdravotní záznamy představují z pohledu ochrany osobních údajů jednu z nejcitlivějších kategorií dat, se kterými zdravotnická zařízení pracují. GDPR jim přiznává zvláštní právní režim a jejich zpracování podmiňuje existencí zákonného důvodu. V praxi jím bývá nejčastěji plnění zákonné povinnosti nebo samotné poskytování zdravotní péče. Pokud dojde k bezpečnostnímu incidentu, například kybernetickému útoku, který ohrozí dostupnost nebo integritu těchto dat, nelze se odvolat na výjimku z ohlašovací povinnosti. Incident musí být hlášen dozorovému úřadu bez výjimky. Z toho pro návrh zdravotnických aplikací plyne jasný závěr: sbírat jen data skutečně nezbytná a ukládat je způsobem minimalizujícím riziko jejich zneužití. (Úřad pro ochranu osobních údajů (ÚOOÚ), 2024)

Interoperabilita různých systémů

České zdravotnictví se dlouhodobě potýká s tím, že jednotlivá zařízení sice digitální systémy provozují, ale tato řešení spolu nedokážou komunikovat. Každý dodavatel historicky vyvíjel vlastní produkt podle vlastních pravidel, bez povinnosti respektovat společné standardy. Výsledkem je prostředí, kde data fakticky končí tam, kde vznikla, a jejich předání jinam vyžaduje manuální zásah nebo tisk papírového výpisu. (Ministerstvo zdravotnictví ČR a Ústav

zdravotnických informací a statistiky ČR, 2025) Standardizační iniciativa HL7 FHIR tento problém řeší tím, že zdravotnická data zpřístupňuje přes otevřená API rozhraní, přičemž na rozdíl od starších přístupů pracuje s jednotlivými datovými prvky záznamu místo přenosu celých dokumentů, což výrazně zjednodušuje integraci i pro aplikace třetích stran. (Intersystems Corporation, Boston, MA, 2024)

Technická infrastruktura a přístupnost

Česká republika v oblasti digitalizace zdravotnictví dlouhodobě zaostává za průměrem Evropské unie. Podle studie KPMG jsou hlavními bariérami nedostatek financí, nekoncepční řízení ze strany státu, nevyhovující legislativa a kybernetická rizika. Česko přitom vydává na zdravotní péči méně, než je unijní průměr, což přímo omezuje prostor pro budování digitální infrastruktury. Respondenti průzkumu navíc za klíčový problém označují absenci silné koordinační agentury, která by za implementaci digitální strategie nesla odpovědnost. (KPMG, 2023)

Samotné nasazení moderních technologií do zdravotnictví naráží na problém s lidským faktorem. Aby měly digitální inovace smysl a byly dlouhodobě funkční, je naprosto nezbytné klást důraz na průběžné vzdělávání veškerého zdravotnického personálu. Rozvoj digitálních kompetencí je však nutný také na straně pacientů, aby vůbec byli schopni nabízené elektronické služby obsluhovat. (Vanis Karel, Navrátil Marek, Hlávka Jakub, 2025) Rozvoj digitálních dovedností zdravotnického personálu v České republice zatím postrádá ucelenou koncepci. K vzdělávání v této oblasti dochází spíše příležitostně a bez celostátní koordinace. Problém je přitom do jisté míry sebeposilující: tam, kde digitální nástroje ještě nejsou zavedeny, nevzniká ani poptávka po školení, a naopak, kde chybí proškolený personál, narážejí implementace nových systémů na odpor či neefektivitu. (Ministr zdraví z. ú., 2025c)

Legislativní rámec

Zdravotnická legislativa reaguje na technologický vývoj se zpožděním, které v praxi vytváří právní nejistotu. Zákon o zdravotních službách, zákon o elektronizaci zdravotnictví a evropské nařízení GDPR tvoří základní právní rámec, avšak otázky jako právní status sdílené elektronické dokumentace, odpovědnost za integritu dat nebo pravidla přeshraničního přenosu zdravotních dat zůstávají v řadě aspektů nedostatečně upraveny. Novela zákona o elektronizaci zdravotnictví, účinná od ledna 2026, tyto mezery částečně adresuje zavedením sdíleného zdravotního záznamu a registru oprávnění. Analytici však upozorňují, že samotný legislativní rámec nestačí. Bez návazného implementačního plánu, finančních incentiv a závazného zapojení poskytovatelů péče hrozí, že se zamýšlené benefity digitalizace v praxi plně neprojeví (Ministr zdraví z. ú., 2025b). Tyto obavy rezonují i napříč oborovými organizacemi, které zdůrazňují, že bez silné politické vůle a jasně definovaných priorit hrozí českému zdravotnictví další prohlubování technologického dluhu a ztráta kroku se zbytkem Evropy (Ministr zdraví z. ú., 2025a).

2 Mobilní aplikace ve zdravotnictví

Mobilní zdravotnictví, v mezinárodní terminologii označované jako mHealth, je součástí širšího konceptu eHealth a zahrnuje veškeré zdravotní intervence realizované prostřednictvím bezdrátových technologií. Od chytrých telefonů a tabletů až po specializovaná zařízení pro vzdálené sledování pacientů. (Schaub Michael P., Lee Jenny Yi-Chen, Pirona Alessandro, 2019) S tím, jak se chytré telefony staly samozřejmou součástí života prakticky každého člověka, začaly mobilní aplikace přebírat roli nejdostupnějšího vstupního bodu do digitální zdravotní péče. Na rozdíl od nemocničních informačních systémů nebo pacientských portálů dostupných jen u počítače je mobilní zařízení s pacientem neustále, a právě tato přítomnost z něj dělá přirozený nástroj pro průběžnou správu zdravotní agendy. Zájem odborné veřejnosti i regulátorů o tuto oblast přitom výrazně roste, stejně jako povědomí o rizicích spojených se sdílením citlivých zdravotních dat prostřednictvím komerčních aplikací. (iROZHLAS, 2022)

Trh s mHealth aplikacemi zaznamenal mimořádný růst — podle reportu IQVIA Institute bylo v globálních aplikačních obchodech dostupných přes 350 000 zdravotnicky zaměřených aplikací. (Olsen, 2021)

2.1 Kategorie mHealth aplikací a jejich přínos

mHealth aplikace lze kategorizovat podle primárního účelu, který plní vůči uživateli. Porozumění těmto kategoriím je relevantní pro určení místa, které v tomto ekosystému zaujímá aplikace eZdraví.

Aplikace pro monitoring zdravotního stavu

Nejrozšířenější kategorií jsou aplikace zaměřené na průběžné sledování fyziologických parametrů — srdeční frekvence, krevního tlaku, hladiny glukózy nebo kvality spánku — zpravidla ve spojení s nositelným zařízením. Uplatňují se jak v oblasti prevence, tak u pacientů s diagnostikovaným onemocněním, kde slouží jako nástroj průběžného self-monitoringu. Výzkum Masarykovy univerzity na vzorku českých adolescentů zjistil, že nadpoloviční většina z nich mobilní zdravotní aplikace aktivně využívá, přičemž výzkumy opakovaně potvrzují pozitivní vliv těchto nástrojů na fyzickou kondici uživatelů. Zejména v oblasti udržení zdravých návyků u těch, kteří ke změně chování již přistoupili. (Lokajová a Šmahel, 2022)

Aplikace pro správu léčby

Druhou kategorií jsou aplikace zaměřené na podporu dodržování léčebného plánu. Základní funkcionalitou je připomínání užívání léků v nastavených časech — pro pacienty s komplexními lékovými schématy prakticky nenahraditelný nástroj. Pokročilejší aplikace umožňují evidovat zásoby léků a sledovat jejich expiraci, případně také zaznamenávat vedlejší účinky. Česká aplikace mZdraví, provozovaná na infrastruktuře certifikované v souladu s GDPR, řadí podporu pacientské adherence k léčbě mezi klíčové přínosy digitalizace péče. (mZdraví, 2024) Do této kategorie spadá i eZdraví, které danou funkcionalitu rozšiřuje o vazbu na konkrétní diagnózu, správu lékových zásob a chronologicky řazenou historii užívání.

Zbývající dvě kategorie — diagnostické a konzultační aplikace a aplikace vzdělávací a preventivní — jsou pro svůj specifický charakter popsány samostatně níže.

2.2 Diagnostické a konzultační aplikace

Třetí kategorií mHealth aplikací jsou nástroje umožňující orientační posouzení zdravotního stavu nebo přímý kontakt s lékařem na dálku. Jde o kategorii, která přesahuje rámec čisté evidence a připomínání a pohybuje se na hranici poskytování zdravotní péče.

Základní vrstvou jsou takzvané symptom checkery. Aplikace, do nichž uživatel zadá pozorované obtíže a obdrží informaci o možných příčinách a doporučení, zda je vhodné vyhledat lékaře. Složitější variantou jsou telemedicínské platformy umožňující videokonzultaci s lékařem bez nutnosti fyzické přítomnosti v ordinaci. V České republice se tento způsob péče začíná stabilně zavádět jako doplněk standardní ambulantní péče, zejména tam, kde je fyzická dostupnost ordinace omezená. (Iniciativa Sníh, 2023)

Z regulatorního hlediska je tato kategorie nejcitlivější. Klíčové je rozlišení mezi aplikací, která pouze prezentuje obecné zdravotní informace, a aplikací, která sama aktivně vyhodnocuje zadané údaje a na jejich základě generuje závěr. Druhá varianta již naplňuje znaky zdravotnického softwaru a podléhá přísnějšímu evropskému nařízení MDR. Výrobci musejí počítat s tím, že jejich produkt vstupuje do regulovaného prostředí a nese s sebou odpovědnost za bezpečnost uživatele. (Koubová, 2023)

2.3 Vzdělávací a preventivní aplikace

Čtvrtou a nejpočetnější kategorií jsou aplikace zaměřené na zdravotní gramotnost, změnu chování a podporu zdravého životního stylu. Na rozdíl od předchozích kategorií nevyžadují zdravotní diagnózu ani kontakt s lékařem — jsou primárně nástrojem pro každodenní prevenci.

Výzkum Masarykovy univerzity zaměřený na české uživatele ukazuje, že preventivní mHealth aplikace se v každodenním životě zaváděly zejména jako podpora fyzické aktivity, tedy pro monitorování pohybu, spánku, výživy nebo plnění stanovených cílů. Důležitý poznatek výzkumu je, že aplikace jsou efektivnější v udržování již zahájené změny chování než v jejím prvotním nastartování. (Lokajová a Šmahel, 2022) Vedle fyzické složky zdraví nabývají na významu také aplikace zaměřené na duševní pohodu. Nabízejí nástroje zvládání stresu nebo zprostředkovávají sdílení zkušeností mezi uživateli se stejnými obtížemi. Evropská komise hodnotí tuto kategorii aplikací jako příležitost ke snížení počtu zbytečných konzultací v nemocnicích a posílení zodpovědnosti občanů za vlastní zdraví. (iROZHLAS, 2022)

Zásadní podmínkou pro skutečný přínos preventivních aplikací je zdravotní gramotnost jejich uživatelů — tedy schopnost vyhledané informace správně pochopit a využít. Podle výzkumu ZP MV ČR z roku 2021 měla nadpoloviční část respondentů obtíže s posouzením věrohodnosti zdravotních informací na internetu — tedy právě v oblasti, která je pro smysluplné využití preventivních aplikací klíčová. (Zdravotní pojišťovna ministerstva vnitra ČR, 2021)

2.4 Souhrnný dopad mHealth na kvalitu péče

Celkový dopad mobilního zdravotnictví na kvalitu péče je hodnocen pozitivně, přestože jeho rozsah se liší podle kategorie aplikace a cílové skupiny. Nejvýrazněji doloženým přínosem je podpora pacientské adherence k léčbě. Nezanedbatelný je také vliv na kvalitu komunikace mezi

pacientem a lékařem — pacient, který přichází na konzultaci vybaven přehledem své medikace a zdravotní historie, umožňuje efektivnější a podložené klinické rozhodování. (mZdraví, 2024)

Z hlediska zaměření této práce je klíčová kategorie aplikací pro správu léčby a evidenci zdravotních záznamů. Stávající nabídka na trhu bývá zaměřena buď pouze na připomínání léků bez vazby na konkrétní diagnózu, nebo na sběr biometrických dat bez správy dokumentace. Aplikace eZdraví vzniká s ambicí tyto dvě dimenze propojit a doplnit je o možnost exportu dokumentace pro potřeby lékaře, čímž reaguje na mezeru, která ve stávajících řešeních přetrvává.

3 Technologický základ aplikace

Volba technologického stacku pro vývoj mobilní aplikace je rozhodnutím s dlouhodobými důsledky. Ovlivňuje nejen okamžitý průběh vývoje, ale také budoucí udržovatelnost kódu, dostupnost odborníků pro případné rozšíření projektu a schopnost aplikace reagovat na nové požadavky operačních systémů. V kontextu zdravotnické aplikace jsou navíc relevantní specifické požadavky: spolehlivá práce s lokálními daty bez závislosti na internetovém připojení, přístup k bezpečnostním funkcím zařízení pro ochranu citlivých zdravotních údajů a dostatečně zralý ekosystém knihoven pokrývající potřebné funkcionality. Tato kapitola popisuje dostupné technologické přístupy k vývoji multiplatformních mobilních aplikací a odůvodňuje volbu konkrétních technologií použitých v aplikaci eZdraví.

3.1 Multiplatformní vývoj mobilních aplikací

Mobilní aplikace lze vyvíjet třemi základními přístupy. Nativní vývoj znamená psaní samostatné kódové základny pro každou platformu v jejím primárním jazyce — Swift nebo Objective-C pro iOS, Kotlin nebo Java pro Android. Výsledná aplikace dosahuje maximálního výkonu a má plný přístup ke všem funkcím operačního systému, avšak za cenu dvojnásobného vývojového úsilí a nutnosti udržovat dvě paralelní kódové základny. Hybridní přístup využívá webové technologie (HTML, CSS, JavaScript) obalené do nativního kontejneru. Tyto aplikace však trpí výkonnostními omezeními a omezeným přístupem k nativním funkcím zařízení. Třetím přístupem je multiplatformní vývoj se sdílenou kódovou základnou, který kombinuje produktivitu webového vývoje s výkonem a přístupem k nativním funkcím. Do této kategorie spadají frameworky React Native a Flutter, které v současnosti dominují trhu multiplatformních řešení. (Occhino, 2015)

3.1.1 React Native

React Native, vyvinutý společností Meta (dříve Facebook) a open-source od roku 2015, umožňuje vývoj mobilních aplikací pro iOS i Android z jedné kódové základny napsané v jazyku JavaScript nebo TypeScript. Na rozdíl od hybridních přístupů nevykresluje aplikaci v prohlížečovém enginu, ale překládá komponenty JavaScriptového uživatelského rozhraní na skutečné nativní komponenty příslušné platformy — tlačítko v React Native aplikaci je na iOS skutečné UIButton a na Androidu skutečné Button. Uživatel proto nevnímá vizuální ani výkonnostní rozdíl oproti nativní aplikaci. (React Native, 2024)

Původní architektura React Native je postavena na odděleném JavaScript vlákně, které obsahuje aplikační logiku, a nativním vlákně, které obsluhuje uživatelské rozhraní. Komunikace mezi nimi probíhá asynchronně prostřednictvím Bridge, což zajišťuje, že výpočetně náročné operace v JavaScriptu neblokují vykreslování rozhraní. Tento model je obzvláště výhodný pro aplikaci pracující s databázovými dotazy. Dotazy probíhají na pozadí a uživatelské rozhraní zůstává responzivní.

Programovací model React Native vychází z deklarativního reaktivního paradigmatu knihovny React. Vývojář popisuje, jak má rozhraní vypadat pro daný stav dat, a framework se stará o efektivní aktualizaci pouze těch částí rozhraní, které se skutečně změnily. Tento přístup výrazně zjednodušuje vývoj složitých interaktivních rozhraní a snižuje počet chyb způsobených nekonzistentním stavem. Komunita React Native patří k největším v ekosystému mobilního

vývoje, což se projevuje v rozsáhlé nabídce open-source knihoven pokrývajících prakticky každou potřebnou funkcionalitu. (React Native, 2024)

3.1.2 Flutter

Flutter je multiplatformní framework vytvořený společností Google, jehož první stabilní verze byla vydána v roce 2018. Na rozdíl od React Native nevyužívá nativní komponenty platformy, ale renderuje celé uživatelské rozhraní vlastním grafickým enginem Skia (od verze 3.10 Impeller) přímo na plátno. Tím dosahuje absolutní vizuální konzistence napříč platformami — aplikace vypadá identicky na iOS, Androidu i na desktopu nebo webu, kam Flutter také cílí. Za tuto konzistenci však platí dvěma způsoby: výsledná aplikace má větší instalační velikost kvůli přibalenému grafickému enginu, a vývojář se musí naučit jazyk Dart, který mimo ekosystém Flutter nemá praktické uplatnění. Pro vývojáře přicházející z prostředí webového vývoje nebo JavaScriptu tak Flutter představuje větší vstupní bariéru než React Native. Flutter je nejvhodnější volbou pro aplikace s nestandardními grafickými požadavky nebo pro projekty, kde je vizuální konzistence napříč platformami klíčová. (Flutter, 2024)

3.1.3 Swift

Swift je programovací jazyk vyvinutý společností Apple a vydaný v roce 2014 jako moderní nástupce jazyka Objective-C pro vývoj aplikací na platformách iOS, macOS, watchOS a tvOS. Jazyk je kompilovaný, staticky typovaný a přináší výrazně bezpečnější práci s pamětí než jeho předchůdce. V kombinaci s frameworkem SwiftUI, který Apple představil v roce 2019, umožňuje deklarativní vývoj nativních uživatelských rozhraní s plnou integrací do systémových funkcí jako HealthKit, ARKit nebo Core Data. Výsledné aplikace dosahují maximálního výkonu a nejúžeji možné integrace s operačním systémem. Zásadním omezením je však platformová exkluzivita — Swift a SwiftUI jsou použitelné výhradně v ekosystému Apple. Vývoj ve Swiftu proto není vhodnou volbou pro projekty, které mají pokrýt také platformu Android, a to bez ohledu na to, jak výhodný by byl výkon nebo integrace s iOS funkcemi. (Apple Inc, 2024)

3.1.4 Srovnání možností

Pro systematické porovnání dostupných přístupů je třeba definovat kritéria relevantní pro konkrétní projekt. V případě aplikace eZdraví jsou klíčovými kritérii multiplatformní pokrytí (iOS i Android z jedné kódové základny), dostupnost knihoven pro lokální databázi a biometrickou autentizaci, výkonnost při práci s databázovými dotazy a přístupnost technologie pro vývojáře se znalostí JavaScriptu. Následující tabulka shrnuje srovnání tří popsaných frameworků z hlediska těchto kritérií.

Tab. 1: Srovnání možností

Framework	*Výhody*	*Nevýhody*
React Native	Multiplatformní, široká podpora, aktivní komunita	Závislost na nativních modulech
Flutter	Jednotný UI/UX, vysoká grafická přizpůsobivost	Menší komunita, nutnost učit se Dart
Swift	Maximální výkon, hluboká integrace s iOS	Omezeno pouze na Apple platformy

Zdroj: vlastní zpracování dle Zlatuška (2026)

Ze srovnání vyplývá, že React Native a Flutter jsou si v mnoha ohledech srovnatelné a oba by technicky projekt umožnily realizovat. Klíčovým diferenciátorem je znalostní profil vývojáře a ekosystém. React Native staví na JavaScriptu a Reactu, což jsou technologie s masivní komunitou a rozsáhlou nabídkou hotových řešení. Swift jako iOS-only technologie byl pro tento projekt od počátku nevhodný.

3.1.5 Výběr řešení

Pro vývoj aplikace eZdraví byl zvolen React Native ve spojení s platformou Expo. Rozhodnutí vychází z kombinace technických a praktických důvodů, které vzájemně posilují jeho opodstatněnost.

Z technického hlediska splňuje React Native všechny požadavky projektu: umožňuje publikování na iOS i Android z jediné kódové základny, disponuje knihovnami pro přístup k lokální SQLite databázi, biometrické autentizaci a systémovému úložišti klíčů, a jeho architektura zajišťuje dostatečný výkon pro aplikaci pracující s relačními daty. Ekosystém React Native je v době psaní práce nejrozsáhlejší ze všech multiplatformních frameworků, což se odráží v dostupnosti knihoven pro specifické potřeby zdravotnické aplikace — od kalendářových komponent po knihovny pro generování PDF.

Z praktického hlediska je React Native postaven na JavaScriptu a TypeScriptu, tedy technologiích, které autor ovládá. Tato skutečnost výrazně snižuje náklady na zahájení projektu a eliminuje riziko spojené s učením nového jazyka paralelně s vývojem samotné aplikace. Výběr Expo jako nadstavby nad React Native dále zjednodušuje přístup k nativním funkcím zařízení a umožňuje iterativní vývoj s okamžitou zpětnou vazbou na fyzickém zařízení bez nutnosti kompilace při každé změně kódu. (Expo, 2024a)

3.2 TypeScript

JavaScript, na kterém je React Native postaven, je dynamicky typovaný jazyk. Datové typy proměnných a návratové hodnoty funkcí nejsou při psaní kódu explicitně deklarovány a jsou určeny až za běhu programu. Tento přístup urychluje počáteční vývoj, avšak v rozsáhlejší kódové základně přináší kategorii chyb, které jsou obtížně odhalitelné testováním: překlep v názvu vlastnosti objektu, předání hodnoty nesprávného typu nebo přístup k neexistujícímu klíči databázového záznamu se projeví až v momentě, kdy danou část kódu uživatel skutečně vykoná. (Microsoft, 2024a)

TypeScript je nadmnožina JavaScriptu vyvinutá společností Microsoft, která do jazyka přidává statický typový systém. Každá proměnná, parametr funkce nebo návratová hodnota může být opatřena typovou anotací definující, jaký druh dat je na daném místě očekáván. Kompilátor TypeScriptu při překladu kódu ověřuje konzistenci těchto anotací a v případě nesouladu hlásí chybu ještě před spuštěním aplikace. Výsledný zkompilovaný JavaScript je funkčně identický s kódem psaným přímo v JavaScriptu. TypeScript je tedy čistě nástrojem vývojové fáze, který do produkčního kódu nepřidává žádnou režii. (Microsoft, 2024b)

Pro aplikaci eZdraví má použití TypeScriptu přímý dopad na tři oblasti. Zaprvé, databázové schéma definované v souboru schema.ts pomocí Drizzle ORM je zdrojem TypeScriptových typů pro veškeré databázové operace — při čtení záznamu z tabulky medication_plans kompilátor přesně ví, jaké vlastnosti výsledný objekt obsahuje a jaké jsou jejich datové typy. Pokus o přístup k neexistujícímu sloupci nebo předání hodnoty nesprávného typu databázovému dotazu je zachycen v době kompilace. Za druhé, typové anotace slouží jako strojově čitelná dokumentace kódu: z datového typu parametru funkce je okamžitě zřejmé, co funkce očekává a co vrací, bez nutnosti číst implementaci nebo komentáře. Za třetí, TypeScript výrazně zlepšuje podporu ze strany vývojových nástrojů: editor dokáže na základě typů nabízet přesné automatické doplňování a okamžitě upozornit na nesprávné použití API knihovny. V kombinaci s ORM vrstvou Drizzle, která je pro TypeScript primárně navržena, TypeScript eliminuje celou třídu potenciálních chyb při práci s databází ještě před spuštěním aplikace. (Product by Drizzle Team, b. r. a)

3.3 Ekosystém Expo

Expo je open-source platforma a sada nástrojů postavená nad frameworkem React Native, která vývojářům poskytuje standardizované prostředí pro vývoj, testování a distribuci multiplatformních mobilních aplikací. Jejím primárním cílem je odstranit složitost správy nativních projektů a potřebu přímé práce v jazycích Swift či Kotlin. Veškerý vývoj je tak plně soustředěn do prostředí JavaScriptu a TypeScriptu, což vývojářům umožňuje zaměřit se výhradně na funkčnost aplikace. (Expo Documentation, 2024a)

Expo Go má však jedno omezení: podporuje pouze moduly, které jsou součástí standardního Expo SDK. Aplikace využívající vlastní nativní moduly třetích stran vyžadují sestavení tzv. development buildu — vlastní verze Expo Go zkompilované s požadovanými závislostmi. Aplikace eZdraví toto omezení nenaráží, neboť veškerá využitá funkcionalita je pokryta standardními moduly Expo SDK.

Expo SDK a využité moduly

Expo SDK je kolekce knihoven zprostředkovávající přístup k nativním funkcím zařízení prostřednictvím jednotného JavaScriptového API. V aplikaci eZdraví bylo využito několik modulů, jejichž výběr přímo reflektuje požadavky aplikace.

Modul `expo-sqlite` poskytuje přístup k databázovému systému SQLite. Na rozdíl od starší synchronní verze nabízí aktuální implementace asynchronní API, které databázové operace provádí mimo hlavní JavaScript vlákno a neblokuje tak vykreslování uživatelského rozhraní. To je kriticky důležité zejména při načítání rozsáhlejší zdravotní historie. (Expo Documentation, 2024c)

Modul `expo-local-authentication` zprostředkovává komunikaci s biometrickými systémy operačního systému. Na platformě iOS se jedná o `LocalAuthentication` Framework obsluhující technologie FaceID a TouchID, na Androidu o ekvivalentní BiometricPrompt API. Klíčovým bezpečnostním principem je, že aplikace nikdy nezíská přístup k biometrickým datům samotným — operační systém zajišťuje celý proces ověření izolovaně ve vyhrazeném bezpečnostním prostoru zařízení. (Apple Inc, 2024b; Expo Documentation, 2024g) Aplikace eZdraví využívá tento modul při každém spuštění: pokud má zařízení funkční biometrii, je před zobrazením obsahu aplikace vždy požadováno ověření. (Apple Inc, 2024b)

Modul `expo-secure-store` slouží k bezpečnému ukládání konfiguračních dat. Na iOS ukládá hodnoty do systémové klíčenky (Keychain Services), na Androidu do Keystore. Oba mechanismy šifrují data na hardwarové úrovni a vážou je ke konkrétnímu zařízení. Extrahovat hodnoty bez znalosti klíče nebo mimo dané zařízení není standardními prostředky možné. V aplikaci eZdraví je prostřednictvím tohoto modulu uchováván identifikátor aktivního uživatelského profilu, který určuje, která sada zdravotních záznamů je aktuálně zobrazována. Alternativní řešení pomocí `AsyncStorage`, které bylo v dřívějších verzích aplikace použito, tato data ukládalo jako nešifrovaný text, a proto bylo nahrazeno. (Expo Documentation, 2024d)

Moduly `expo-print` a `expo-sharing` zajišťují export zdravotní dokumentace. `expo-print` generuje PDF dokument z HTML šablony vykreslené systémovým renderovacím strojem WebKit/Blink a `expo-sharing` předá výsledný soubor systémovému dialogu sdílení, jenž umožňuje odeslání přes e-mail, uložení do cloudového úložiště nebo tisk. (Expo Documentation, 2024e)

Expo Router a navigační model

Pro navigaci v aplikaci eZdraví byl zvolen Expo Router místo tradičně používané knihovny React Navigation. Obě řešení jsou si technicky příbuzná. Expo Router je postavený přímo na základech React Navigation, liší se však přístupem k definici navigační struktury. Zatímco React Navigation vyžaduje ruční registraci každé obrazovky a explicitní konfiguraci navigačního stromu v kódu, Expo Router přebírá konvenci souborového routování, kde adresářová struktura složky `app/` přímo odpovídá navigačním cestám aplikace bez nutnosti dodatečné konfigurace. Pro projekt eZdraví, kde je navigační struktura přehledná a pevně daná, tento přístup snižuje množství boilerplate kódu a zajišťuje, že navigační logika je čitelná přímo ze struktury projektu. (Expo Documentation, 2024f)

Expo Router implementuje souborové routování, v němž struktura adresáře `app/` přímo odpovídá navigačním cestám aplikace. Každý `soubor.tsx` v tomto adresáři automaticky registruje odpovídající obrazovku. Skupiny obrazovek sdílející společný layout jsou organizovány do složek s názvem v kulatých závorkách. Složka `(tabs)` například obsahuje všechny obrazovky přístupné z hlavní navigační lišty a jejich společný layout soubor definující vzhled a chování lišty. Obrazovky mimo tuto skupinu, například detail diagnózy nebo formulář pro přidání záznamu jsou zařazeny do zásobníkového navigátoru (Stack), jenž zajišťuje standardní chování přechodu tam a zpět. (Expo Documentation, 2024f)

Předávání parametrů mezi obrazovkami je řešeno prostřednictvím URL parametrů, k nimž přistupuje hook `useLocalSearchParams`. Díky tomu je například detail diagnózy dostupný

z domovské obrazovky, z kalendáře i z detailu návštěvy vždy pod stejnou cestou `/disease-detail?id=X` a aplikace nepotřebuje spravovat globální stav pro navigaci.

Reaktivní model a správa stavu

Uživatelské rozhraní React Native je deklarativní. Vývojář nepopisuje, jak rozhraní změnit, ale jak má rozhraní vypadat pro daný stav. Změna stavu automaticky spustí překreslení příslušné části stromu komponent. Správa stavu v aplikaci eZdraví je realizována výhradně prostřednictvím vestavěných hooks: `useState` pro lokální stav komponenty (například hodnoty formuláře), `useEffect` pro vedlejší efekty spouštěné po prvním vykreslení obrazovky a `useFocusEffect` z Expo Routeru pro opakované načtení dat při každém návratu na obrazovku, což je v aplikaci pracující s databází kriticky důležité, protože záznamy mohly být mezitím přidány nebo upraveny z jiné obrazovky. (React, 2024b)

3.4 Lokální databáze a ORM

Volba datového úložiště je v mobilní aplikaci pracující se zdravotními záznamy jedním z nejzásadnějších architektonických rozhodnutí. Ovlivňuje nejen výkon a spolehlivost aplikace, ale také bezpečnost citlivých dat a možnosti budoucího rozšíření.

3.4.1 Relační versus dokumentový datový model

Moderní mobilní aplikace staví na dvou základních modelech perzistentního ukládání dat. Relační databáze organizují data do tabulek s pevně definovanou strukturou a explicitními vztahy vyjádřenými cizími klíči. Dokumentové databáze (označované jako NoSQL) ukládají data jako volně strukturované dokumenty, typicky ve formátu JSON, bez nutnosti předem definovat schéma.

Pro zdravotní záznamy s pevně definovanými vazbami mezi entitami je relační model přirozenou volbou. Nemoc musí být propojena s konkrétním uživatelem a může být spojena s více návštěvami lékaře a více léčebnými režimy. Léčebný režim musí referovat na konkrétní krabičku v lékárničce a každý záznam o užití léku musí referovat na příslušný léčebný režim. Porušení jakékoli z těchto vazeb by vedlo k nekonzistentním datům. Například záznamu o užití léku bez navázané lékové zásoby, která by měla být snížena. Relační databáze tento typ referenční integrity vynucuje automaticky prostřednictvím omezení cizích klíčů (FOREIGN KEY constraints), zatímco dokumentová databáze by tuto logiku vyžadovala implementovat ručně v aplikačním kódu. (SQLite, 2024a)

3.4.2 SQLite v mobilním prostředí

SQLite je souborový relační databázový systém, v němž celá databáze reziduje v jediném souboru na disku. Na rozdíl od klient-server databází (PostgreSQL, MySQL) nevyžaduje žádný serverový proces. Knihovna je přilinkována přímo do spustitelného souboru aplikace a databázové operace probíhají v procesu aplikace samotné. Právě proto je SQLite de facto standardem pro lokální datové úložiště v mobilních aplikacích a je integrován přímo v operačním systému iOS i Android. (SQLite, 2024b)

Ačkoliv SQLite teoreticky plně podporuje ACID transakce, z důvodu omezení některých lokálních SQLite driverů pro React Native a zvolené ORM vrstvy nejsou v aplikaci eZdraví využívány striktní databázové transakce na úrovni enginu. Zajištění konzistence dat při provádění složených operací (například vytvoření záznamu o užití léku v tabulce `medication_logs` a současný odečet z lékárničky v tabulce `inventory`) je řešeno na aplikační úrovni. Bezpečnost takové operace je zajištěna preventivní validací v rámci sekvenčního asynchronního bloku (async/await): aplikace nejprve zkontroluje aktuální stav zásob a teprve poté asynchronně zapíše log o užití léku a provede aktualizaci příslušného záznamu. Tento přístup minimalizuje riziko desynchronizace dat a je podrobněji rozebrán v kapitole 8.2. (SQLite, 2024c)

Konkrétní schéma databáze aplikace eZdraví je tvořeno sedmi tabulkami: `users`, `diseases`, `visits`, `visit_documents`, `inventory`, `medication_plans` a `medication_logs`. Tabulka `inventory` reprezentuje fyzické krabičky léků se svými zásobami a je záměrně oddělena od tabulky `medication_plans`, která popisuje léčebný režim — tedy jak, kdy a v jakém množství se lék z konkrétní krabičky užívá. Toto oddělení umožňuje jednu krabičku léku použít pro více léčebných režimů (například při dvou souběžných onemocněních) a zároveň evidovat historii spotřeby nezávisle na aktuálním stavu zásoby.

3.4.3 ORM a Drizzle

Přímá práce s databází prostřednictvím SQL dotazů psaných jako textové řetězce přináší v aplikaci psané v TypeScriptu dvě nevýhody. Zaprvé, SQL dotazy nejsou typově kontrolovány kompilátorem — překlep v názvu sloupce nebo předání hodnoty nesprávného typu je zachycen až za běhu aplikace, nikoli při kompilaci. Zadruhé, změna databázového schématu vyžaduje ruční dohledání a úpravu všech dotazů napříč kódovou základnou.

ORM (Object-Relational Mapping) je softwarová vrstva mapující databázové tabulky na objekty programovacího jazyka. Vývojář pracuje s typově bezpečnými objekty a ORM na základě jejich definice generuje SQL dotazy automaticky.

V aplikaci eZdraví byl zvolen Drizzle ORM, jehož odlišující vlastností oproti alternativám (TypeORM, Prisma) je nulová abstrakce za běhu aplikace. Drizzle sestavuje SQL dotazy s minimálním výkonnostním overhead a veškerá typová kontrola probíhá v době kompilace. Schéma databáze je definováno v souboru `schema.ts` jako TypeScriptové objekty, z nichž kompilátor automaticky odvozuje typy pro všechny operace čtení i zápisu. Pokus o přístup k neexistujícímu sloupci nebo přiřazení hodnoty nesprávného typu je tak zachycen ještě před spuštěním aplikace. (Product by Drizzle Team, b. r. a)

Systém migrací Drizzle ORM řeší problém vývoje databázového schématu v čase. Každá změna schématu — přidání sloupce, vytvoření nové tabulky — je zachycena jako migrační soubor obsahující odpovídající SQL příkazy. Při každém spuštění aplikace hook `useMigrations` porovná aktuální verzi schématu v databázi na zařízení s migracemi uloženými v kódu aplikace a automaticky aplikuje případné rozdíly. Uživatelé, kteří instalují aktualizaci aplikace s upraveným schématem, tak o svá data nepřijdou. Databáze je bezpečně aktualizována při prvním spuštění nové verze. (Product by Drizzle Team, b. r. b)

3.5 Proces vývoje

Vývoj aplikace eZdraví probíhal iterativně bez předchozí návrhové fáze. Vzhledem k tomu, že jsem byl primárním uživatelem aplikace a zároveň měl přesnou představu o požadované funkcionalitě, bylo upuštěno od formálního návrhu rozhraní. Vývoj začal přímo implementací. Tento přístup odpovídá agilní metodice, v níž specifikace a implementace probíhají souběžně a požadavky se upřesňují na základě průběžné zpětné vazby místo předem fixované dokumentace. (Sommerville, 2016)

3.5.1 Implementační fáze

Vývoj probíhal v prostředí Visual Studio Code s využitím Expo Go na fyzickém zařízení iPhone, ale i na virtuálním Apple Simulatoru pro průběžné testování. Výhodou tohoto přístupu bylo, že designová a implementační rozhodnutí vznikala souběžně v reakci na reálné chování aplikace, zejména v oblasti správy dat a výběru nativních modulů, kde se řada technických omezení projevila až při praktickém testování. Výsledné rozhraní je proto přímým produktem tohoto iterativního procesu.

Funkční testování aplikace probíhalo průběžně během vývoje na reálném zařízení. Důraz byl kladen zejména na testování hraničních stavů správy databáze, konkrétně chování aplikace při nulovém stavu zásoby léku, při smazání záznamu s vazbou na jiné entity a při konfliktu datumových rozsahů léčebného režimu a navázané diagnózy. Tyto scénáře jsou z pohledu integrity zdravotních dat kritické a jejich neošetření by vedlo k nekonzistentním nebo zavádějícím zobrazením v historii záznamu.

4 Principy návrhu uživatelského rozhraní

Uživatelský zážitek (User Experience, UX) a uživatelské rozhraní (User Interface, UI) představují dva vzájemně propojené, ale odlišné aspekty designu digitálních produktů. Zatímco UI se zabývá vizuálními a interaktivními prvky, se kterými uživatel přímo interaguje, UX zahrnuje celkovou zkušenost uživatele s produktem, včetně emočních a praktických aspektů této interakce (Norman Don, Nielsen Jakob, 1998).

4.1 Definice a vztah UX a UI

Don Norman, který poprvé zavedl termín „user experience" v 90. letech během svého působení ve společnosti Apple, společně s Jakobem Nielsenem vymezují UX jako souhrn veškerých aspektů toho, jak koncový uživatel přichází do styku se společností, jejími službami a produkty. (Norman & Nielsen, 1998) Tato definice zdůrazňuje komplexnost UX, která přesahuje pouhé technické aspekty a zahrnuje i emocionální a kontextové faktory.

UI design naproti tomu představuje vizuální a interaktivní vrstvu produktu, přes kterou uživatelé komunikují s jeho funkcemi. Podle vzdělávací organizace Interaction Design Foundation tvoří UI design pomyslný most mezi uživatelem a technologií, přičemž se zaměřuje na předvídání toho, co uživatelé potřebují udělat, a zajišťuje, aby rozhraní obsahovalo prvky, které k těmto akcím snadno přístupným způsobem vedou (Interaction Design Foundation, 2024a).

Rozdíl mezi UX a UI hodnotou lze dobře ilustrovat na domovské obrazovce aplikace eZdraví. UI vrstva vizuálně odděluje léky na pravidelné užívání od přípravků užívaných podle potřeby. UX hodnota tohoto rozhodnutí však spočívá jinde. Nemocný člověk ve stresu nebo bolesti nemusí procházet celou historií medikace ani si pamatovat dávkovací schéma. Aplikace mu okamžitě prezentuje pouze akce relevantní pro daný den, čímž snižuje kognitivní zátěž v momentě, kdy je uživatel nejzranitelnější.

4.2 Principy použitelnosti a přístupnosti

Podobně jako Maslowova hierarchie potřeb, která organizuje lidské potřeby od základních fyziologických po seberealizaci, existuje i hierarchie uživatelských potřeb v digitálním prostředí, kterou definoval Walter (2011). Na nejnižší úrovni této hierarchie stojí funkčnost, což znamená, že produkt musí především spolehlivě plnit svůj základní účel. Nad ní se nachází úroveň spolehlivosti, která zajišťuje stabilní a konzistentní chování produktu. Třetí úroveň představuje použitelnost, tedy snadnost používání a učení se práci s produktem. Na nejvyšší úrovni této hierarchie stojí příjemnost, která zajišťuje, že produkt vyvolává u uživatele pozitivní emocionální odezvu. Hierarchie naznačuje, že nejprve musí být splněny základní požadavky na funkčnost a spolehlivost, než se můžeme zabývat vyššími úrovněmi uživatelské zkušenosti jako je použitelnost a příjemnost používání.

4.2.1 Nielsenovy heuristiky

Jakob Nielsen definoval deset základních principů uživatelského rozhraní, známých jako "heuristiky použitelnosti", které jsou pravidelně aktualizovány a publikovány v rámci výzkumné skupiny Nielsen Norman Group (Nielsen, 1994). Tyto principy začínají viditelností stavu systému,

která zajišťuje, že uživatel vždy ví, co se v systému děje. Následuje shoda mezi systémem a reálným světem, díky níž systém komunikuje jazykem uživatele a používá známé koncepty. Uživatelská kontrola a svoboda jako třetí princip poskytuje možnost vrátit se zpět nebo zrušit nežádoucí akci. Čtvrtým principem jsou konzistence a standardy, které vyžadují jednotné používání prvků a postupů v celém systému. Pátý princip, prevence chyb, se snaží předcházet problémům dříve, než nastanou.

Druhá polovina pravidel začíná šestým principem, rozpoznáním místo vzpomínání, který zdůrazňuje důležitost viditelných možností namísto nutnosti pamatovat si postupy. Sedmý princip flexibility a efektivity použití zajišťuje, že systém je vhodný jak pro začátečníky, tak pro pokročilé uživatele. Osmý princip estetického a minimalistického designu klade důraz na jednoduchost a přehlednost rozhraní. Devátý princip se věnuje pomoci uživatelům při rozpoznání, pochopení a zotavení se z chyb prostřednictvím srozumitelných chybových hlášek. Poslední, desátý princip zdůrazňuje potřebu kvalitní nápovědy a dokumentace, přestože by systém měl být v ideálním případě použitelný i bez nich. Tyto heuristiky společně tvoří komplexní rámec pro vytváření uživatelsky přívětivých a efektivních rozhraní.

Návrh aplikace eZdraví se s těmito principy vědomě vyrovnává na několika úrovních. Princip viditelnosti stavu systému je uplatněn v kartotéce návštěv, kde barevné kódování ikon okamžitě komunikuje stav vyšetření. Modrá ikona označuje návštěvu, která již proběhla, oranžová signalizuje plánovanou. Podobně v lékárničce uživatel vždy vidí aktuální zůstatek zásoby v krabičce, aniž by musel cokoli dohledávat. Princip shody mezi systémem a reálným světem se projevuje v důsledném oddělení „Lékárničky" — fyzických krabiček léků, které má pacient doma — od „Plánů užívání", které reprezentují dávkovací režim předepsaný lékařem. Toto rozlišení přesně kopíruje mentální model, který pacienti v reálném světě přirozeně používají.

Princip prevence chyb a zotavení z chyb je v aplikaci řešen na více úrovních. Při pokusu o zaznamenání užití léku aplikace nejprve ověří aktuální stav zásoby v databázi, pokud by odečet způsobil záporný zůstatek, operace je zablokována. Pokud zásoba klesne na nulu, aplikace nezobrazí chybové hlášení, ale nabídne dialogové okno s dotazem, zda si uživatel přeje prázdnou krabičku přesunout do historie. Aplikace je navíc chráněna biometrickou autentizací přes FaceID nebo TouchID, která zabraňuje nechtěným zásahům cizí osoby do zdravotních záznamů. Princip rozpoznání místo vzpomínání je uplatněn při přidávání nové návštěvy lékaře. Uživatel nemusí ručně vypisovat název diagnózy, ale vybírá ji z modálního menu již existujících nemocí vedených v aplikaci. Konečně princip estetického a minimalistického designu se odráží v záměrném odstranění redundantních textových popisků z karet návštěv. Text „Stav: Plánovaná" byl odstraněn, protože tuto informaci plně a přehledněji komunikuje barva ikony — uvolněné místo bylo využito pro přehlednější zobrazení názvu oddělení.

4.2.2 Gestalt principy v UI designu

Gestalt principy, původně vyvinuté v psychologii vnímání, představují základní pravidla o tom, jak lidský mozek přirozeně organizuje vizuální prvky do smysluplných celků. Tyto postuláty, které poprvé definovali němečtí psychologové v první polovině 20. století, mají zásadní význam pro současný UI design. Jak uvádí expertní publikace organizace Interaction Design Foundation, lidský mozek má silnou tendenci vnímat objekty jako organizované vzory nebo celky spíše než jako jednotlivé, oddělené prvky (Interaction Design Foundation, 2024b).

V oblasti uživatelského rozhraní je princip blízkosti jedním z nejzákladnějších. Prvky, které jsou si fyzicky blízké, vnímáme jako související skupinu. Princip uzavření pak říká, že mozek dokáže vnímat skupinu prvků jako celek i bez explicitního ohraničení, pokud jsou dostatečně vizuálně svázány. V aplikaci eZdraví jsou oba principy uplatněny prostřednictvím komponent Cards. Vizuální karty se zaoblenými rohy a jemným stínem, do nichž jsou seskupeny informace o jednom léku nebo návštěvě. Uživatel okamžitě chápe, že název léku, čas užití a tlačítko „Vzít" tvoří jeden funkční celek, aniž by bylo nutné použít tlusté oddělovací čáry nebo nadměrné mezery.

Princip podobnosti říká, že prvky sdílející vizuální charakteristiky jako barvu, tvar nebo velikost vnímáme jako příbuzné a funkčně související. V aplikaci eZdraví jsou všechna hlavní potvrzovací tlačítka (uložení záznamu, odškrtnutí léku) sjednocena primární barvou aplikace. Naopak kritické a urgentní akce — léky kategorie SOS určené pro akutní stavy nebo operace mazání záznamů jsou konzistentně označeny výstražnou červenou barvou. Uživatel si tak podvědomě a rychle zafixuje vizuální jazyk aplikace bez nutnosti číst popisky.

Princip symetrie odráží přirozenou tendenci vnímat symetrické uspořádání jako harmonické, což se uplatňuje při rozvržení prvků na obrazovce. Princip společného osudu říká, že prvky pohybující se stejným směrem nebo měnící se stejným způsobem vnímáme jako související. Tento jev je obzvláště důležitý při navrhování animací a přechodů mezi obrazovkami v moderních mobilních rozhraních.

5 Návrh aplikace

Návrh uživatelského rozhraní aplikace eZdraví probíhal iterativním a agilním způsobem formou přímé implementace do kódu (tzv. code-first přístup). Namísto tvorby statických grafických prototypů byl design od počátku utvářen a validován přímo ve vývojovém prostředí pomocí frameworku React Native a platformy Expo. Tento postup, podpořený funkcí okamžitého propisu změn (Fast Refresh), umožnil souběžné řešení vizuální identity i funkční logiky rozhraní. Hlavní výhodou tohoto přístupu byla možnost okamžitého testování ergonomie a uživatelské zkušenosti (UX) na reálném fyzickém zařízení, což vedlo k efektivnější optimalizaci navigačních prvků a celkové odezvy aplikace. Výsledná podoba rozhraní je tak produktem průběžného testování a plynulých úprav reagujících na praktické zkušenosti z vývoje.

5.1 Funkce aplikace

Aplikace eZdraví vznikla jako odpověď na konkrétní problém: pacienti v České republice nemají standardně dostupný nástroj, který by jim umožnil vést vlastní zdravotní dokumentaci v přehledné a strukturované podobě. Existující řešení se zaměřují buď výhradně na připomínání léků bez vazby na diagnózu, nebo na sběr biometrických dat bez správy dokumentace. Aplikace eZdraví tyto dvě dimenze propojuje a doplňuje je o možnost exportu dokumentace pro potřeby lékaře. Aplikace řeší čtyři hlavní oblasti.

Evidence zdravotní historie. Uživatel může zaznamenat prodělané i probíhající diagnózy včetně jejich typu (akutní nebo chronická), datového rozsahu, poznámek lékaře a navazující fotografické dokumentace z návštěv.

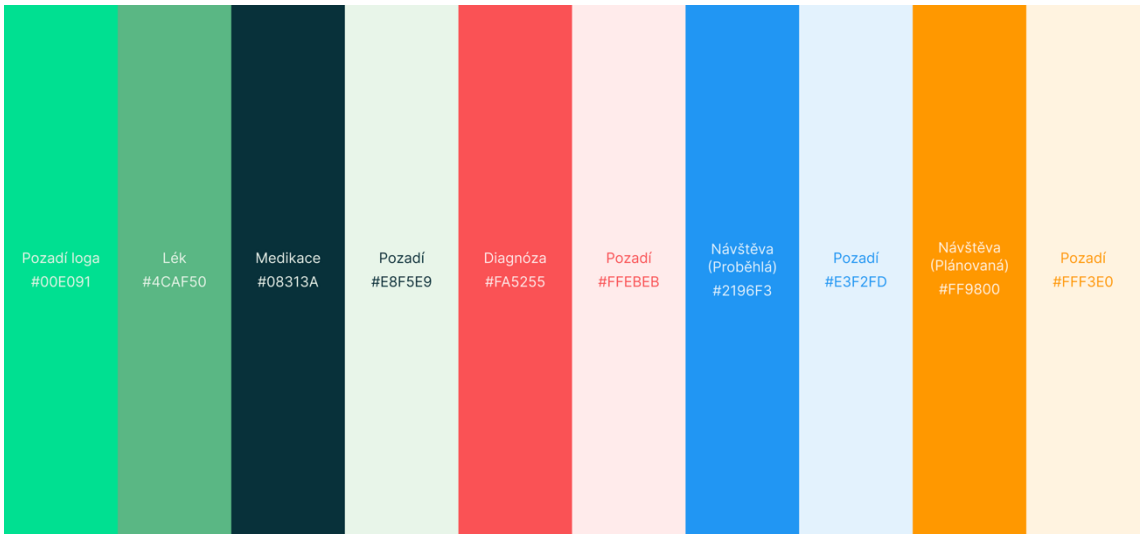
Správa léčebných režimů. Aplikace odděluje fyzickou zásobu léků v domácí lékárničce od léčebného plánu předepsaného lékařem. Při každém záznamu užití léku dochází k automatickému odpočtu zásoby a systém upozorní na její vyčerpání.

Evidence návštěv lékaře. Uživatel zaznamenává plánované i proběhlé návštěvy s vazbou na konkrétní diagnózu, přičemž aplikace barevně odlišuje jejich stav a umožňuje přiložit fotografii lékařské zprávy.

Záloha a sdílení dokumentace. Kompletní zdravotní dokumentace je exportovatelná do formátu PDF pro předání lékaři, nebo do formátu JSON pro účely zálohy a přenosu na jiné zařízení.

5.2 Vizuální identita a design systém

Vizuální identita aplikace eZdraví je postavena na neutrální zelené jako primární barvě. Zelená (#4CAF50) odkazuje na prostředí zdravotní péče, dosahuje dostatečného kontrastu vůči bílému pozadí a působí důvěryhodně a klidně — vlastnosti, které jsou v kontextu zdravotnické aplikace žádoucí. Sekundárně jsou v rozhraní využívány červená pro záznamy diagnóz a onemocnění a modrá pro záznamy návštěv lékaře. Barevné kódování kategorií je konzistentně uplatněno napříč celou aplikací — na domovské obrazovce, v kalendářním přehledu i v detailních pohledech — a umožňuje uživateli okamžitou vizuální orientaci v typu záznamu bez nutnosti číst popisky.



Obr. 1: Barvy

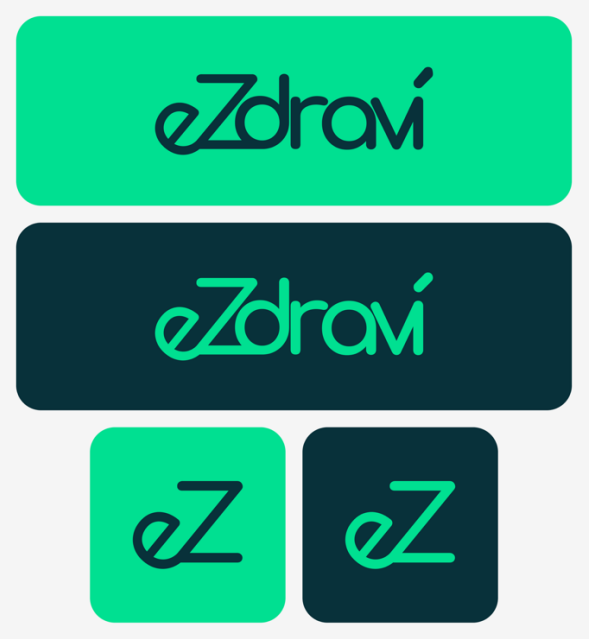
Zdroj: vlastní zpracování dle Zlatuška (2025)

Typografie vychází z výchozího systémového fontu příslušné platformy, což zajišťuje maximální čitelnost bez závislosti na externích zdrojích. Ikonografie je řešena prostřednictvím sady MaterialCommunityIcons, jejíž ikony jsou uživatelům systémů Android i iOS dostatečně povědomé. (Expo Documentation, 2024i) Navigační lišta je realizována jako plovoucí panel se zaoblenými rohy umístěný v dolní části obrazovky. Panel obsahuje pět hlavních sekcí označených ikonami: Domů, Časová osa, Lékárnička, Kartotéka a Účet. Plovoucí design lišty byl zvolen záměrně, odlišuje aplikaci od generického systémového vzhledu a zároveň nijak neomezuje ergonomiku ovládání palcem.

Základním prvkem vizuální identity je autorské typografické logo, které bylo navrženo ve vektorovém grafickém editoru Adobe Illustrator. Logo je definováno moderním bezpatkovým (sans-serif) písmem, jehož čisté a geometrické linie korespondují s požadavky na přehlednost a důvěryhodnost zdravotnických aplikací.

Ústředním grafickým motivem loga je vytvořená typografická ligatura (slitina) počátečních písmen „e“ a „Z“. Spojení malého písmene „e“, které odkazuje na elektronizaci a moderní přístup (eHealth), a dominantního velkého „Z“, symbolizujícího samotné zdraví a zdravotní záznamy, do jednoho plynulého celku není pouze estetickým prvkem. Tato vizuální zkratka metaforicky vyjadřuje hlavní cíl aplikace: plynulé a bezbariérové propojení digitálního světa s osobní zdravotní péčí.

Logo bylo od počátku navrženo ve dvou variantách – plné textové znění pro domovskou obrazovku a zkrácená varianta (ikona) pro účely ikony aplikace na ploše (app icon) a favicony. Tento minimalistický přístup zajišťuje, že si logo zachovává vysokou čitelnost a rozpoznatelnost bez ohledu na to, zda je zobrazeno na velkém displeji tabletu, nebo jako drobná ikona v notifikační liště mobilního telefonu.



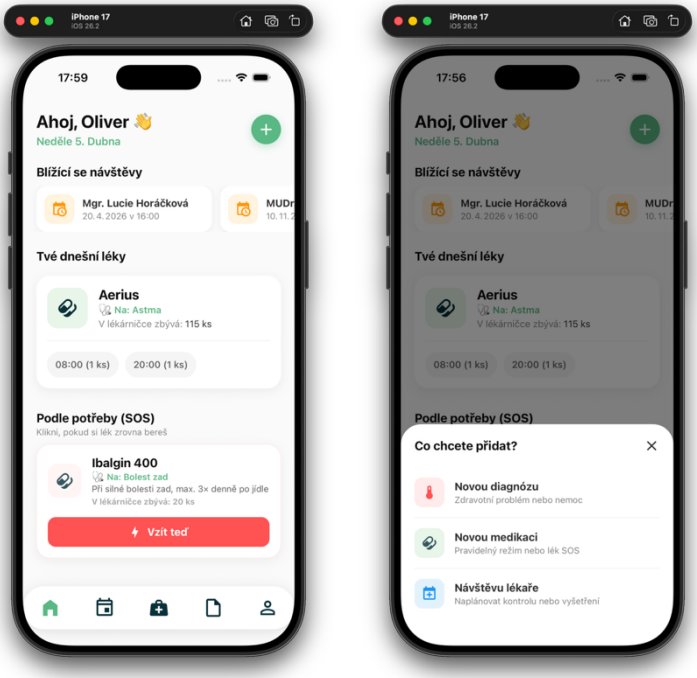
Obr. 2: Vizuální identita
Zdroj: vlastní zpracování dle Zlatuška (2025)

5.3 Popis uživatelského prostředí

První kontakt uživatele s aplikací je klíčový pro její následné používání a celkový uživatelský zážitek (UX). Z tohoto důvodu byl navržen přehledný onboardingový proces, který uživatele po prvním spuštění seznámí s hlavními pilíři aplikace: elektronickou zdravotní kartou, chytrou lékárničkou a generováním reportů. Pro zachování Nielsenova principu "viditelnosti stavu systému" je ve spodní části obrazovky umístěn indikátor postupu (navigační tečky), díky kterému uživatel přesně ví, v jaké fázi tutoriálu se nachází.

Celý proces je zakončen obrazovkou pro vytvoření uživatelského profilu. V souladu s principem minimalistického designu a snižováním kognitivní zátěže je vyžadováno pouze zadání jména či přezdívky. Aplikace nevyžaduje složité registrace přes e-mail ani tvorbu hesel, což maximalizuje šanci, že uživatel proces úspěšně dokončí a začne aplikaci okamžitě využívat. (viz Příloha 1)

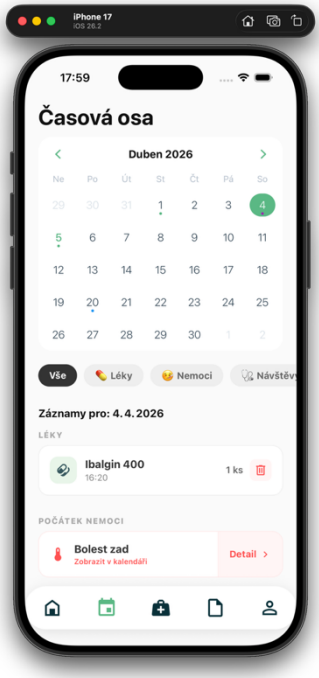
Domovská obrazovka plní funkci osobního dashboardu. Po otevření aplikace je uživateli zobrazeno personalizované přivítání s jeho jménem a aktuálním datem. Obrazovka je rozdělena do tematických sekcí: blížící se návštěvy lékaře, dnešní medikace s konkrétními časy dávek a léky předepsané na základě potřeby (SOS). Každý lék je v sekci dnešní medikace prezentován jako karta zobrazující název, navázanou diagnózu, zbývající zásobu v lékárničce a naplánované časy dávek jako interaktivní tlačítka. Stisknutím tlačítka s časem uživatel záznam dávky potvrdí a zásoba v lékárničce se automaticky sníží. SOS léky jsou prezentovány odděleně s výrazným tlačítkem „Vzít teď", které umožňuje okamžitý zápis bez nutnosti procházet formuláře. V pravém horním rohu je plovoucí tlačítko pro přidání nového záznamu, jež po stisknutí zobrazí modální nabídku se třemi volbami: nová diagnóza, nová medikace a návštěva lékaře.



Obr. 3: Domovská obrazovka + Modální okno pro přidávání záznamů

Zdroj: vlastní zpracování dle Zlatuška (2026)

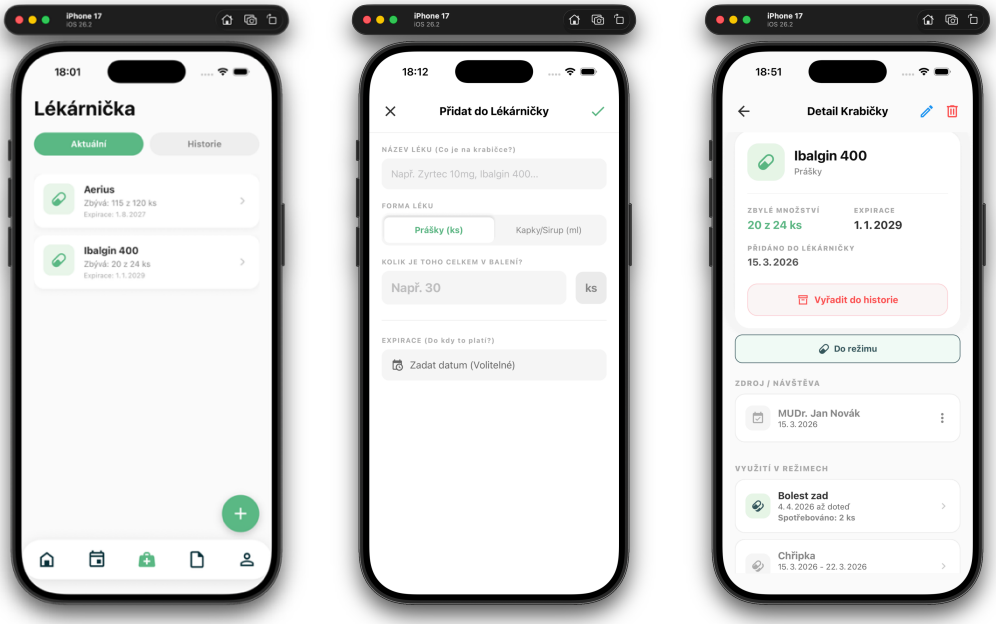
Obrazovka Časová osa poskytuje kalendářní přehled zdravotních záznamů. V horní části je zobrazen měsíční kalendář, v němž jsou dny obsahující záznamy označeny barevnými tečkami odpovídajícími kategorii záznamu. Zelená pro léky, červená pro diagnózy, modrá pro návštěvy. Po výběru konkrétního dne jsou pod kalendářem zobrazeny všechny záznamy příslušného dne rozdělené do kategorií. Filtrační lišta umožňuje omezit zobrazení na vybranou kategorii. Tato obrazovka plní roli průřezového přehledu a umožňuje uživateli sledovat průběh léčby v čase bez nutnosti procházet jednotlivé detailní záznamy.



Obr. 4: Obrazovka časové osy (Kalendář)

Zdroj: vlastní zpracování dle Zlatuška (2026)

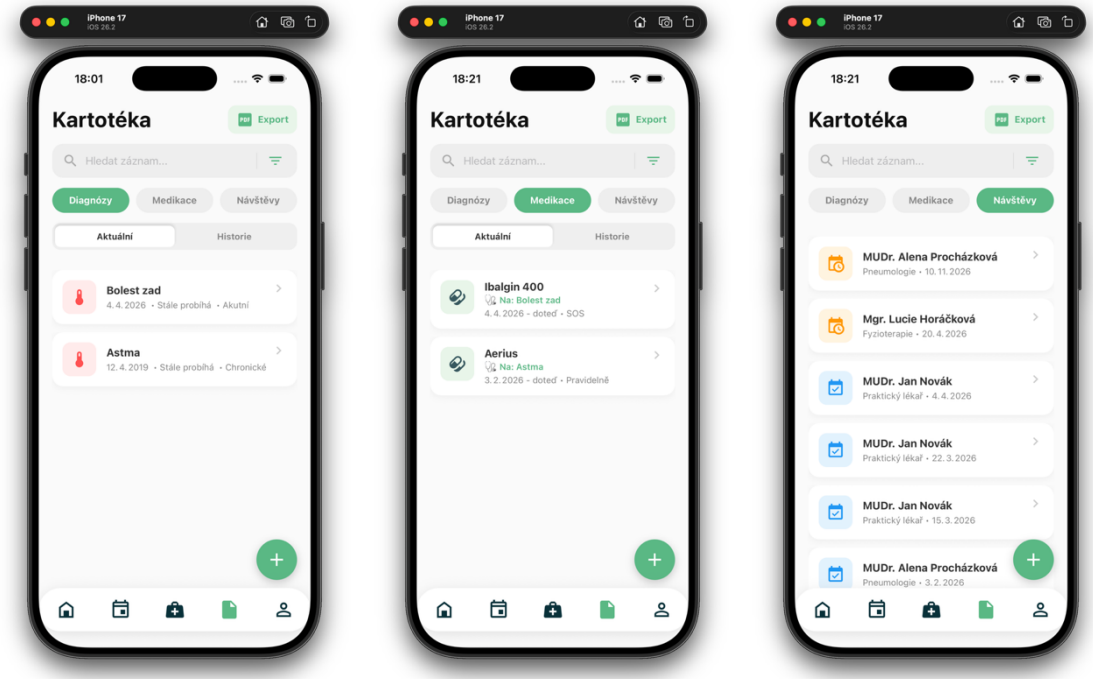
Obrazovka Lékárnička slouží ke správě fyzických zásob léků. Záznamy jsou rozděleny na záložky Aktuální a Historie. Každá krabička je prezentována kartou zobrazující název léku, formu (tablety nebo kapky/sirup), zbývající a celkové množství a datum expirace. Pokud jsou zásoby vyčerpány, krabičku je možné přeřadit do Historie a označit ji jako vyřazenou. Přidání nové krabičky je dostupné přes plovoucí tlačítko.



Obr. 5: Obrazovka lékárničky

Zdroj: vlastní zpracování dle Zlatuška (2026)

Obrazovka Kartotéka je centrálním místem pro správu zdravotní dokumentace a organizuje záznamy do tří záložek: Diagnózy, Medikace a Návštěvy. V každé záložce jsou záznamy dále filtrovatelné na Aktuální a Historické. Diagnózy jsou zobrazeny s datem vzniku, typem (akutní nebo chronické) a stavem průběhu. Medikace zobrazují navázanou diagnózu, datový rozsah léčebného režimu a typ dávkování (pravidelné nebo SOS). Návštěvy jsou řazeny chronologicky a barevně odlišeny podle stavu — oranžová pro plánované, modrá pro proběhlé. Na obrazovce Kartotéka je přístupné tlačítko pro export celé zdravotní dokumentace do formátu PDF. (viz Příloha 2)

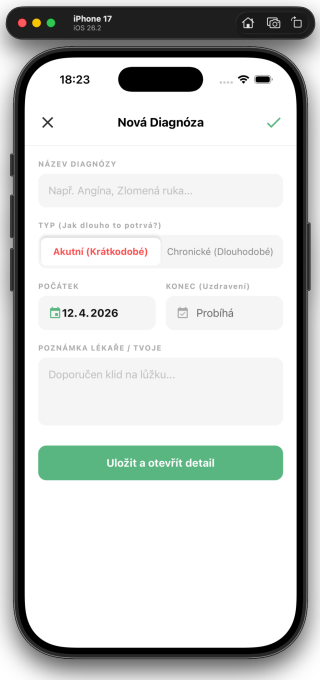


Obr. 6: Obrazovka kartotéka

Zdroj: vlastní zpracování dle Zlatuška (2026)

Formuláře pro přidávání záznamů jsou navrženy jako modální obrazovky přístupné z plovoucího tlačítka na domovské obrazovce. Každý z formulářů je tematicky strukturován a obsahuje pouze pole relevantní pro daný typ záznamu, čímž je naplněn princip minimalistického designu.

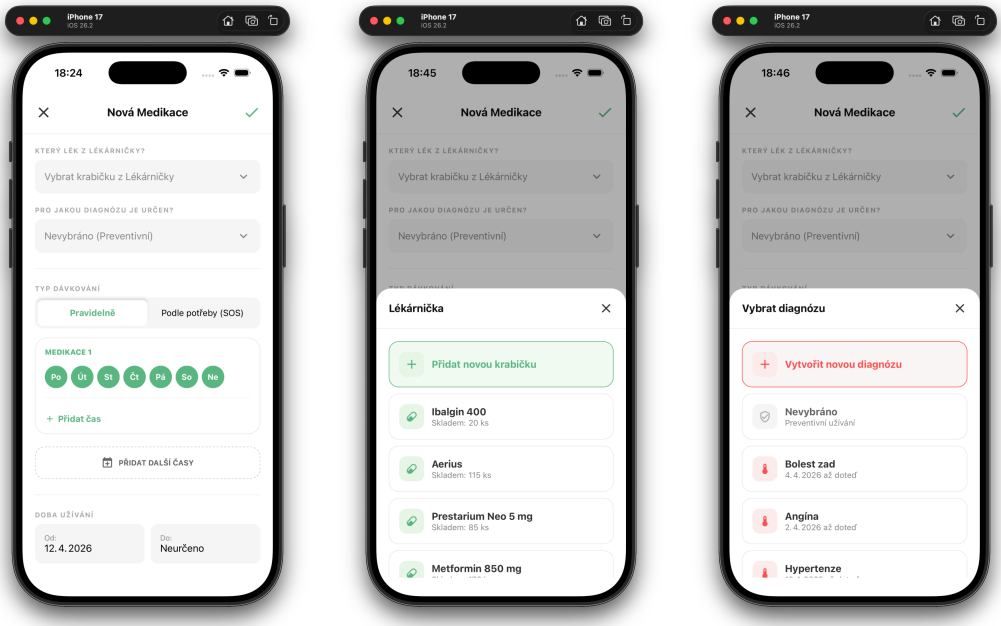
Formulář pro přidání nové diagnózy obsahuje pole pro název diagnózy, výběr typu průběhu (akutní nebo chronické), datový rozsah (počátek a konec nebo stav „Probíhá") a volitelnou poznámku lékaře. Výběr typu průběhu je řešen přepínačem se dvěma volbami, přičemž výchozí stav je nastaven na „Akutní (Krátkodobé)", který odpovídá nejčastějšímu případu užití. Formulář je zakončen tlačítkem „Uložit a otevřít detail", které po uložení záznamu automaticky přesměruje uživatele na detail nově vytvořené diagnózy — uživatel tak může okamžitě navázat přidáním medikace nebo návštěvy bez nutnosti záznamu vyhledávat.



Obr. 7: Formulář přidání nové diagnózy

Zdroj: vlastní zpracování dle Zlatuška (2026)

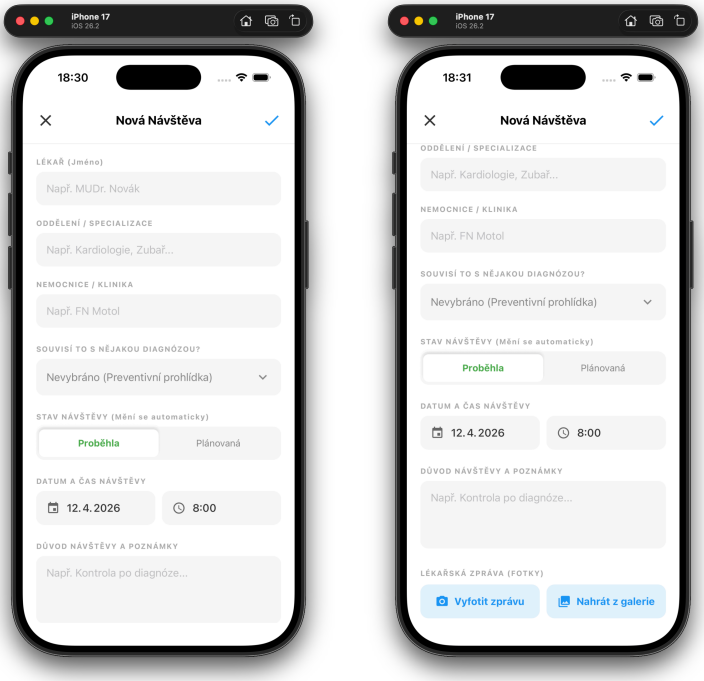
Formulář pro přidání nové medikace reflektuje konceptuální oddělení lékárničky a léčebných plánů popsané v kapitole 6.2. Uživatel nejprve vybere konkrétní krabičku z lékárničky a přiřadí medikaci k diagnóze. Typ dávkování lze zvolit jako pravidelné nebo podle potřeby (SOS). Pro pravidelné dávkování formulář umožňuje nastavit dny v týdnu a konkrétní časy podání, přičemž lze přidat více nezávislých časových plánů v rámci jednoho léčebného režimu. Datový rozsah léčby je volitelný — pole „Do" lze ponechat jako „Neurčeno" pro chronická onemocnění bez předem stanoveného konce léčby.



Obr. 8: Formulář přidání nové medikace

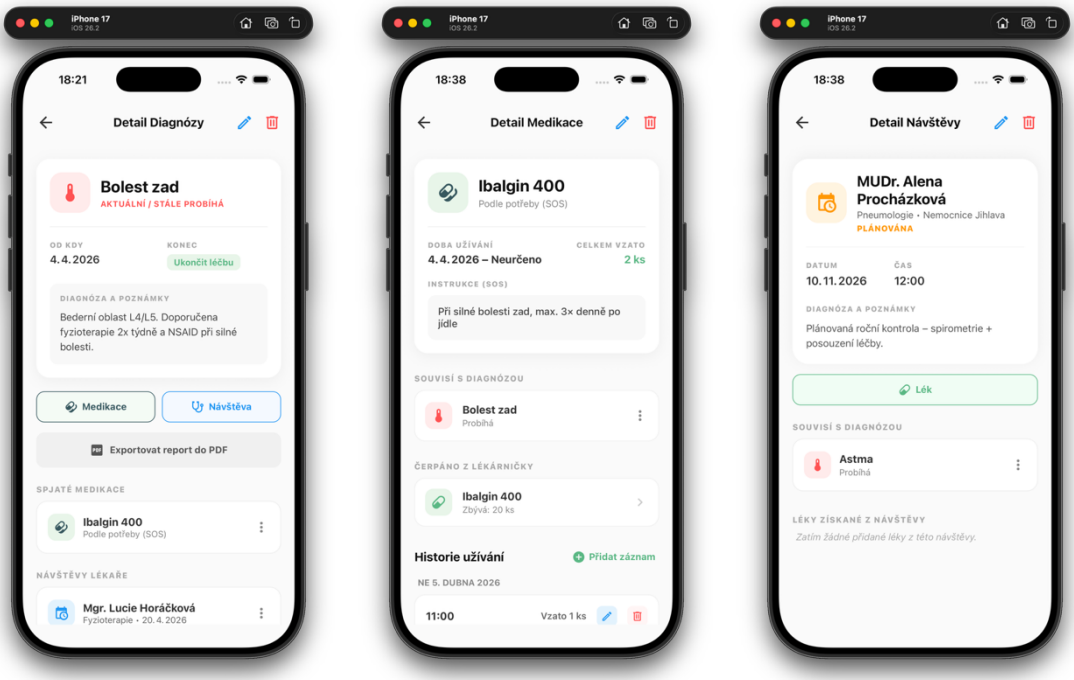
Zdroj: vlastní zpracování dle Zlatuška (2026)

Formulář pro přidání nové návštěvy lékaře obsahuje pole pro jméno lékaře, oddělení a nemocnici, propojení s diagnózou, stav návštěvy (proběhlá nebo plánovaná), datum a čas a poznámky. Stav návštěvy se přepíná přepínačem a mění se automaticky na základě zadaného data — pokud je datum v minulosti, systém předvyplní stav „Proběhla". Propojení s diagnózou je nepovinné, což umožňuje evidovat i preventivní prohlídky bez vazby na konkrétní onemocnění, pro které je připravena výchozí volba „Nevybráno (Preventivní prohlídka)".



Obr. 9: Formulář přidání nové návštěvy
Zdroj: vlastní zpracování dle Zlatuška (2026)

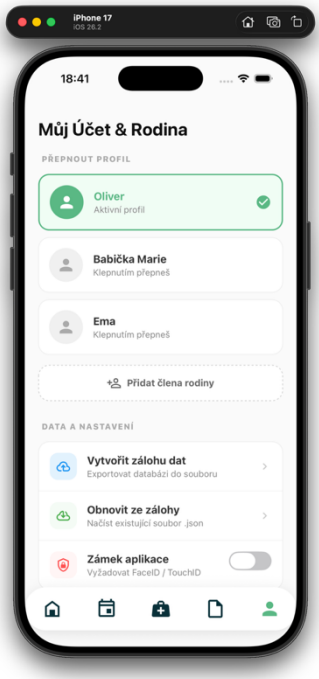
Detailní obrazovky jsou implementovány pro každou kategorii záznamu. Při návrhu zobrazení záznamů v Kartotéce byl kladen důraz na Nielsenův princip estetického a minimalistického designu. V rané fázi vývoje obsahovala karta návštěvy u lékaře kromě jména doktora a data také textovou informaci o stavu návštěvy („Plánovaná" nebo „Proběhla"). Z hlediska UX se však tato textová informace ukázala jako redundantní, neboť stav je již uživateli jasně komunikován prostřednictvím barvy a tvaru ikony v levé části karty (oranžová ikona hodin pro plánované, modrá ikona kalendáře s odškrtnutím pro proběhlé). Textový popisek byl proto odstraněn a uvolněné místo bylo využito pro zobrazení názvu lékařského oddělení (např. Ortopedie, Oční). Toto řešení odpovídá mentálním modelům pacientů, kteří si při zpětném vyhledávání v historii snáze vybaví odbornost pracoviště než konkrétní jméno ošetřujícího lékaře. Detail diagnózy zobrazuje kompletní informace o onemocnění včetně datového rozsahu, navázaných medikací a návštěv lékaře s možností přímé navigace do každého z propojených záznamů. Detail medikace obsahuje souhrn léčebného režimu a chronologicky řazenou historii užití léku s možností zpětného zápisu nebo opravy záznamu. Detail návštěvy zobrazuje datum, čas a poznámky lékaře a propojuje návštěvu s příslušnou diagnózou. Vzájemné provázání entit umožňuje procházet zdravotní dokumentaci jako síť kontextově propojených záznamů, nikoli jako izolované položky.



Obr. 10: Detail diagnózy, medikace a návštěvy

Zdroj: vlastní zpracování dle Zlatuška (2026)

Obrazovka Účet umožňuje správu uživatelských profilů. Aplikace podporuje více profilů v rámci jednoho zařízení, což umožňuje sledovat zdravotní záznamy více členů rodiny. Přepínání aktivního profilu je dostupné přímo na obrazovce. V sekci Nastavení jsou přístupné funkce pro export dat a nastavení biometrického zabezpečení (FaceID / TouchID).



Obr. 11: Obrazovka účet

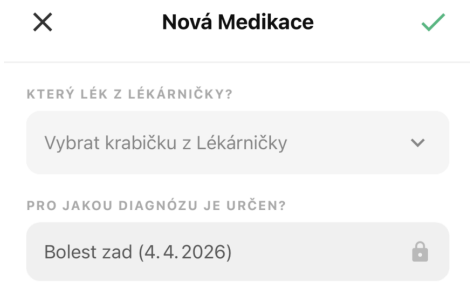
Zdroj: vlastní zpracování dle Zlatuška (2026)

5.4 Ochrana proti chybám uživatele (Error Prevention)

Jedním z klíčových principů návrhu uživatelského rozhraní (např. dle Nielsenových heuristik) je prevence chyb. Aplikace pro evidenci zdravotních záznamů musí být navržena tak, aby minimalizovala riziko zadání chybných nebo nekonzistentních dat, které by mohly vést ke zmatení pacienta. V rámci implementace byly aplikovány následující ochranné mechanismy:

Vizuální a funkční uzamčení kontextu

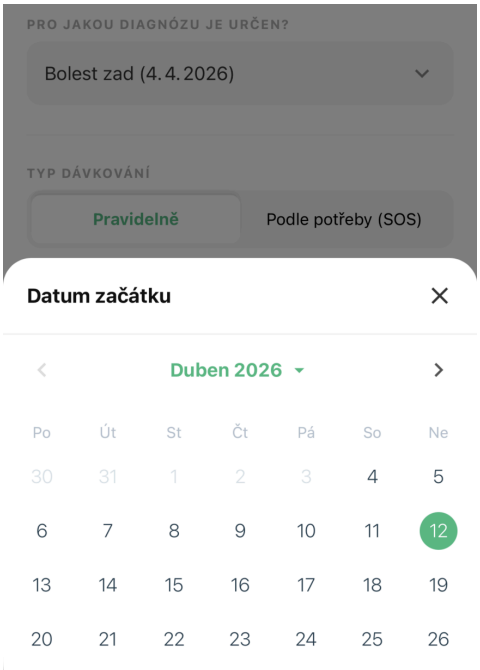
Pokud uživatel přistupuje k formuláři pro přidání léku nebo návštěvy přímo z detailu konkrétní nemoci, aplikace tuto vazbu automaticky předvyplní a výběrové menu uzamkne. Tím se předchází lidské chybě, kdy by uživatel omylem přiřadil záznam k nesprávné diagnóze. UI tuto skutečnost komunikuje vizuálně (např. ikonou zámku).



Obr. 12: Vizuální a funkční uzamčení kontextu
Zdroj: vlastní zpracování dle Zlatuška (2026)

Omezení datových vstupů v kalendáři

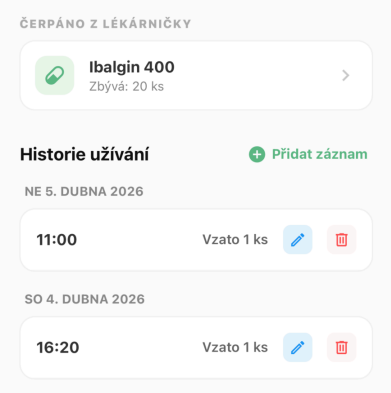
Při plánování léků nebo návštěv spojených s konkrétní nemocí kalendářové komponenty (např. u počátečních a koncových dat) dynamicky omezují výběr dnů na základě trvání dané diagnózy. Uživatel tak nemůže naplánovat užití léku na datum, které předchází vypuknutí nemoci.



Obr. 13: Omezení datových vstupů v kalendáři
Zdroj: vlastní zpracování dle Zlatuška (2026)

Inteligentní přepočet zásob při retroaktivních změnách

Aplikace počítá s tím, že uživatelé mohou dělat chyby při odškrtávání užitých léků. Systém umožňuje zpětné přidání nebo smazání záznamu v historii, přičemž automaticky a konzistentně přepočítává aktuální fyzický stav zásob v lékárničce, a to i s ohledem na status vyřazených a aktivních léků.



Obr. 14: Možnost přidávání záznamu v historii užívání léku
Zdroj: vlastní zpracování dle Zlatuška (2026)

Ochrana před vznikem "zombie" záznamů

Při ukončení léčby (přepnutí nemoci do stavu vyléčeno) systém provádí kaskádovou kontrolu. Pokud detekuje stále aktivní plány užívání léků navázané na tuto nemoc, proaktivně se uživatele dotáže, zda si přeje tyto plány také ukončit, čímž se zabraňuje zbytečným budoucím notifikacím.

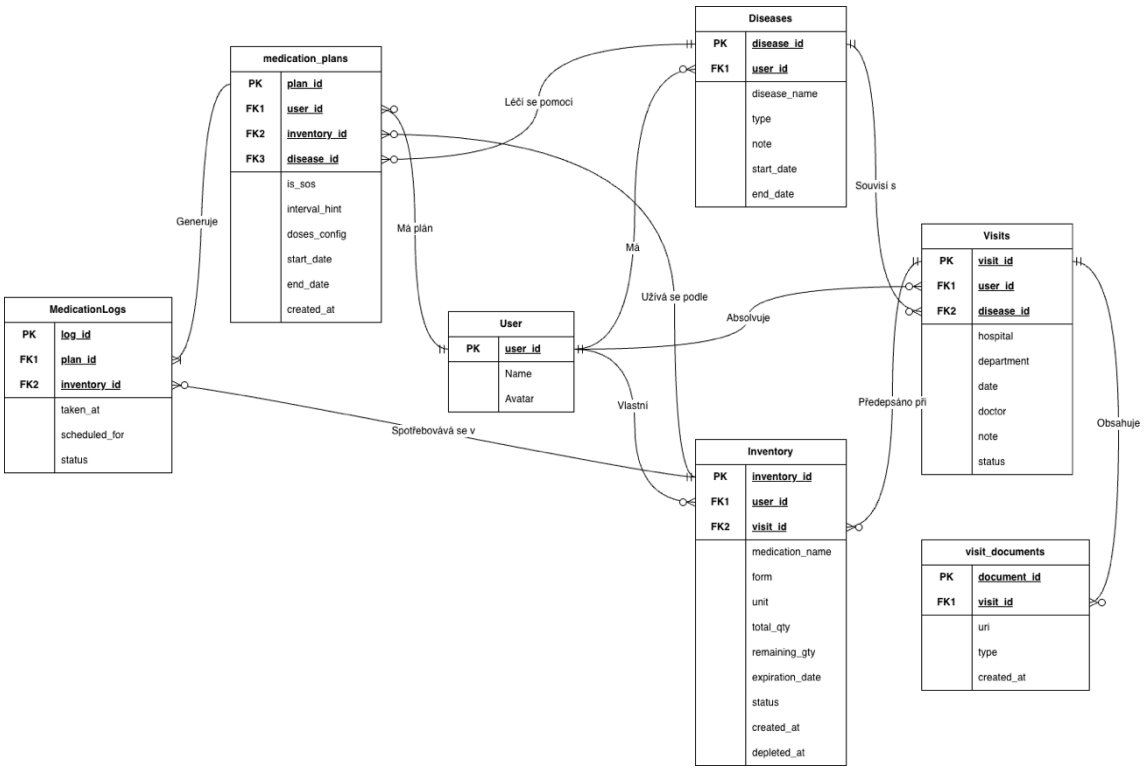
Obr. 15: Kaskádová kontrola při ukončení léčby
Zdroj: vlastní zpracování dle Zlatuška (2026)

6 Návrh a implementace databáze

Datová vrstva aplikace eZdraví je realizována prostřednictvím relační databáze SQLite provozované lokálně na zařízení uživatele. Návrh databázového schématu vychází z analýzy domény osobní zdravotní agendy a jejích klíčových entit: uživatel, diagnóza, návštěva lékaře, dokumenty z návštěvy, fyzická zásoba léků, léčebný režim a záznamy o užívání. Schéma je definováno v TypeScriptu prostřednictvím Drizzle ORM v souboru `db/schema.ts` a verzováno systémem migrací. Inicializace databázového spojení je realizována v souboru `db/index.ts`, kde je nejprve otevřen nebo vytvořen databázový soubor `ezdravi.db` voláním `openDatabaseSync`, a tento soubor je následně předán instanci Drizzle ORM spolu s definicí schématu. Výsledný objekt `db` je exportován jako singleton a importován ve všech obrazovkách aplikace, které potřebují přistupovat k datům.

6.1 Datový model

Databázové schéma aplikace eZdraví tvoří sedm tabulek, jejichž vzájemné vztahy vycházejí z reálné logiky správy zdravotní agendy pacienta. Každá tabulka reprezentuje jednu jasně ohraničenou doménu a s ostatními tabulkami je propojena prostřednictvím cizích klíčů, čímž je zajištěna referenční integrita dat.



Obr. 16: ER Diagram

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulka `users` slouží jako kořenová entita celého datového modelu. Uchovává jméno uživatele a časové razítko vytvoření profilu. Všechny ostatní entity jsou prostřednictvím sloupce `user_id` navázány na konkrétního uživatele, což umožňuje podporu více profilů na jednom zařízení a každý uživatel vidí výhradně svá vlastní data.

```
// 1. UŽIVATELÉ (Users)
export const users = sqliteTable('users', {
  user_id: integer('user_id').primaryKey({ autoIncrement: true }),
  name: text('name').notNull(),
  created_at: text('created_at').notNull(),
});
```

Obr. 17: Tabulka users

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulka `diseases` reprezentuje záznamy diagnóz. Název entity prošel během vývoje konceptuální revizí. Původní označení 'nemoc' bylo nahrazeno termínem 'diagnóza', který lépe vystihuje šíři evidovaných stavů — kromě klasických onemocnění zahrnuje například i úrazy nebo chronické stavy. Kromě názvu a poznámky uchovává datum počátku a datum ukončení nemoci. Sloupec `end_date` může nabývat hodnoty `NULL`, což vyjadřuje, že nemoc stále probíhá. Sloupec `type` rozlišuje dva typy onemocnění: akutní (`ACUTE`) pro krátkodobá onemocnění a chronická (`CHRONIC`) pro dlouhodobá. Toto rozlišení ovlivňuje způsob zobrazení záznamu v uživatelském rozhraní i logiku datumového omezení navázaných léčebných režimů.

```
// 2. Diagnózy (Diagnoses)
export const diseases = sqliteTable('diseases', {
  disease_id: integer('disease_id').primaryKey({ autoIncrement: true }),
  user_id: integer('user_id').references(() => users.user_id),
  disease_name: text('disease_name').notNull(),
  type: text('type').notNull(), // 'ACUTE' nebo 'CHRONIC'
  note: text('note'),
  start_date: text('start_date'),
  end_date: text('end_date'), // Pokud je null, nemoc stále probíhá
});
```

Obr. 18: Tabulka diseases (Diagnózy)

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulka `visits` eviduje návštěvy lékaře. Je volitelně propojena s tabulkou `diseases` cizím klíčem `disease_id`, což umožňuje vázat návštěvu na konkrétní onemocnění. Sloupec `status` nabývá hodnot `PLANNED` pro plánované a `COMPLETED` pro proběhlé návštěvy. Tento stav je v aplikaci automaticky odvozován porovnáním uloženého data a času návštěvy s aktuálním časem.

```
// 3. NÁVŠTĚVY LÉKAŘE (Visits)
export const visits = sqliteTable('visits', {
  visit_id: integer('visit_id').primaryKey({ autoIncrement: true }),
  user_id: integer('user_id').references(() => users.user_id),
  disease_id: integer('disease_id').references(() => diseases.disease_id),
  hospital: text('hospital'),
  department: text('department'),
  date: text('date'),
  doctor: text('doctor'),
  note: text('note'),
  medical_report: text('medical_report'),
  status: text('status'), // 'PLANNED', 'COMPLETED', 'MISSED'
});
```

Obr. 19: Tabulka visits

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulky `inventory` a `medication_plans` tvoří jádro systému správy léků a jsou vzájemně propojeny cizím klíčem. Jejich oddělení vychází z klíčového konceptuálního rozhodnutí popsaného v následující podkapitole.

```
// 4. LÉKÁRNIČKA – Fyzický sklad (Inventory)
export const inventory = sqliteTable('inventory', {
  inventory_id: integer('inventory_id').primaryKey({ autoIncrement: true }),
  user_id: integer('user_id').references(() => users.user_id),
  visit_id: integer('visit_id').references(() => visits.visit_id),

  medication_name: text('medication_name').notNull(), // Např. "Zyrtec"
  form: text('form').notNull(), // 'PILL' nebo 'SYRUP'
  unit: text('unit').notNull(), // 'ks' nebo 'ml'

  total_qty: real('total_qty').notNull(), // Kolik toho bylo na začátku (např. 30)
  remaining_qty: real('remaining_qty').notNull(), // Kolik reálně zbývá v krabičce (např. 25)

  expiration_date: text('expiration_date'), // Kdy lék projde
  status: text('status').default('ACTIVE'), // 'ACTIVE' (mám ho) nebo 'DEPLETED' (došel/vyhozen)

  created_at: text('created_at').notNull(),
  depleted_at: text('depleted_at'), // Kdy jsem krabičku dobral (pro historii)
});
```

Obr. 20: Tabulka inventory

Zdroj: vlastní zpracování dle Zlatuška (2026)

```
// 5. KARTOTÉKA – Režim užívání (Medication Plans)
// Zde se definuje, JAK se konkrétní krabička z Lékárničky používá pro danou nemoc.
export const medicationPlans = sqliteTable('medication_plans', {
  plan_id: integer('plan_id').primaryKey({ autoIncrement: true }),
  user_id: integer('user_id').references(() => users.user_id),

  // VAZBY NA OKOLÍ
  inventory_id: integer('inventory_id').references(() => inventory.inventory_id).notNull(), // Která fyzická krabička se bere
  disease_id: integer('disease_id').references(() => diseases.disease_id), // Na jakou nemoc to je (volitelné, může to být prevence)

  // TYP UŽÍVÁNÍ
  is_sos: integer('is_sos', { mode: 'boolean' }).default(false), // Je to jen podle potřeby?
  interval_hint: text('interval_hint'), // Např. "při bolesti, max po 6h" (pro SOS)

  // FORMÁT: [{"days":["1","2"],"doses":[{"time":"08:00","amount":1}]}]
  doses_config: text('doses_config').notNull(),

  // OD KDY DO KDY TENTO PLÁN PLATÍ
  start_date: text('start_date'),
  end_date: text('end_date'), // Pokud plán ukončím a vytvořím nový, tady bude datum konce

  created_at: text('created_at').notNull(),
});
```

Obr. 21: Tabulka medication plans

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulka `medication_logs` ukládá jednotlivé záznamy o skutečném užití léku. Je napojena na tabulku `medication_plans` cizím klíčem `plan_id` a na tabulku `inventory` cizím klíčem `inventory_id`. Uchovává naplánovaný datum a čas dávky (`scheduled_date`, `scheduled_time`), reálný čas užití (`taken_at`), užité množství (`amount`) a stav záznamu (`TAKEN` nebo `MISSED`). Oddělené uložení naplánovaného a skutečného času umožňuje aplikaci sledovat, zda byl lék vzat včas nebo zpětně.

```
// 6. HISTORIE UŽÍVÁNÍ – Logy (Medication Logs)
export const medicationLogs = sqliteTable('medication_logs', {
  log_id: integer('log_id').primaryKey({ autoIncrement: true }),
  plan_id: integer('plan_id').references(() => medicationPlans.plan_id).notNull(), // Kterého plánu se to týká
  inventory_id: integer('inventory_id').references(() => inventory.inventory_id).notNull(),

  scheduled_date: text('scheduled_date'), // ISO YYYY-MM-DD (Pro snazší filtrování "Dneška")
  scheduled_time: text('scheduled_time'), // HH:mm (Kdy se to mělo vzít, u SOS bude null)

  taken_at: text('taken_at'), // ISO Timestamp reálného odkliknutí (kdy to opravdu polkl)

  amount: real('amount').notNull(), // Kolik ks/ml se reálně vzalo (abychom to uměli "vrátit" do krabičky při chybě)
  status: text('status').notNull(), // 'TAKEN' (vzal) nebo 'MISSED' (úmyslně nevzal)
});
```

Obr. 22: Tabulka medication logs

Zdroj: vlastní zpracování dle Zlatuška (2026)

Tabulka `visit_documents` slouží k uchovávání referencí na obrazovou dokumentaci, jako jsou fotografie lékařských zpráv a detailů vyšetření. Namísto ukládání objemných binárních dat přímo do databáze obsahuje tabulka pouze textové URI cesty (`document_uri`) k souborům uloženým lokálně v zařízení. Pomocí cizího klíče `visit_id` je každý dokument pevně navázán na konkrétní návštěvu v tabulce `visits`, což umožňuje evidovat libovolné množství dokumentů k jedné prohlídce a zachovává čistotu hlavní tabulky návštěv.

```
// 7. LÉKAŘSKÉ ZPRÁVY A DOKUMENTY (Visit Documents)
export const visitDocuments = sqliteTable('visit_documents', {
  document_id: integer('document_id').primaryKey({ autoIncrement: true }),
  visit_id: integer('visit_id').references(() => visits.visit_id).notNull(), // K jaké návštěvě to patří

  uri: text('uri').notNull(), // Cesta k souboru (fotce) v telefonu
  type: text('type').default('IMAGE'), // 'IMAGE' nebo 'PDF'

  created_at: text('created_at').notNull(),
});
```

Obr. 23: Tabulka visitDocuments

Zdroj: vlastní zpracování dle Zlatuška (2026)

6.2 Konceptuální oddělení lékárničky a léčebného režimu

Nejdůležitějším architektonickým rozhodnutím datového modelu je rozdělení správy léků do dvou samostatných tabulek `inventory` a `medication_plans`. Toto rozdělení není technickou konvencí, ale odráží reálný rozdíl mezi dvěma odlišnými koncepty, které by při sloučení do jedné tabulky způsobovaly datové nekonzistence.

Tabulka `inventory` reprezentuje fyzický objekt jako krabičku léku v lékárničce. Eviduje, kolik tablet nebo mililitrů bylo v krabičce původně (`total_qty`) a kolik jich reálně zbývá (`remaining_qty`). Tabulka neví nic o tom, jak, kdy nebo na jakou nemoc se lék užívá — její jedinou odpovědností je sledovat fyzický stav zásoby.

Tabulka `medication_plans` reprezentuje léčebný záměr, konfiguraci toho, jak se konkrétní krabička z lékárničky v daném období užívá. Uchovává vazbu na krabičku (`inventory_id`), vazbu na onemocnění (`disease_id`), typ dávkování (pravidelné nebo SOS), dávkovací harmonogram (`doses_config`) a platnost režimu (`start_date`, `end_date`). Sloupec `doses_config` ukládá harmonogram ve formátu JSON, kde každý prvek pole definuje dny v týdnu a příslušné dávky s časy. Například hodnota [{"days": ["1","2","3","4","5","6","7"], "doses":[{"time":"08:00","amount":1},{"time":"20:00","amount":1}]}] vyjadřuje pravidelné dávkování jedné tablety každý den ráno v 8:00 a večer ve 20:00.

Toto oddělení přináší tři konkrétní výhody. Zaprvé, jedna fyzická krabička léku může být v průběhu svého životního cyklu spojena s více léčebnými režimy, například při dvou různých onemocněních nebo při změně dávkování. Zadruhé, ukončení léčebného režimu nemá žádný vliv na fyzický stav zásoby v lékárničce. Krabička zůstává evidována jako aktivní, dokud není skutečně vyčerpána. Za třetí, historie veškerého užívání uložená v tabulce `medication_logs` zůstává konzistentní i po skončení léčebného režimu, protože záznamy referují na konkrétní plán a konkrétní krabičku, které v databázi vždy zůstávají.

6.3 Vývoj schématu prostřednictvím migrací

Databázové schéma aplikace eZdraví nebylo navrženo ve své finální podobě od počátku, ale vyvíjelo se iterativně v průběhu implementace. Každá změna schématu je zachycena jako samostatný migrační soubor ve složce `drizzle/`, přičemž Drizzle ORM při každém spuštění aplikace ověřuje, které migrace již byly na zařízení aplikovány, a případné nové migrace automaticky provede. Tímto mechanismem je zajištěno, že uživatelé, kteří nainstalují aktualizaci aplikace se změněným schématem, o svá data nepřijdou.

Série osmi migračních souborů dokumentuje klíčové architektonické změny, ke kterým v průběhu vývoje došlo. První migrace (`0000`) vytvořila počáteční schéma s tabulkami `users`, `diseases`, `visits` a jedinou tabulkou `medications`, která v sobě kombinovala jak fyzické informace o léku, tak dávkovací konfiguraci. Dávkovací harmonogram byl v této verzi reprezentován třemi booleovskými sloupci `morning`, `noon` a `evening`, což neumožňovalo přesné zadání časů dávek. (viz Příloha 3)

Druhá migrace (`0001`) nahradila booleovské sloupce textovým sloupcem `doses_times` jako první krok ke strukturovanějšímu harmonogramu. Třetí migrace (`0002`) (viz Příloha 4) přepracovala tabulku `medications` do podoby s flexibilním sloupcem `doses_config` ve formátu JSON a přidala sloupec `days_of_week` pro určení dnů v týdnu. Čtvrtá migrace (`0003`) doplnila chybějící vazby na nemoci a návštěvy a přidala příznak SOS dávkování.

```
ALTER TABLE `medications` ADD `doses_times` text;--> statement-breakpoint
ALTER TABLE `medications` DROP COLUMN `morning`;--> statement-breakpoint
ALTER TABLE `medications` DROP COLUMN `noon`;--> statement-breakpoint
ALTER TABLE `medications` DROP COLUMN `evening`;
```

Obr. 24: Migrace (0001)

Zdroj: vlastní zpracování dle Zlatuška (2026)

```
ALTER TABLE `medications` ADD `disease_id` integer REFERENCES diseases(disease_id);--> statement-breakpoint
ALTER TABLE `medications` ADD `visit_id` integer REFERENCES visits(visit_id);--> statement-breakpoint
ALTER TABLE `medications` ADD `is_sos` integer DEFAULT false;--> statement-breakpoint
ALTER TABLE `medications` ADD `interval_hint` text;--> statement-breakpoint
ALTER TABLE `medications` ADD `start_date` text;--> statement-breakpoint
ALTER TABLE `medications` ADD `end_date` text;--> statement-breakpoint
ALTER TABLE `medications` DROP COLUMN `days_of_week`;
```

Obr. 25: Migrace (0003)

Zdroj: vlastní zpracování dle Zlatuška (2026)

Nejvýznamnější architektonická změna přišla s pátou migrací (`0004`), která implementovala klíčové konceptuální oddělení popsané v předchozí podkapitole. Původní tabulka `medications` byla zrušena a nahrazena dvěma novými tabulkami: `inventory` pro fyzickou evidenci krabičky a `medication_plans` pro konfiguraci léčebného režimu. Tabulka `medication_logs` byla

zároveň přepracována tak, aby referovala na `medication_plans` namísto původní tabulky `medications`. (viz Příloha 5)

Šestá migrace (`0005`) doplnila do tabulky `inventory` vazbu na návštěvu lékaře, čímž bylo umožněno evidovat, při které návštěvě byl lék předepsán. Sedmá migrace (`0006`) přidala přímou vazbu `inventory_id` do tabulky `medication_logs`, která umožňuje aplikaci přímo snižovat zásobu při záznamu užití bez nutnosti procházet přes tabulku `medication_plans`.

Série migrací byla v pozdější fázi vývoje doplněna o osmou migraci (`0007`), která reagovala na požadavek z praxe umožnit přikládání fyzických lékařských zpráv k jednotlivým návštěvám lékaře. Původní úvaha ukládat dokumenty přímo do tabulky visits by omezila systém na jeden dokument na návštěvu. Proto byla vytvořena nová, oddělená tabulka `visit_documents` s relací 1:N vůči tabulce návštěv. Toto normalizované řešení zajistilo vysokou flexibilitu systému bez narušení stávající datové struktury a referenční integrity.

```
CREATE TABLE `visit_documents` (
    `document_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `visit_id` integer NOT NULL,
    `uri` text NOT NULL,
    `type` text DEFAULT 'IMAGE',
    `created_at` text NOT NULL,
    FOREIGN KEY (`visit_id`) REFERENCES `visits`(`visit_id`) ON UPDATE no action ON DELETE no action
);
```

Obr. 26: Migrace (0005)

Zdroj: vlastní zpracování dle Zlatuška (2026)

7 Implementace aplikace

Implementace aplikace eZdraví byla realizována v prostředí React Native s využitím platformy Expo a TypeScriptu jako primárního programovacího jazyka. Tato kapitola popisuje strukturu projektu, způsob organizace kódu a implementaci jednotlivých funkčních celků aplikace. Důraz je kladen na popis konkrétních technických rozhodnutí a jejich odůvodnění.

7.1 Struktura projektu

Projekt je organizován do několika hlavních složek, jejichž rozdělení odráží separaci odpovědností jednotlivých vrstev aplikace.

Složka `app/` je kořenovým adresářem pro veškerou navigační logiku a implementaci obrazovek. Expo Router interpretuje souborovou strukturu uvnitř této složky jako navigační schéma. Každý soubor s příponou `.tsx` automaticky registruje odpovídající obrazovku. Vnořená složka `(tabs)/` obsahuje pět obrazovek hlavní navigace (domovská obrazovka, kalendář, lékárnička, kartotéka a uživatel) spolu s jejich společným konfiguračním souborem `_layout.tsx`. Obrazovky dostupné přes zásobníkový navigátor – formuláře pro přidání záznamů a detailní pohledy – jsou umístěny přímo v kořeni složky `app/`. Kořenový soubor `app/index.tsx` zastává zvláštní roli: je to první soubor spuštěný po startu aplikace a obstarává logiku onboardingu a autentizace, přičemž uživatele přesměrovává buď do stručného tutoriálu, nebo přímo do hlavní části aplikace.

Složka `components/` sdružuje sdílené komponenty využívané napříč více obrazovkami. Klíčovými soubory jsou `CalendarPicker.tsx` — vlastní komponenta pro výběr data, jejíž vznik byl motivován technickými omezeními nativní iOS komponenty popsanými v kapitole 8 — `MedicationCard.tsx`, která zapouzdřuje jednotný vizuální styl karty léku zobrazované na domovské obrazovce.

Složka `db/` obsahuje dva soubory tvořící datovou vrstvu aplikace. Soubor `schema.ts` definuje strukturu databáze prostřednictvím Drizzle ORM, soubor `index.ts` inicializuje databázové spojení. Složka `drizzle/` umístěná na stejné úrovni jako `app/` obsahuje migrační soubory popsané v kapitole 6.

Složka `hooks/` je připravena pro vlastní React hooks sdílené napříč obrazovkami. V aktuální verzi aplikace je správa stavu realizována přímo ve vrstvě komponent prostřednictvím vestavěných hooks frameworku React.

7.2 Onboarding a autentizace

Logika prvního spuštění aplikace a zabezpečení přístupu k ní jsou implementovány jako dva oddělené, spolupracující moduly. Toto rozdělení odpovědností (angl. separation of concerns) zvyšuje přehlednost kódu a zároveň řeší potenciální problémy s dvojí inicializací komponent v prostředí React Strict Mode (React, 2024a).

Onboarding v souboru index.tsx

Soubor `index.tsx` slouží výhradně jako vstupní bod pro nové uživatele. Po spuštění provede dotaz do lokální databáze a ověří, zda již existuje záznam o uživateli. Pokud ano, přesměruje

aplikaci okamžitě do hlavní navigace, aniž by zobrazila jakýkoliv onboardingový obsah. Pokud uživatel v databázi nalezen není, spustí se sekvence tří úvodních obrazovek (tutorial), jejichž stav je řízen výčtovým typem AppState s hodnotami LOADING, TUTORIAL_1, TUTORIAL_2, TUTORIAL_3 a REGISTER. Po dokončení tutorialu je uživatel vyzván k zadání jména, které je následně uloženo jako první záznam v tabulce users. Zabezpečení aplikace tento soubor již nezajišťuje. Tato odpovědnost byla přenesena do nadřazené vrstvy.

Autentizace v souboru _layout.tsx

Zabezpečení přístupu je implementováno v kořenovém souboru `_layout.tsx`, který tvoří obal pro celou navigační strukturu aplikace. Tento soubor funguje jako stavový automat se třemi implicitními stavy: LOADING (probíhá inicializace databáze a ověření nastavení), LOCKED (zámek je aktivní, aplikace čeká na biometrické ověření) a UNLOCKED (přístup byl udělen, aplikace se normálně vykresluje). Klíčovým bezpečnostním principem je skutečnost, že komponenty hlavní navigace (`Stack`) jsou do DOM vykreslovány až po úspěšném přechodu do stavu UNLOCKED. Útočník tedy nemůže obejít zámek pouhou manipulací s navigací. (viz Příloha 6)

Pro zamezení opakovaného vyvolání biometrického dialogu, což je problém typický pro vývojové prostředí Expo s aktivním React Strict Mode, který komponenty záměrně inicializuje dvakrát, jsou zavedeny dvě proměnné v globálním scope modulu: `globalAuthLock` a `globalAlreadyUnlocked`. Tyto proměnné přetrvávají po celou dobu životního cyklu JavaScriptového vlákna a slouží jako sdílená paměť mezi případnými vícenásobnými instancemi komponenty. Díky tomu je biometrické ověření vyvoláno právě jednou, bez ohledu na interní chování React rendereru.

```
// Globální paměť řešící problém s vícenásobným vykreslováním v React Strict Mode
let globalAuthLock = false;       // Brání dvojitému vyvolání FaceID
let globalAlreadyUnlocked = false; // Říká všem instancím, že už je odemčeno
```

Obr. 27: Bezpečnostní stavový automat (Ochrana přes ověření)

Zdroj: vlastní zpracování dle Zlatuška (2026)

```
const triggerAuth = async () => {
  // Pokud už se autentizuje, nebo už je odemčeno, tlačítko nereaguje
  if (globalAuthLock || globalAlreadyUnlocked) return;
  globalAuthLock = true;

  try {
      const auth = await LocalAuthentication.authenticateAsync({
      promptMessage: 'Odemkněte eZdraví',
      fallbackLabel: 'Použít kód',
      cancelLabel: 'Zrušit',
      disableDeviceFallback: false,
    });

      if (auth.success) {
        globalAlreadyUnlocked = true; // Zapsáno do globální paměti!
        setIsUnlocked(true);
      }
  } catch(e) {
      console.log(e);
  } finally {
      // Zámek tlačítka uvolníme až po 1 sekundě pro jistotu
      setTimeout(() => {
          globalAuthLock = false;
      }, 1000);
  }
};
```

Obr. 28: Implementace biometrické autentizace s ochranou proti dvojitému spuštění (triggerAuth)

Zdroj: vlastní zpracování dle Zlatuška (2026)

Biometrická autentizace je realizována prostřednictvím knihovny `expo-local-authentication`, která na zařízeních Apple využívá rozhraní Face ID nebo Touch ID a na platformě Android odpovídající biometrické API (Expo Documentation, 2024g). Zámek aplikace je volitelný. Jeho aktivace je uložena v `expo-secure-store` (Expo Documentation, 2024d) pod klíčem `app_lock_enabled`. Pokud uživatel zámek neaktivoval, příznak `globalAlreadyUnlocked` je nastaven na hodnotu `true` ihned při inicializaci, čímž se zamykací obrazovka přeskočí bez jakéhokoliv zpoždění.

Konfigurace oprávnění v app.json

Platforma iOS vyžaduje, aby každá aplikace, která přistupuje k biometrickým funkcím zařízení, deklarovala účel tohoto přístupu v podobě textového řetězce zobrazeného uživateli při prvním vyžádání oprávnění (Apple Inc, 2024b). Tato deklarace je součástí konfiguračního souboru `app.json` v sekci pluginu `expo-local-authentication` pod klíčem `faceIDPermission`. V aplikaci eZdraví byl tento řetězec definován jako: „Aplikace eZdraví potřebuje FaceID pro bezpečné uzamčení tvých osobních zdravotních dat." Absence tohoto záznamu by způsobila zamítnutí aplikace v procesu revize App Store, případně pád aplikace při pokusu o vyvolání biometrického dialogu na fyzickém zařízení.

7.3 Hlavní navigace

Hlavní navigační struktura aplikace je definována v souboru `app/(tabs)/_layout.tsx` prostřednictvím komponenty `Tabs` z Expo Routeru. Záložky jsou pojmenovány podle souborů ve složce `(tabs)/` a odpovídají pěti hlavním sekcím aplikace: domovská obrazovka (`index`), kalendář (`calendar`), lékárnička (`inventory`), kartotéka (`records`) a účet (`user`). Vizuálně je navigační lišta realizována jako plovoucí panel s bílým pozadím a zaoblenými rohy, umístěný 24 pixelů nad spodním okrajem obrazovky s horizontálními odsazeními 20 pixelů. Stín definovaný parametry `shadowOpacity` a `shadowRadius` vizuálně odděluje lištu od obsahu pod ní. Parametr `safeAreaInsets: { bottom: 0 }` zabraňuje operačním systémům iOS a Android automaticky zvyšovat lištu kvůli domovskému proužku zařízení. Bez tohoto nastavení by plovoucí lišta byla posunuta příliš vysoko a obsah pod ní by nebyl správně odsazen. Texty záložek jsou skryty nastavením `tabBarShowLabel: false`, čímž je dosaženo čistějšího vizuálního dojmu.

Ikonografie navigační lišty je plně sjednocena a využívá výhradně standardizovanou vektorovou sadu `MaterialCommunityIcons`, která je distribuována přímo jako součást ekosystému Expo (Expo Documentation, 2024i). Tento přístup eliminuje závislost na externích SVG souborech a zajišťuje maximální vizuální konzistenci napříč celou aplikací. Odezva na interakci uživatele je řešena dynamickou změnou stavu ikon: aktivní záložka je reprezentována vyplněnou verzí ikony (např. home) v primární zelené barvě, zatímco neaktivní záložky využívají obrysovou variantu (např. home-outline) v sekundární šedé barvě.

Technické řešení pozicování ikon bylo optimalizováno a upouští od složitého stylování pomocí pevných vnitřních okrajů (paddingů). Původní řešení na platformě iOS kolidovalo se systémovou bezpečnou zónou (Safe Area) a způsobovalo nežádoucí ořezávání spodních hran některých ikon. Aktuální implementace spoléhá výhradně na flexibilní uspořádání (Flexbox) a nastavení

vlastnosti `tabBarIconStyle`, které zajišťuje automatické a bezpečné vycentrování prvků přesně do středu plovoucí lišty bez ohledu na specifika konkrétního zařízení a jeho operačního systému.

7.4 Domovská obrazovka a denní přehled

Domovská obrazovka implementovaná v souboru `app/(tabs)/index.tsx` plní funkci operačního dashboardu aplikace. Při každém zobrazení obrazovky jsou z databáze načtena data relevantní pro aktuální den: plánované dávky léků, SOS léky s aktivním léčebným režimem a blížící se návštěvy lékaře. Načítání dat je spouštěno hookem `useFocusEffect` v kombinaci s `useCallback`, nikoli standardním `useEffect`. Toto rozhodnutí řeší konkrétní chování Expo Routeru: záložkové obrazovky jsou po prvním načtení udržovány v paměti a při přepnutí na ně se znovu nerenderují, takže hook `useEffect` s prázdným polem závislostí by se znovu nespustil. `useFocusEffect` naopak reaguje na událost získání fokusu obrazovky, tedy na každé přepnutí záložky nebo návrat ze zásobníkové obrazovky a zajišťuje tak, že data zobrazená na domovské obrazovce vždy odpovídají aktuálnímu stavu databáze. Bez tohoto řešení by uživatel, který by přidal nový záznam a vrátil se na domovskou obrazovku, neviděl aktualizovaná data.

```
useFocusEffect(
  useCallback(() => {
    loadData();
  }, [])
);
```

Obr. 29: Zajištění aktuality dat při návratu na obrazovku (useFocusEffect)
Zdroj: vlastní zpracování dle Zlatuška (2026)

Denní přehled léků je konstruován dotazem na tabulku `medication_plans`, který filtruje záznamy aktuálního uživatele s platným datovým rozsahem. Pro každý nalezený plán je z pole `doses_config` (uloženého ve formátu JSON) parsován dávkovací harmonogram a porovnáván s aktuálním dnem v týdnu. Pokud harmonogram pro daný den obsahuje naplánované dávky, komponenta pro každou dávku zobrazí interaktivní tlačítko s naplánovaným časem a množstvím. Stisknutím tlačítka je vytvořen záznam v tabulce `medication_logs` a zároveň je snížena hodnota `remaining_qty` v tabulce `inventory`. Pokud po odečtení klesne zásoba na nulu nebo méně, je stav záznamu v `inventory` automaticky přepnut na `DEPLETED` a vyplněno pole `depleted_at`. Tato logika zajišťuje, že stav lékárničky vždy přesně odráží fyzický stav zásoby bez nutnosti manuální aktualizace ze strany uživatele.

SOS léky jsou prezentovány v oddělené sekci s výrazným tlačítkem „Vzít teď". Na rozdíl od pravidelných dávek nevyžadují SOS léky výběr konkrétního času — záznam je vytvořen s okamžitým časovým razítkem. Sekce blížících se návštěv zobrazuje záznamy z tabulky `visits` se stavem `PLANNED` a datem v budoucnosti, seřazené chronologicky, přičemž jsou zobrazeny maximálně dva nejbližší záznamy.

7.5 Správa zdravotních záznamů

Správa zdravotních záznamů je soustředěna do obrazovky kartotéky implementované v souboru `app/(tabs)/records.tsx` a sady detailních a formulářových obrazovek dostupných přes zásobníkový navigátor.

Obrazovka kartotéky organizuje záznamy do tří kategorií přepínatelných záložkami: diagnózy, medikace a návštěvy. Každá kategorie dále nabízí filtrování na aktuální a historické záznamy. Diagnózy s hodnotou `end_date` rovnou `NULL` jsou klasifikovány jako aktuální, záznamy s vyplněným datem ukončení jako historické. Stejná logika platí pro medikace: záznamy s `end_date` rovným `NULL` jsou zobrazeny jako probíhající léčebné režimy. Návštěvy jsou kategorizovány na základě hodnoty sloupce `status` — záznamy se stavem `PLANNED` jsou aktuální, záznamy se stavem `COMPLETED` historické. Obrazovka dále disponuje textovým vyhledávacím polem, které filtruje záznamy v reálném čase porovnáváním zadaného řetězce s názvem záznamu bez ohledu na velikost písmen. Vyhledávání je implementováno čistě na straně klienta, nad již načtenými daty bez dalších databázových dotazů. V pravém horním rohu je umístěno tlačítko pro export dokumentace do formátu PDF realizovaný prostřednictvím modulu `expo-print`.

```
// Generování dynamického obsahu a konverze do PDF
const { uri } = await Print.printToFileAsync({
  html: htmlContent,
  base64: false
});
await Sharing.shareAsync(uri, {
  mimeType: 'application/pdf',
  dialogTitle: 'Sdílet lékařský report',
  UTI: 'com.adobe.pdf'
});
```

Obr. 30: Generování PDF reportů

Zdroj: vlastní zpracování dle Zlatuška (2026)

Export zdravotní dokumentace do formátu PDF

Klíčovou funkcionalitou aplikace eZdraví z hlediska komunikace pacienta s lékařem je možnost exportu zdravotní dokumentace do formátu PDF. Funkce je dostupná z detailní obrazovky diagnózy a generuje strukturovaný report konkrétní zdravotní epizody zahrnující veškeré relevantní záznamy navázané na dané onemocnění.

Proces generování PDF probíhá ve čtyřech fázích. V první fázi jsou z databáze načtena všechna data potřebná pro sestavení reportu: základní informace o onemocnění, jméno pacienta z tabulky `users`, seznam navázaných léčebných režimů s vazbou na záznamy v tabulce `inventory` a seznam návštěv lékaře spojených s danou diagnózou. Tato data jsou již k dispozici v paměti komponenty jako součást jejího stavu načteného při otevření detailní obrazovky, takže fáze sběru dat nevyžaduje žádné dodatečné databázové dotazy.

Ve druhé fázi je dynamicky sestavena HTML šablona s vloženými kaskádovými styly. Šablona obsahuje záhlaví s názvem onemocnění a jménem pacienta, informační blok s datovým rozsahem a typem onemocnění, volitelnou sekci s poznámkami lékaře a dvě tabulky — tabulku léčebných režimů a tabulku návštěv lékaře. Hodnoty jsou do šablony vkládány prostřednictvím JavaScriptových template literals přímým interpolováním dat. Zvláštní pozornost si zaslouží zpracování sloupce `doses_config`, jehož hodnota je uložena jako řetězec ve formátu JSON.

Pro potřeby reportu je tento řetězec parsován pomocnou funkcí `formatDoses`, která z pole dávkovacích konfigurací sestaví lidsky čitelný text. Například Každý den: 08:00 (1 ks), 20:00 (1 ks). Tím je zajištěno, že technický formát úložiště dat je před předáním lékaři převeden do srozumitelné podoby. Šablona dále obsahuje zápatí s datem generování reportu.

Ve třetí fázi je sestavená HTML šablona předána funkci `Print.printToFileAsync` z modulu `expo-print`. Tato funkce spustí systémový WebKit renderovací engine, který HTML dokument zpracuje a uloží výsledný PDF soubor do dočasného adresáře aplikace na zařízení. Návratová hodnota funkce obsahuje URI cestu k vytvořenému souboru.

Ve čtvrté fázi je URI souboru předáno funkci `Sharing.shareAsync` z modulu `expo-sharing` spolu s metadaty MIME typu `application/pdf`. Tato funkce vyvolá systémový dialog sdílení operačního systému, který uživateli nabídne dostupné způsoby předání dokumentu jako je odeslání e-mailem, uložení do cloudového úložiště, předání do jiné aplikace nebo přímý tisk prostřednictvím systémového tiskového dialogu. Celý mechanismus exportu tak nevyžaduje žádnou serverovou infrastrukturu a funguje zcela offline, což je v souladu s offline-first architekturou aplikace.

Detailní obrazovky jsou implementovány pro každou ze tří kategorií záznamů. Detail diagnózy (`disease-detail.tsx`) zobrazuje název, typ, datový rozsah a poznámky onemocnění, a dále přehled navázaných léčebných režimů a návštěv lékaře načtených samostatnými databázovými dotazy. Detail návštěvy (`visit-detail.tsx`) prezentuje datum, čas, jméno lékaře a poznámky, přičemž zobrazuje také navázanou diagnózu a léky předepsané při dané návštěvě. Detail medikace (`medication-detail.tsx`) zobrazuje konfiguraci léčebného režimu, celkové množství léku odebrané v rámci daného plánu a chronologicky seřazenou historii jednotlivých záznamů užití. U každého záznamu v historii je umožněno zpětné přidání nebo oprava prostřednictvím modálního formuláře s výběrem data a času.

Vzájemné provázání entit napříč detailními obrazovkami přineslo specifickou navigační výzvu popsanou v kapitole 8. Formulářové obrazovky pro přidání a úpravu záznamů (`add-disease.tsx`, `add-visit.tsx`, `add-medication.tsx`, `add-inventory.tsx`) jsou dostupné jak z kartotéky, tak z kontextu konkrétní entity jako je například přidání medikace přímo z detailu diagnózy. V takovém případě je formuláři předán parametr `preselected_disease_id`, který předvyplní vazbu na diagnózu a zároveň zablokuje možnost tuto vazbu změnit prostřednictvím vizuálního indikátoru zámku. Tento mechanismus eliminuje uživatelské chyby způsobené nechtěnou změnou kontextu při přidávání záznamu z provázané obrazovky.

7.6 Lékárnička

Obrazovka lékárničky implementovaná v souboru `app/(tabs)/inventory.tsx` zobrazuje přehled fyzických zásob léků uložených v databázi. Záznamy jsou rozděleny do dvou záložek: aktuální krabičky se stavem ACTIVE a archivované krabičky se stavem DEPLETED. Tato filtrace je prováděna na straně klienta nad kompletně načteným polem položek, čímž se eliminuje nutnost opakovaných databázových dotazů při přepínání záložek.

Renderování seznamu je realizováno komponentou `FlatList`, která na rozdíl od `ScrollView` s vnořenými pohledy renderuje pouze viditelné položky seznamu, čímž optimalizuje výkon při

větším počtu záznamů. Každá položka seznamu je obalena komponentou `Swipeable` z knihovny `react-native-gesture-handler`, která umožňuje gesto přejetí prstem doleva pro odhalení akce smazání. Použití `GestureHandlerRootView` jako kořenového obalu celé obrazovky je nutnou podmínkou pro správnou funkci gestových komponent na platformě Android, kde bez tohoto obalu gesture handling nefunguje.

Logika smazání krabičky rozlišuje dvě situace. Pokud na krabičku nenavazuje žádný léčebný plán ani žádný záznam v historii užívání, může být trvale smazána z databáze. Pokud takové vazby existují, smazání je zamítnuto s uživatelsky srozumitelným vysvětlením — tento ochranný mechanismus chrání integritu historických záznamů. Alternativou k trvalému smazání je archivace: prázdná krabička může být přesunuta do záložky Historie voláním rychlé akce „Vyřadit", která aktualizuje sloupec `status` na `DEPLETED` a vyplní časové razítko `depleted_at`.

Detail krabičky implementovaný v souboru `inventory-detail.tsx` zobrazuje podrobné informace o zásobě léku včetně vizuálního progressbaru znázorňujícího poměr spotřebovaného a zbývajícího množství. Z detailu krabičky je přístupná editace záznamu a přehled navázaných léčebných režimů, přičemž platí stejný navigační mechanismus předávání kontextových parametrů jako u ostatních detailních obrazovek.

7.7 Kalendářní přehled

Obrazovka časové osy implementovaná v souboru `app/(tabs)/calendar.tsx` poskytuje průřezový pohled na zdravotní historii uspořádanou podle data. Zobrazení je rozděleno na měsíční kalendář v horní části obrazovky a seznam záznamů pro vybraný den v části dolní.

Měsíční kalendář je realizován komponentou `Calendar` z knihovny `react-native-calendars`. Dny obsahující zdravotní záznamy jsou označeny barevnými tečkami pod číslem dne — zelená pro záznamy o užití léků, červená pro počátky nebo konce onemocnění a modrá pro návštěvy lékaře. Tečky jsou generovány ze slovníku `markedDates`, jehož klíče jsou datové řetězce ve formátu `YYYY-MM-DD` a hodnoty obsahují pole teček s definovanou barvou. Tento slovník je sestavován při každém načtení obrazovky výpočtem průniku všech záznamů z databáze se sadou dat v aktuálně zobrazeném měsíci. Pro optimalizaci jsou data načítána vždy při změně zobrazeného měsíce nebo při získání fokusu obrazovky.

Výběr dne v kalendáři aktualizuje stav `selectedDate`, jehož změna spustí filtrování načtených záznamů a zobrazení relevantních položek pod kalendářem. Záznamy jsou rozděleny do tematických skupin s nadpisy (Léky, Počátek nemoci, Návštěvy) a každá položka je navigovatelná do příslušného detailního pohledu. Filtrační lišta nad seznamem záznamů umožňuje omezit zobrazení na jednu kategorii, přičemž stav filtru je udržován lokálně v komponentě a nemá vliv na databázové dotazy.

7.8 Sdílená komponenta CalendarPicker

Komponenta `CalendarPicker` implementovaná v souboru `components/CalendarPicker.tsx` je sdílenou komponentou pro výběr data, která je využívána na všech obrazovkách aplikace vyžadujících zadání datumu — ve formulářích pro přidání diagnózy, návštěvy, záznamu o užití léku i pro zadání data expirace v lékárničce. Vznik

samostatné sdílené komponenty namísto přímého použití systémové komponenty byl motivován technickými omezeními popsanými podrobněji v kapitole 8.

Z funkčního hlediska komponenta zobrazuje modální panel vysunující se ze spodního okraje obrazovky, jehož jádrem je komponenta `Calendar` z knihovny `react-native-calendars`. Vlastní navigační hlavička nahrazuje výchozí záhlaví kalendáře a umožňuje přechod na předchozí nebo následující měsíc tlačítky se šipkami, přičemž šipky jsou vizuálně zneaktivněny na hranicích povoleného rozsahu dat definovaného parametry `minDate` a `maxDate`. Klepnutím na název aktuálního měsíce a roku je vyvolán sekundární modal s výběrem měsíce a roku, který prezentuje mřížku dvanácti měsíců a horizontálně scrollovatelný seznam roků v povoleném rozsahu. Měsíce mimo povolený rozsah jsou zašedlé a neklikatelné. Při otevření sekundárního modalu je seznam roků automaticky přeskrolován na aktuálně zobrazovaný rok voláním `scrollTo` s krátkým zpožděním, které zajišťuje dokončení renderování před provedením skrolování.

Komponenta přijímá parametry `themeColor` pro přizpůsobení primární barvy konkrétnímu kontextu použití — zelená pro standardní výběr data, červená pro výběr data ukončení onemocnění, modrá pro výběr data návštěvy. Volitelný parametr `deleteLabel` aktivuje tlačítko pro smazání aktuálně vybraného data, které je využíváno v kontextech, kde je datum volitelné, jako je datum expirace léku nebo datum ukončení onemocnění.

7.9 Zálohování a obnova dat

Vzhledem k tomu, že aplikace eZdraví pracuje s vysoce citlivými osobními a zdravotními údaji, bylo již v úvodní fázi návrhu rozhodnuto o využití výhradně lokálního databázového úložiště. Tento přístup sice maximalizuje ochranu soukromí, avšak přináší riziko ztráty dat v případě poškození nebo ztráty samotného mobilního zařízení. Z tohoto důvodu představuje implementace robustního mechanismu pro zálohování a obnovu dat kritickou funkcionalitu celého systému.

Zvolené řešení umožňuje exportovat kompletní stav databáze do přenositelného formátu JSON a opětovně jej importovat, čímž je zajištěna plná kontrola uživatele nad jeho daty i možnost snadného přenosu mezi zařízeními v offline režimu.

Architektura exportu dat

Proces exportu je implementován jako asynchronní operace, jejímž úkolem je agregovat veškerý obsah lokální SQLite databáze. Aplikace postupně provádí dotazy `SELECT` nad všemi sedmi tabulkami schématu (`users`, `diseases`, `visits`, `inventory`, `medicationPlans`, `medicationLogs`, `visitDocuments`). Využití ORM vrstvy Drizzle (Product by Drizzle Team, b. r. a) zajišťuje, že jsou tato data rovnou mapována do nativních JavaScriptových objektů, což výrazně zjednodušuje jejich následnou serializaci. Zásadním prvkem stability tohoto procesu je využití záchytných bloků `.catch(() => [])` u každého databázového dotazu. Toto řešení garantuje, že v případě prázdné tabulky nebo dílčí chyby nedojde k pádu celé exportní funkce, ale do výsledného objektu se bezpečně zapíše prázdné pole. (viz Příloha 7)

Získaný textový řetězec obsahující JSON je následně nutné fyzicky zapsat do souborového systému zařízení. K tomuto účelu byla využita knihovna `expo-file-system/legacy` (Expo Documentation, 2024j), konkrétně metoda `writeAsStringAsync`, která vytvoří dočasný soubor

`ezdravi_zaloha.json` v mezipaměti aplikace (`documentDirectory`). Použití modulu s příponou `/legacy` bylo nezbytné kvůli kompatibilitě v rámci vývojového prostředí Expo SDK 54 (Expo Documentation, 2024j).

Aby mohl uživatel se souborem reálně manipulovat, je vygenerovaná cesta předána systémovému modulu `expo-sharing` (Expo Documentation, 2024h). Voláním metody `Sharing.shareAsync` s definovaným MIME typem (`application/json`) se vyvolá nativní dialog operačního systému, prostřednictvím kterého může uživatel zálohu bezpečně uložit na cloudové úložiště, odeslat e-mailem nebo přenést na jiné zařízení.

Bezpečná obnova a referenční integrita

Zatímco export je z pohledu databáze pasivní operací pouze pro čtení, import dat představuje technicky nejnáročnější proces s vysokým rizikem porušení datové integrity. Proces začíná interakcí uživatele, který prostřednictvím rozhraní `expo-document-picker` (Expo Documentation, 2024ch) vybere záložní JSON soubor z úložiště telefonu. Aplikace načte obsah souboru pomocí metody `readAsStringAsync` a provede jeho deserializaci. Ještě před samotným zápisem probíhá základní validace struktury — pokud načtený objekt neobsahuje klíčová data (např. pole `users` a `diseases`), proces je okamžitě přerušen.

Samotná obnova dat probíhá po potvrzení uživatelem v rámci izolovaného asynchronního bloku. Prvním, avšak kritickým krokem, je kompletní vyčištění stávající databáze. Zde vyvstává problém s referenční integritou relační databáze SQLite (SQLite, 2024a) — pokud by byla odstraněna nadřazená tabulka dříve než tabulky na ni napojené, databáze by operaci zamítla kvůli konfliktu s cizími klíči. Z tohoto důvodu byl implementován striktní kaskádový model mazání, který postupuje od nejvíce závislých záznamů (historie užívání a plány) směrem k nadřazeným entitám a končí u samotných uživatelů. I zde je uplatněn bezpečnostní mechanismus `.catch(() => {})`, aby proces mazání nezhavaroval na případných neexistujících záznamech. (viz Příloha 8)

Po úspěšném vyčištění databáze nastupuje fáze importu. Aplikace iteruje přes jednotlivá pole parsovaného JSON objektu a pomocí metody `db.insert().values()` vkládá celá pole záznamů zpět do příslušných tabulek. Operace vkládání jsou podmíněny kontrolou délky pole (např. `backup.users?.length > 0`), což zabraňuje pokusům o zápis prázdných dat. Díky zachování původních primárních klíčů v záložním souboru dojde k přesné rekonstrukci všech vazeb a relací. Celý tento robustní mechanismus garantuje, že po dokončení importu bude databáze v naprosto identickém a konzistentním stavu, v jakém se nacházela v momentě vytvoření zálohy.

8 Řešení technických výzev

Vývoj aplikace eZdraví se v průběhu implementace střetl s několika nestandardními technickými problémy, jejichž řešení si vyžádalo hlubší analýzu chování použitých technologií a přijetí netriviálních architektonických rozhodnutí. Tato kapitola dokumentuje tři nejvýznamnější výzvy, jejich příčiny a implementovaná řešení.

8.1 Nestabilita nativní komponenty pro výběr data na iOS

Prvním a z hlediska dopadu na uživatelskou zkušenost nejzávažnějším problémem byla nestabilita nativní komponenty pro výběr data na platformě iOS.

Popis problému

Komponenta `react-native-community/datetimepicker` zprostředkovává přístup k nativnímu systémovému prvku pro výběr data a času na platformách iOS a Android. Při použití parametru `display="inline"`, který zobrazuje picker jako vložený prvek přímo v rozhraní namísto modálního dialogu, docházelo na iOS k občasným pádům aplikace. Příčinou je sdílení nativního stavu UIKit komponenty při jejím opakovaném montování a unmontování v rámci React Native stromu komponent. Při unmontování komponenty iOS zanechává zkompromitovaný nativní stav, který způsobuje selhání při následném použití jakéhokoli pickeru ve stejné aplikační relaci.

Druhý, nezávislý problém se projevoval jako tzv. epoch bug. Datum uložené v SQLite databázi je persistováno jako textový řetězec ve formátu `YYYY-MM-DD`. Při načtení tohoto řetězce a jeho přímém předání konstruktoru `new Date()` dochází v JavaScriptovém enginu k interpretaci hodnoty jako UTC půlnoci. Vzhledem k tomu, že uživatelé v časových pásmech východně od UTC (včetně středoevropského časového pásma UTC+1 nebo UTC+2 v letním čase) pracují s lokálním časem posunutým vpřed, výsledný objekt `Date` po konverzi zpět do lokálního času reprezentoval předchozí den. Nativní komponenta `picker`, která jako vstup přijímá objekt `Date`, pak zobrazovala nesprávné datum. V krajním případě — například při přímém předání řetězce `"2024-01-01"` — docházelo k parsování jako UTC půlnoci 1. ledna 2024, které se v lokálním čase středoevropského uživatele projevilo jako 31. prosince 2023 ve 23:00 nebo 22:00, a picker se chybně inicializoval k datu 1. ledna 1970 (UTC epoch), pokud byl řetězec neplatný nebo prázdný.

Řešení

Oba problémy byly eliminovány zásadní architektonickou změnou: nativní iOS komponenta pro výběr data byla zcela nahrazena vlastní sdílenou komponentou `CalendarPicker` postavenou na knihovně `react-native-calendars`, která je implementována výhradně v JavaScriptu bez vazby na nativní UIKit prvky. Tím byl odstraněn celý zdroj problémů se sdíleným nativním stavem.

Klíčovým principem nového řešení je udržování stavu data po celou dobu životního cyklu komponenty výhradně v textovém formátu `YYYY-MM-DD` (ISO 8601). Tento formát je přímo kompatibilní s formátem používaným pro ukládání dat v SQLite, takže konverze při čtení z databáze a zápisu do ní nevyžaduje průchod přes objekt `Date`. Vizuální zobrazení data pro uživatele je generováno přímým parsováním textového řetězce — rok, měsíc a den jsou extrahovány jako samostatná čísla a předány konstruktoru `new Date(rok, měsíc - 1,`

den), přičemž tento konstruktor interpretuje hodnoty jako lokální čas a UTC posun se neuplatní. Výsledkem je, že datum vybrané uživatelem je vždy identické s datem uloženým v databázi bez ohledu na časové pásmo zařízení.

Pro výběr času, kde nativní spinner na iOS nadále zůstává nejvhodnějším řešením z hlediska uživatelské zkušenosti, je zachována knihovna `react-native-modal-datetime-picker` s parametrem `display="spinner"`. Spinner mode na rozdíl od inline mode nevyvolává problém sdíleného nativního stavu a zůstává stabilní.

Doplňkovým ochranným mechanismem je pomocná funkce `getSafeDate`, která ošetřuje potenciálně neplatné hodnoty dat načtených z databáze — prázdné řetězce, hodnotu `null` nebo nesmyslná historická data — a zabraňuje tak pádům aplikace při inicializaci datových komponent:

```
const getSafeDate = (val: any): Date => {
  if (!val || val === 'null' || val === '') return new Date();
  const d = new Date(val);
  if (isNaN(d.getTime()) || d.getFullYear() < 1990) return new Date();
  return d;
};
```

Obr. 31: Defenzivní ošetření datových hodnot (getSafeDate)
Zdroj: vlastní zpracování dle Zlatuška (2026)

8.2 Datová integrita zásoby léků při záznamu užití

Druhá výzva se týkala zajištění konzistentního stavu databáze při operaci záznamu užití léku, která vyžaduje atomickou aktualizaci dvou tabulek.

Popis problému

Datový model aplikace odděluje fyzickou zásobu léků (tabulka `inventory`) od záznamů o jednotlivých užitích (tabulka `medication_logs`). Toto oddělení je konceptuálně správné a popsané v kapitole 6, avšak přináší implementační požadavek: operace záznamu užití musí vždy zahrnovat dva databázové zápisy — vytvoření záznamu v tabulce `medication_logs` a snížení hodnoty `remaining_qty` v tabulce `inventory`. Pokud by druhá operace selhala nebo nebyla provedena, databáze by se ocitla v nekonzistentním stavu, kdy existuje záznam o odebrání léku bez odpovídajícího snížení zásoby.

Další výzvu představovalo ošetření hraničního stavu vyčerpání zásoby. Bez aktivní kontroly mohla hodnota `remaining_qty` klesnout pod nulu, pokud by bylo zaznamenáno více dávek, než odpovídalo skutečné zásobě. Rovněž bylo nutné zajistit automatickou archivaci krabičky ve chvíli, kdy je zásoba vyčerpána, bez nutnosti manuálního zásahu uživatele.

Řešení

Operace záznamu užití léku je implementována jako sekvenční blok, jehož logika před zápisem vždy ověřuje aktuální stav zásoby. Před vytvořením záznamu v `medication_logs` je načtena aktuální hodnota `remaining_qty` z tabulky `inventory`. Pokud by odečtení požadované dávky způsobilo pokles pod nulu, operace je přerušena a uživatel je informován o nedostatečné zásobě.

V případě, že je operace platná, jsou v jediném asynchronním bloku provedeny oba databázové zápisy. Pokud po odečtení dávky dosáhne `remaining_qty` hodnoty nula nebo nižší, je v témže bloku aktualizován stav krabičky: sloupec `status` je nastaven na `DEPLETED` a sloupec `depleted_at` je vyplněn aktuálním časovým razítkem.

Implementace tohoto řešení je zachycena v následující ukázce — nejprve je vytvořen záznam v tabulce `medicationLogs`, poté je snížena zásoba v `inventory`:

```
// Vytvoření nezvratného záznamu o užití léku
await db.insert(medicationLogs).values({
  plan_id: activeSosMed.plan_id,
  inventory_id: activeSosMed.inventory_id,
  scheduled_date: todayStr,
  scheduled_time: timeStr,
  taken_at: now.toISOString(),
  amount: amount,
  status: 'TAKEN'
});

// Snížení skladové zásoby v lékárničce
await db.update(inventory)
  .set({ remaining_qty: activeSosMed.remaining_qty - amount })
  .where(eq(inventory.inventory_id, activeSosMed.inventory_id));
```

Obr. 32: Ošetření integrity dat (Transakční logika léků)
Zdroj: vlastní zpracování dle Zlatuška (2026)

Druhá ukázka zachycuje ochrannou logiku při vyčerpání zásoby — pokud `remaining_qty` dosáhne nuly, systém nabídne dialogové okno pro archivaci prázdné krabičky. (viz Příloha 9)

Tento přístup zajišťuje, že stav zásoby v lékárničce vždy přesně odpovídá součtu zaznamenaných užití, a zároveň automaticky přesouvá vyčerpané krabičky do archivu bez nutnosti jakéhokoli manuálního úkonu ze strany uživatele.

Stejná logika zpětného přírůstku zásoby je implementována i pro operaci smazání záznamu o užití: při odebrání záznamu z `medication_logs` je příslušná hodnota `amount` přičtena zpět k `remaining_qty` a pokud byl stav krabičky `DEPLETED`, je obnoven na `ACTIVE`. Tím je zajištěna symetrie operací a konzistence dat i při retroaktivních opravách záznamu.

8.3 Navigační smyčky mezi propojenými entitami

Třetí výzva vycházela ze způsobu, jakým Expo Router spravuje navigační historii, a z požadavku na vzájemnou navigovatelnost provázaných entit aplikace.

Popis problému

Expo Router implementuje navigaci prostřednictvím zásobníkového modelu, v němž každé volání `router.push()` přidá novou obrazovku na vrchol zásobníku. Záznamy v aplikaci eZdraví jsou vzájemně provázány — detail diagnózy zobrazuje navázané návštěvy a léčebné režimy, detail návštěvy zobrazuje navázanou diagnózu a předepsané léky, detail léčebného režimu zobrazuje navázanou diagnózu. Bez ošetření by bylo možné vytvořit nekonečnou navigační smyčku: detail diagnózy, detail návštěvy, detail diagnózy, detail návštěvy, ..., přičemž každý krok by přidal novou obrazovku do zásobníku. Uživatel by musel opakovaně mačkat tlačítko Zpět k návratu do původního kontextu, čímž by byla zásadně narušena použitelnost aplikace.

Řešení

Problém navigačních smyček byl vyřešen implementací mechanismu kontextových navigačních parametrů, který lze přirovnat k principu breadcrumb navigace — každá obrazovka „ví", odkud uživatel přišel, a podle toho upravuje své chování.

Při navigaci z jedné detailní obrazovky na jinou jsou do URL parametrů předány identifikátory zdrojových entit: `from_disease_id`, `from_visit_id`, `from_plan_id` a `from_inventory_id`. Cílová obrazovka tyto parametry načte pomocí hooku `useLocalSearchParams` a využije je k podmíněnému vykreslení navigačních odkazů na propojené entity. Pokud identifikátor propojené entity odpovídá hodnotě příslušného `from_*` parametru, odkaz na tuto entitu je vizuálně zneaktivněn a nahrazen popiskem „Aktuálně zobrazeno". Tím je eliminována možnost navigovat zpět na obrazovku, ze které uživatel právě přišel, a navigační smyčka je přerušena.

Druhým prvkem řešení je strategické použití metody `router.replace()` namísto `router.push()` v kontextech, kde přidání nové obrazovky do zásobníku nedává sémantický smysl. Konkrétně po uložení nového záznamu prostřednictvím formulářové obrazovky je uživatel přesměrován na detail nově vytvořeného záznamu voláním `router.replace()`, nikoli `router.push()`. Formulářová obrazovka je tím nahrazena detailem v zásobníku namísto navrstvení, takže tlačítko Zpět vrátí uživatele přímo do kontextu, ze kterého formulář otevřel, a nikoliv zpět do prázdného formuláře. (viz Příloha 10)

Kombinace kontextových parametrů a selektivního použití `router.replace()` zajišťuje, že navigační zásobník v každém okamžiku odpovídá přirozené uživatelské cestě a tlačítko Zpět funguje intuitivně bez ohledu na hloubku navigace mezi propojenými záznamy.

```
const { id, from_plan_id, from_inventory_id, from_visit_id } = useLocalSearchParams();
const diseaseId = Number(id);
const fromPlanId = from_plan_id ? Number(from_plan_id) : null;
const fromInventoryId = from_inventory_id ? Number(from_inventory_id) : null;
const fromVisitId = from_visit_id ? Number(from_visit_id) : null;

const historyParams = {
  from_disease_id: diseaseId,
  from_plan_id: fromPlanId || '',
  from_inventory_id: fromInventoryId || '',
  from_visit_id: fromVisitId || '',
};
```

Obr. 33: Načtení kontextových parametrů pro prevenci navigačních smyček

Zdroj: vlastní zpracování dle Zlatuška (2026)

```
  } else {
    const result = await db.insert(diseases).values(payload).returning({ insertedId: diseases.disease_id });
    router.replace({ pathname: '/disease-detail', params: { id: result[0].insertedId } });
  }
```

Obr. 34: Nahrazení formuláře detailem v navigačním zásobníku (router.replace)

Zdroj: vlastní zpracování dle Zlatuška (2026)

8.4 Testování a nasazení aplikace

Funkční testování aplikace eZdraví probíhalo průběžně po celou dobu vývoje ve dvou prostředích. Primárním prostředím byl simulátor iOS dostupný v rámci vývojového nástroje Xcode, který umožňoval rychlé ověření funkcionality po každé změně kódu bez nutnosti fyzického přenosu sestavení na zařízení. Simulátor byl využíván zejména pro testování navigačních toků, správy databáze a vizuálního rozložení rozhraní.

Paralelně probíhalo testování na fyzickém zařízení iPhone 17 prostřednictvím aplikace Expo Go. Testování na reálném hardware bylo nezbytné zejména pro ověření funkcionalit závislých na hardwaru zařízení — biometrické autentizace, chování nativních komponent a celkové odezvy aplikace při reálném používání. Fyzické zařízení odhalilo několik problémů, které se na simulátoru neprojevovaly, mimo jiné nestabilitu nativní komponenty pro výběr data popsanou v kapitole 8.1.

V závěrečné fázi vývoje byla aplikace předána k neformálnímu uživatelskému testování členům rodiny. Testerům nebyl poskytnut žádný návod k použití. Cílem bylo ověřit, zda je rozhraní aplikace intuitivní bez předchozí instrukce. Zpětná vazba byla pozitivní a opakovaně zazněl zájem o to, aby byla aplikace dostupná pro širší veřejnost, což potvrzuje naplnění cíle vytvořit snadno použitelnou aplikaci pro správu zdravotních záznamů.

Aplikace nebyla v rámci této práce publikována v App Store ani Google Play. Distribuována byla výhradně prostřednictvím Expo Go v průběhu vývoje. Publikace v aplikačních obchodech by vyžadovala sestavení produkčního buildu pomocí nástroje EAS Build, získání vývojářského certifikátu Apple Developer Program a splnění požadavků procesu revize App Store. Tyto kroky jsou identifikovány jako přirozené navazující aktivity v rámci budoucího rozvoje aplikace.

Závěr

Tato bakalářská práce se zabývala vývojem mobilní aplikace pro osobní evidenci zdravotních záznamů, která reaguje na současnou potřebu digitalizace zdravotnictví a větší kontroly pacientů nad vlastními zdravotními údaji. Hlavním cílem bylo vytvořit uživatelsky přívětivou aplikaci, která umožní pacientům efektivně spravovat své zdravotní záznamy, včetně užívaných léků, prodělaných nemocí a návštěv u lékaře.

V rámci práce byla provedena důkladná analýza dostupných technologií pro vývoj mobilních aplikací, na jejímž základě byl zvolen framework React Native. Tato volba se ukázala jako optimální vzhledem k požadavkům na multiplatformní dostupnost, efektivitu vývoje a možnost využití široké škály knihoven pro implementaci klíčových funkcí.

Značná pozornost byla věnována návrhu uživatelského rozhraní a uživatelské zkušenosti, přičemž byl kladen důraz na přístupnost a použitelnost pro široké spektrum uživatelů. Aplikace byla navržena s ohledem na specifické potřeby zdravotnického sektoru, včetně vysokého kontrastu pro lepší čitelnost, intuitivní navigace a minimalizace rušivých elementů. Implementace těchto principů přispěla k vytvoření rozhraní, které je snadno použitelné i pro starší nebo technicky méně zdatné uživatele.

V oblasti správy dat bylo implementováno řešení využívající lokální databázi, které zajišťuje bezpečné ukládání citlivých zdravotních údajů a umožňuje práci s aplikací i bez připojení k internetu. Současně byla zajištěna možnost exportu dat ve formátu PDF pro snadné sdílení informací s lékaři.

Jádro vyvinuté aplikace tvoří modul pro komplexní správu zdravotních záznamů, který integruje evidenci medikace, diagnóz a lékařských prohlídek. Z bezpečnostního hlediska je systém navržen s ohledem na ochranu citlivých údajů prostřednictvím lokálního úložiště a biometrické autentizace uživatele. Pro vizualizaci zdravotní historie v čase byl implementován interaktivní kalendář. Celý ekosystém je pak doplněn o modul exportu dat, jenž zajišťuje snadné a bezpečné sdílení informací mezi pacientem a zdravotnickým personálem.

Přínos této práce spočívá především v poskytnutí nástroje, který pomáhá pacientům lépe kontrolovat a spravovat své zdravotní záznamy, což může vést k lepší komunikaci s lékaři a celkově efektivnější zdravotní péči. Do budoucna by bylo vhodné aplikaci rozšířit o další funkce, jako je například synchronizace dat s cloudem nebo integrace s existujícími zdravotnickými systémy.

Závěrem lze konstatovat, že vytvořená aplikace představuje významný krok směrem k digitalizaci správy osobních zdravotních záznamů a má potenciál přispět ke zlepšení kvality zdravotní péče prostřednictvím lepší informovanosti a aktivnějšího zapojení pacientů do péče o vlastní zdraví.

V rámci budoucího rozvoje aplikace by bylo vhodné implementovat pokročilejší správu paměti (tzv. garbage collection) u fotodokumentace. V současné verzi dochází při odstranění návštěvy ke smazání referencí z databáze, nicméně fyzické soubory zůstávají v lokálním úložišti zařízení, což by při dlouhodobém intenzivním používání mohlo vést k neefektivnímu využití paměti telefonu.

Zadání práce předpokládalo implementaci funkce oznámení (notifikací) pro připomínání užívání léků. Po analýze dostupných technologií bylo zjištěno, že lokální push notifikace v prostředí Expo Go vyžadují sestavení vlastního development buildu s modulem `expo-notifications`, který není součástí standardního Expo SDK. Vzhledem k zaměření práce na offline-first architekturu a lokální správu dat byla tato funkcionalita identifikována jako přirozené rozšíření pro příští vývojovou iteraci. V aktuální verzi aplikace je připomínání léků řešeno prostřednictvím domovské obrazovky, která při každém otevření zobrazuje dnešní naplánované dávky s přesnými časy podání.

S ohledem na budoucí rozvoj aplikace byla identifikována možnost optimalizace samotného zdrojového kódu. Během iterativního vývoje a rychlého prototypování narostla velikost některých souborů (zejména formulářových obrazovek). V další fázi životního cyklu softwaru by bylo vhodné provést hlubší refaktoring a extrahovat opakující se uživatelské prvky – jako jsou modální okna pro potvrzování akcí nebo specifické karty záznamů – do izolovaných a znovupoužitelných React komponent. Zdrojový kód aplikace je dostupný ve veřejném repozitáři na adrese: https://github.com/ogold66/ezdravi-bp.

Seznam použité literatury

<StrictMode>. Online. React. 2024a. Dostupné z: https://react.dev/reference/react/StrictMode. [cit. 2026-04-08].

About SQLite. Online. SQLite. 2024b, 2025-11-13. Dostupné z: https://www.sqlite.org/about.html. [cit. 2026-03-20].

ANTHROPIC. Claude. Online. Anthropic. 2025. Dostupné z: https://claude.ai. [cit. 2026-01-15].

APPLE INC. LocalAuthentication. Online. Apple. 2024b. Dostupné z: https://developer.apple.com/documentation/localauthentication. [cit. 2026-03-20].

APPLE INC. Swift. Online. Apple. 2024. Dostupné z: https://developer.apple.com/swift/. [cit. 2026-04-08].

ATHERTON, Jim. Development of the Electronic Health Record. Online. AMA Journal of Ethics. March 2011. Dostupné z: https://journalofethics.ama-assn.org/article/development-electronic-health-record/2011-03. [cit. 2026-04-08].

Core Components and Native Components. Online. React Native. 2024. Dostupné z: https://reactnative.dev/docs/intro-react-native-components. [cit. 2026-04-08].

ČESKÁ SPRÁVA SOCIÁLNÍHO ZABEZPEČENÍ (ČSSZ). eNeschopenka: základní informace pro lékaře a zdravotnická zařízení. Online. eNeschopenka. 2020. Dostupné z: https://www.cssz.cz/web/eneschopenka/zakladni-informace-pro-lekare-a-zdravotnicka-zarizeni. [cit. 2026-04-08].

ECONOMIA, A.S. VZP ukončila provoz portálu IZIP, nahradí jej vlastním webem za 12 milionů korun. Online. Hospodářské noviny. 2015. Dostupné z: https://domaci.hn.cz/c1-64937980-vzp-ukoncila-provoz-portalu-izip-nahradi-jej-vlastnim-webem-za-12-milionu-korun. [cit. 2026-01-03].

Expo DocumentPicker. Online. Expo Documentation. 2024ch. Dostupné z: https://docs.expo.dev/versions/latest/sdk/document-picker/. [cit. 2026-03-01].

Expo FileSystem. Online. Expo Documentation. 2024j. Dostupné z: https://docs.expo.dev/versions/latest/sdk/filesystem/. [cit. 2026-03-01].

Expo LocalAuthentication. Online. Expo Documentation. 2024g. Dostupné z: https://docs.expo.dev/versions/latest/sdk/local-authentication/. [cit. 2026-03-22].

Expo Print. Online. Expo Documentation. 2024e. Dostupné z: https://docs.expo.dev/versions/latest/sdk/print/. [cit. 2026-03-20].

Expo SecureStore. Online. Expo Documentation. 2024d. Dostupné z: https://docs.expo.dev/versions/latest/sdk/securestore/. [cit. 2026-03-20].

Expo Sharing. Online. Expo Documentation. 2024h. Dostupné z: https://docs.expo.dev/versions/latest/sdk/sharing/. [cit. 2026-03-01].

Expo Sqlite. Online. Expo Documentation. 2024c. Dostupné z: https://docs.expo.dev/versions/latest/sdk/sqlite/. [cit. 2025-03-20].

Expo Vector Icons Directory. Online. Expo Documentation. 2024i. Dostupné z: https://icons.expo.fyi/. [cit. 2026-03-01].

FAKULTNÍ NEMOCNICE OLOMOUC. Úvod do telemedicíny. Online. FNOL. 2023. Dostupné z: https://ntmc.fnol.cz/uvod-do-telemediciny. [cit. 2026-04-08].

FLUTTER. FAQ. Online. Flutter. 2024. Dostupné z: https://docs.flutter.dev/resources/faq. [cit. 2026-04-08].

GOOGLE. Gemini Advanced. Online. Google. 2025. Dostupné z: https://gemini.google.com. [cit. 2026-01-15].

INICIATIVA SNÍH. Telemedicína v České republice jako pomocník pro lepší dostupnost zdravotní péče. Online. Iniciativa Sníh. 2023. Dostupné z: https://www.iniciativa-snih.cz/telemedicina-v-cr-dostupnost-zdravotni-pece/. [cit. 2026-04-08].

INTERACTION DESIGN FOUNDATION. The Gestalt Principles. Online. Interaction Design Foundation, 2024b. Dostupné z: https://www.interaction-design.org/literature/topics/gestalt-principles. [cit. 2026-04-09].

INTERACTION DESIGN FOUNDATION. User Interface (UI) Design [online]. Interaction Design Foundation, 2024a. Dostupné z: https://www.interaction-design.org/literature/topics/ui-design. [cit. 2026-04-09].

INTERSYSTEMS CORPORATION, BOSTON, MA. FHIR: budoucnost interoperability ve zdravotnictví. Online. InterSystems. 2024. Dostupné z: https://www.intersystems.com/cz/reseni/fhir/. [cit. 2026-04-08].

Introduction to Expo Router. Online. Expo Documentation. 2024f. Dostupné z: https://docs.expo.dev/router/introduction/. [cit. 2026-03-22].

Introduction to Expo. Online. Expo Documentation. 2024a. Dostupné z: https://docs.expo.dev/get-started/introduction/. [cit. 2026-03-08].

KOUBOVÁ, Michaela. Většinu zdravotnických aplikací čeká certifikace u oznámených subjektů. Online. Zdravotnický deník. 2023. Dostupné z: https://www.zdravotnickydenik.cz/2023/01/vetsinu-zdravotnickych-aplikaci-ceka-certifikace-u-oznamenych-subjektu/. [cit. 2026-04-05].

KPMG ČESKÁ REPUBLIKA. Připravenost ČR na digitalizaci zdravotnictví. Online. KPMG. 2023. Dostupné z: https://kpmg.com/cz/cs/home/pro-media/tiskove-zpravy/2023/07/cesko-zavadi-digitalizaci-zdravotnictvi.html. [cit. 2026-03-08].

LEVÍČKOVÁ, Žaneta. Mobilní zdravotnictví s sebou nese mnohá rizika. Přes 80 procent aplikací porušuje obecné zásady. Online. iROZHLAS. 2022. Dostupné z: https://www.irozhlas.cz/veda-technologie/aplikace-mhealth-mobilni-zdravotnictvi-nukib-kyberbezpecnost-ochrana-osobnich_2212041555_lev [cit. 2026-04-08].

LOKAJOVÁ, Adéla a David ŠMAHEL. Výzkumná zpráva: Používání mHealth aplikací u dospívajících [online]. IRTIS, Masarykova univerzita. 2022. Dostupné z: https://irtis.muni.cz/cs/aktuality/novinky-a-clanky/vyzkumna-zprava-pouzivani-mhealth-aplikaci-u-dospivajicich. [cit. 2026-04-08].

MICROSOFT. The Basics. Online. TypeScript. 2024a. Dostupné z: https://www.typescriptlang.org/docs/handbook/2/basic-types.html. [cit. 2026-04-05].

MICROSOFT. TypeScript for JavaScript Programmers. Online. TypeScript. 2024b. Dostupné z: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. [cit. 2026-04-08].

MINISTERSTVO ZDRAVOTNICTVÍ ČR A ÚSTAV ZDRAVOTNICKÝCH INFORMACÍ A STATISTIKY ČR. Kapitola IX. Elektronizace zdravotnictví. Online. NZIP. 2025 Dostupné z: https://www.nzip.cz/data/koncepce2025/k09.pdf. [cit. 2026-04-05].

MINISTERSTVO ZDRAVOTNICTVÍ ČR. Národní strategie elektronického zdravotnictví ČR 2025–2035. Online. Praha: Ministerstvo zdravotnictví ČR. 2025. Dostupné z: https://mzd.gov.cz/narodni-strategie-elektronickeho-zdravotnictvi-cr-2025-2035/. [cit. 2026-04-08].

MINISTR ZDRAVÍ Z. Ú. Česko zaostává v digitalizaci zdravotnictví: Experti vyzývají politiky, aby se v předvolebních programech zavázali k nápravě. Online. Ministrzdravi. 2025a. Dostupné z: https://www.ministrzdravi.cz/aktualita/cesko-zaostava-v-digitalizaci-zdravotnictvi-experti-vyzyvaji-politiky-aby-se-v-predvolebnich-programech-zavazali-k-naprave/. [cit. 2026-03-06].

MINISTR ZDRAVÍ Z. Ú. Digitální gramotnost zdravotníků je zásadní pro úspěšnou digitální transformaci zdravotnictví. Online. Ministr zdraví. 2025c. Dostupné z: https://www.ministrzdravi.cz/digitalni- kompetence-ve-zdravotnictvi/. [cit. 2026-03-08].

MINISTR ZDRAVÍ Z. Ú. Evropská budoucnost digitalizace zdravotnictví. Jaké jsou vyhlídky Čechů do roku 2030? Online. Ministrzdravi. 2024. Dostupné z: https://www.ministrzdravi.cz/aktualita/evropska-budoucnost-digitalizace-zdravotnictvi-jake-jsou-vyhlidky-cechu-do-roku-2030/. [cit. 2026-03-22].

MINISTR ZDRAVÍ Z. Ú. Když zákon dohání realitu. Novela eHealth přináší základ, ale chybí návazný plán a motivace. Online. Ministrzdravi. 2025b. Dostupné z: https://www.ministrzdravi.cz/aktualita/novela-e-health/. [cit. 2026-03-07].

mZdraví. Často kladené otázky. Online. mZdraví. 2024. Dostupné z: https://www.mzdravi.cz/. [cit. 2026-04-08].

NATIONAL LIBRARY OF MEDICINE. The “PROMIS” of Computer-Based Medical Records. Online. Circulatingnow. 2019. Dostupné z: https://circulatingnow.nlm.nih.gov/2019/06/27/the-promis-of-computer-based-medical-records/. [cit. 2026-01-03].

NIELSEN, Jakob. 10 Usability Heuristics for User Interface Design [online]. Nielsen Norman Group, 1994. Jan. 30, 2024 Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/. [cit. 2026-03-09].

NORMAN, Don a NIELSEN, Jakob. The Definition of User Experience (UX). Online. Nielsen Norman Group. 1998. Dostupné z: https://www.nngroup.com/articles/definition-user-experience/. [cit. 2026-03-05].

OCCHINO, Tom. React Native: Bringing modern web techniques to mobile. Online. Engineering at Meta. 2015. Dostupné z: https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/. [cit. 2026-03-10].

OLSEN, Emily. Digital health apps balloon to more than 350,000 available on the market, according to IQVIA report. Online. MobiHealthNews. 2021. Dostupné z https://www.mobihealthnews.com/news/digital-health-apps-balloon-more-350000-available-market-according-iqvia-report. [cit. 2026-03-08].

PO MEDINĚ. Nové technologie ve zdravotnictví. Online. Pomedine. 2024. Dostupné z: https://www.pomedine.cz/nove-technologie-ve-zdravotnictvi/. [cit. 2026-03-05].

PRODUCT BY DRIZZLE TEAM. Drizzle migrations fundamentals. Online. Drizzle ORM. B. r. b. Dostupné z: https://orm.drizzle.team/docs/migrations. [cit. 2026-03-11].

PRODUCT BY DRIZZLE TEAM. Drizzle ORM. Online. Drizzle ORM. B. r. a. Dostupné z: https://orm.drizzle.team/docs/overview. [cit. 2026-03-11].

SCHAUB, Michael P., LEE, Jenny Yi-Chen, PIRONA, Alessandro. mHealth aplikace jako možnost intervence při řešení problematiky užívání drog a jeho následků. Online. Národní monitorovací středisko pro drogy a závislosti. 2019. Dostupné z: https://www.drogy-info.cz/data/obj_files/33056/855/Zaostreno%202_2019_mHealth%20aplikace.pdf. [cit. 2026-04-08].

SOMMERVILLE, Ian. Software Engineering. 10. vyd. Harlow: Pearson Education, 2016. ISBN 978-0133943030.

SQLite Foreign Key Support. Online. SQLite. 2024a. Dostupné z: https://www.sqlite.org/foreignkeys.html. [cit. 2026-03-20].

SQLite is Transactional. Online. SQLite. 2024c. Dostupné z: https://www.sqlite.org/transactional.html. [cit. 2026-03-20].

STÁTNÍ ÚSTAV PRO KONTROLU LÉČIV (SÚKL). eRecept: Pro pacienty se od 1. ledna 2018 nic nemění. Online. SÚKL. 2017. Dostupné z: https://sukl.gov.cz/media/tiskove-zpravy/erecept-pro-pacienty-se-od-1-ledna-2018-nic-nemeni/. [cit. 2026-04-08].

TERRELL, Katie Hanna, LEE, Kristen (ed.). Personal health record (PHR). Online. TechTarget. 2024. Dostupné z: https://www.techtarget.com/searchhealthit/definition/personal-health-record-PHR. [cit. 2026-04-08].

ÚŘAD PRO OCHRANU OSOBNÍCH ÚDAJŮ (ÚOOÚ). Zdravotnictví: otázky a odpovědi. Online. ÚOOÚ. 2024. Dostupné z: https://uoou.gov.cz/verejnost/qa-otazky-a-odpovedi/zdravotnictvi. [cit. 2026-04-08].

UseState. Online. React. 2024b. Dostupné z: https://react.dev/reference/react/useState. [cit. 2026-03-10].

VANIS, Karel; NAVRÁTIL, Marek a HLÁVKA, Jakub. Studie 2025: Digitalizace a elektronizace: chybějící pilíř českého zdravotnictví. Online. Iniciativa pro efektivní zdravotnictví. 2025. Dostupné z: https://www.efektivnizdravotnictvi.cz/post/studie-2025-digitalizace-a-elektronizace-chybejici-pilir-ceskeho-zdravotnictvi. [cit. 2026-04-04].

WALTER, Aaron. Designing for Emotion. New York: A Book Apart, 2011. ISBN 978-1937557003.

ZDRAVOTNÍ POJIŠŤOVNA MINISTERSTVA VNITRA ČR (ZP MV ČR). Zdravotní gramotnost v ČR se zlepšila. Online. ZP MV ČR. 2021. Dostupné z: https://www.zpmvcr.cz/o-nas/aktuality/zdravotni-gramotnost-v-cr-se-zlepsila. [cit. 2026-04-08].

Přílohy

Přílohy obsahují snímky obrazovek aplikace a zdrojové kódy.

Příloha 1 zachycuje onboardingový proces aplikace. Jde o sekvenci úvodních obrazovek při prvním spuštění.

Příloha 2 zobrazuje ukázku vygenerovaného PDF reportu zdravotní dokumentace.

Příloha 3 obsahuje SQL kód první migrační verze databázového schématu (`migrace 0000`) zachycující počáteční strukturu tabulek.

Příloha 4 zobrazuje SQL kód třetí migrační verze (`migrace 0002`), která přepracovala tabulku medications na flexibilní formát s JSON konfigurací dávkování.

Příloha 5 obsahuje SQL kód páté migrační verze (`migrace 0004`), která implementovala klíčové konceptuální oddělení lékárničky. Původní tabulka medications byla nahrazena dvěma novými tabulkami inventory a medication_plans.

Příloha 6 zobrazuje zdrojový kód zamykací obrazovky aplikace v souboru `_layout.tsx` implementující stavový automat biometrické autentizace.

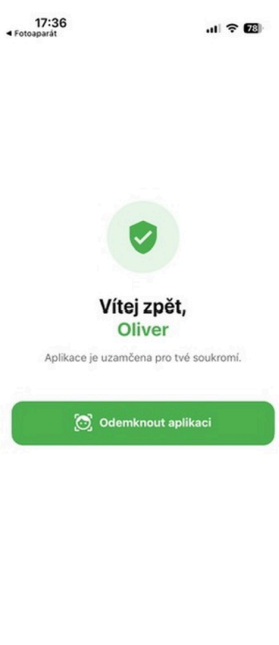
Příloha 7 zachycuje funkci `handleExportData` zajišťující export kompletní databáze do záložního souboru JSON a jeho sdílení prostřednictvím systémového dialogu.

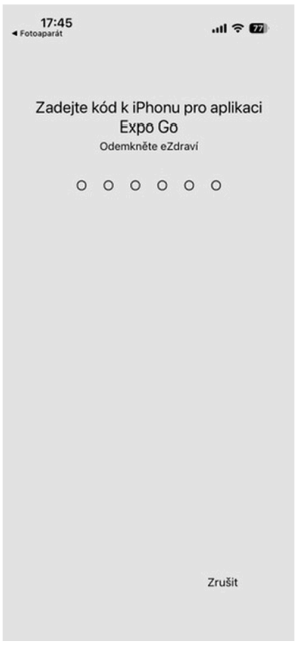
Příloha 8 zobrazuje logiku obnovy dat ze zálohy — kaskádové mazání stávající databáze a následný import záznamů z JSON souboru.

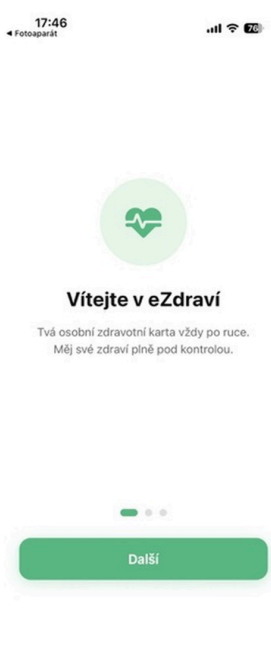
Příloha 9 obsahuje ochrannou logiku při vyčerpání zásoby léku — dialogové okno nabízející archivaci prázdné krabičky při pokusu o další odběr.

Příloha 10 zachycuje implementaci detailní obrazovky medikace s využitím kontextových navigačních parametrů `from_*` pro prevenci navigačních smyček.

Příloha 1

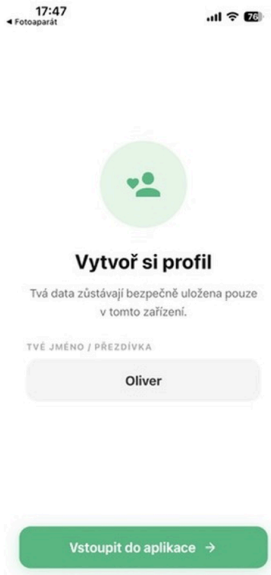

17:36
Fotoaparát
Vítej zpět,
Oliver
Aplikace je uzamčena pro tvé soukromí.
Odemknout aplikaci


17:45
Fotoaparát
Zadejte kód k iPhonu pro aplikaci Expo Go
Odemkněte eZdraví
Zrušit

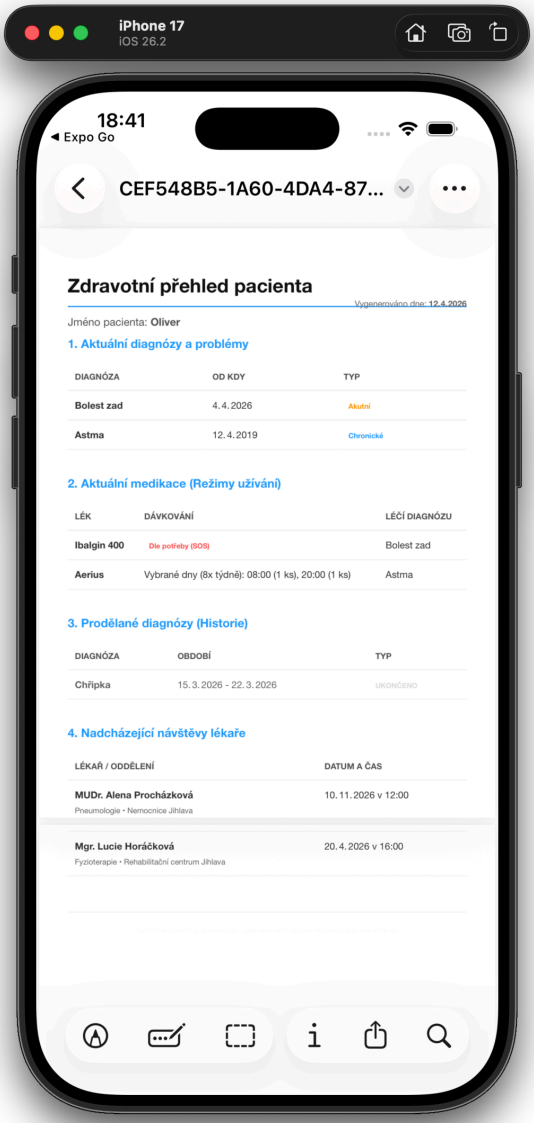

17:46
Fotoaparát
Vítejte v eZdraví
Tvá osobní zdravotní karta vždy po ruce.
Měj své zdraví plně pod kontrolou.
Další


17:46
Fotoaparát
Chytrá lékárnička
Sleduj zásoby a expirace svých léků. Už nikdy ti nedojdou léky ve špatnou chvíli.
Další


17:46
Fotoaparát
Návštěvy a Reporty
Uchovávej historii diagnóz a generuj PDF reporty pro svého lékaře na jedno kliknutí.
Vytvořit profil


17:47
Fotoaparát
Vytvoř si profil
Tvá data zůstávají bezpečně uložena pouze v tomto zařízení.
TVÉ JMÉNO / PŘEZDÍVKA
Oliver
Vstoupit do aplikace

Příloha 2


iPhone 17
iOS 26.2
18:41
Expo Go
CEF548B5-1A60-4DA4-87...
Zdravotní přehled pacienta
Vygenerováno dne: 12.4.2026
Jméno pacienta: Oliver
1. Aktuální diagnózy a problémy
DIAGNÓZA
OD KDY
TYP
Bolest zad
4.4.2026
Akutní
Astma
12.4.2019
Chronické
2. Aktuální medikace (Režimy užívání)
LÉK
DÁVKOVÁNÍ
LÉČÍ DIAGNÓZU
Ibalgin 400
Dle potřeby (SOS)
Bolest zad
Aerius
Vybrané dny (8x týdně): 08:00 (1 ks), 20:00 (1 ks)
Astma
3. Prodělané diagnózy (Historie)
DIAGNÓZA
OBDOBÍ
TYP
Chřipka
15.3.2026 - 22.3.2026
UKONČENO
4. Nadcházející návštěvy lékaře
LÉKAŘ / ODDĚLENÍ
DATUM A ČAS
MUDr. Alena Procházková
Pneumologie • Nemocnice Jihlava
10.11.2026 v 12:00
Mgr. Lucie Horáčková
Fyzioterapie • Rehabilitační centrum Jihlava
20.4.2026 v 16:00

Příloha 3

```
CREATE TABLE `diseases` (
    `disease_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `user_id` integer,
    `disease_name` text NOT NULL,
    `type` text NOT NULL,
    `note` text,
    `start_date` text,
    `end_date` text,
    FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
CREATE TABLE `medication_logs` (
    `medication_logs_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `medication_id` integer NOT NULL,
    `taken_at` text,
    `scheduled_for` text,
    `status` text,
    FOREIGN KEY (`medication_id`) REFERENCES `medications`(`medication_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
CREATE TABLE `medications` (
    `medication_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `user_id` integer,
    `disease_id` integer,
    `visit_id` integer,
    `medication_name` text NOT NULL,
    `form` text,
    `unit` text,
    `is_sos` integer DEFAULT false,
    `morning` integer DEFAULT false,
    `noon` integer DEFAULT false,
    `evening` integer DEFAULT false,
    `interval_hint` text,
    `start_date` text,
    `end_date` text,
    `total_qty` integer,
    `remaining_qty` integer,
    FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action,
    FOREIGN KEY (`disease_id`) REFERENCES `diseases`(`disease_id`) ON UPDATE no action ON DELETE no action,
    FOREIGN KEY (`visit_id`) REFERENCES `visits`(`visit_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
CREATE TABLE `users` (
    `user_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `name` text NOT NULL,
    `avatar` text
);
--> statement-breakpoint
CREATE TABLE `visits` (
    `visit_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `user_id` integer,
    `disease_id` integer,
    `hospital` text,
    `department` text,
    `date` text,
    `doctor` text,
    `note` text,
    `medical_report` text,
    `status` text,
    FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action,
    FOREIGN KEY (`disease_id`) REFERENCES `diseases`(`disease_id`) ON UPDATE no action ON DELETE no action
);
```

Příloha 4

```
PRAGMA foreign_keys=OFF;--> statement-breakpoint
CREATE TABLE `__new_medications` (
	`medication_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
	`user_id` integer,
	`medication_name` text NOT NULL,
	`form` text NOT NULL,
	`unit` text NOT NULL,
	`days_of_week` text DEFAULT '1,2,3,4,5,6,7',
	`doses_config` text NOT NULL,
	`total_qty` real NOT NULL,
	`remaining_qty` real NOT NULL,
	`created_at` text NOT NULL,
	FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
INSERT INTO `__new_medications`("medication_id", "user_id", "medication_name", "form", "unit", "days_of_week", "doses_config", "total_qty", "remaining_qty", "created_at")
SELECT "medication_id", "user_id", "medication_name", "form", "unit", "days_of_week", "doses_config", "total_qty", "remaining_qty", "created_at" FROM `medications`;--> statement-breakpoint
DROP TABLE `medications`;--> statement-breakpoint
ALTER TABLE `__new_medications` RENAME TO `medications`;--> statement-breakpoint
PRAGMA foreign_keys=ON;--> statement-breakpoint
CREATE TABLE `__new_medication_logs` (
	`log_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
	`medication_id` integer,
	`taken_at` text NOT NULL,
	`scheduled_for` text NOT NULL,
	`status` text NOT NULL,
	FOREIGN KEY (`medication_id`) REFERENCES `medications`(`medication_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
INSERT INTO `__new_medication_logs`("log_id", "medication_id", "taken_at", "scheduled_for", "status") SELECT "log_id", "medication_id", "taken_at", "scheduled_for", "status" FROM `medication_logs`;--> statement-breakpoint
DROP TABLE `medication_logs`;--> statement-breakpoint
ALTER TABLE `__new_medication_logs` RENAME TO `medication_logs`;--> statement-breakpoint
ALTER TABLE `users` ADD `created_at` text NOT NULL;--> statement-breakpoint
ALTER TABLE `users` DROP COLUMN `avatar`;
```

Příloha 5

```
CREATE TABLE `inventory` (
    `inventory_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `user_id` integer,
    `medication_name` text NOT NULL,
    `form` text NOT NULL,
    `unit` text NOT NULL,
    `total_qty` real NOT NULL,
    `remaining_qty` real NOT NULL,
    `expiration_date` text,
    `status` text DEFAULT 'ACTIVE',
    `created_at` text NOT NULL,
    `depleted_at` text,
    FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
CREATE TABLE `medication_plans` (
    `plan_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `user_id` integer,
    `inventory_id` integer NOT NULL,
    `disease_id` integer,
    `is_sos` integer DEFAULT false,
    `interval_hint` text,
    `doses_config` text NOT NULL,
    `start_date` text,
    `end_date` text,
    `created_at` text NOT NULL,
    FOREIGN KEY (`user_id`) REFERENCES `users`(`user_id`) ON UPDATE no action ON DELETE no action,
    FOREIGN KEY (`inventory_id`) REFERENCES `inventory`(`inventory_id`) ON UPDATE no action ON DELETE no action,
    FOREIGN KEY (`disease_id`) REFERENCES `diseases`(`disease_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
DROP TABLE `medications`;--> statement-breakpoint
PRAGMA foreign_keys=OFF;--> statement-breakpoint
CREATE TABLE `__new_medication_logs` (
    `log_id` integer PRIMARY KEY AUTOINCREMENT NOT NULL,
    `plan_id` integer NOT NULL,
    `scheduled_date` text,
    `scheduled_time` text,
    `taken_at` text,
    `amount` real NOT NULL,
    `status` text NOT NULL,
    FOREIGN KEY (`plan_id`) REFERENCES `medication_plans`(`plan_id`) ON UPDATE no action ON DELETE no action
);
--> statement-breakpoint
INSERT INTO `__new_medication_logs`("log_id", "plan_id", "scheduled_date", "scheduled_time", "taken_at", "amount", "status")
SELECT "log_id", "plan_id", "scheduled_date", "scheduled_time", "taken_at", "amount", "status" FROM `medication_logs`;--> statement-breakpoint
DROP TABLE `medication_logs`;--> statement-breakpoint
ALTER TABLE `__new_medication_logs` RENAME TO `medication_logs`;--> statement-breakpoint
PRAGMA foreign_keys=ON;
```

Příloha 6

```
// Ochranný blok: Vykreslení zamykací obrazovky. Zamykací obrazovka se ukáže jen tehdy, pokud je zámek zapnutý a aplikace NENÍ odemčená
if (isLockEnabled && !isUnlocked && !globalAlreadyUnlocked) {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center', backgroundColor: '#FFF', padding: 20 }}>
      <View style={{ width: 100, height: 100, borderRadius: 50, backgroundColor: '#E8F5E9', justifyContent: 'center', alignItems: 'center', marginBottom: 30 }}>
        <MaterialCommunityIcons name="shield-check" size={50} color="#4CAF50" />
      </View>
      <Text style={{ fontSize: 26, fontWeight: 'bold', color: '#111' }}>Vítej zpět,</Text>
      <Text style={{ fontSize: 26, fontWeight: 'bold', color: '#4CAF50', marginBottom: 15 }}>{userName || 'Uživateli'}</Text>
      <Text style={{ fontSize: 15, color: '#666', marginBottom: 50 }}>Aplikace je uzamčena pro tvé soukromí.</Text>

      <TouchableOpacity
        onPress={triggerAuth}
        activeOpacity={0.8}
        style={{ flexDirection: 'row', backgroundColor: '#4CAF50', paddingVertical: 18, paddingHorizontal: 30, borderRadius: 16, alignItems: 'center', gap: 10, width: '100%', justifyContent: 'center' }}
      >
        <MaterialCommunityIcons name="face-recognition" size={24} color="#FFF" />
        <Text style={{ color: '#FFF', fontSize: 16, fontWeight: 'bold' }}>Odemknout aplikaci</Text>
      </TouchableOpacity>
    </View>
  );
}

return (
  <Stack screenOptions={{ headerShown: false }}>
    <Stack.Screen name="(tabs)"/>
  </Stack>
);
};
```

```
const handleExportData = async () => {
  try {
    const backup = {
      users: await db.select().from(users).catch(() => []),
      diseases: await db.select().from(diseases).catch(() => []),
      visits: await db.select().from(visits).catch(() => []),
      inventory: await db.select().from(inventory).catch(() => []),
      medicationPlans: await db.select().from(medicationPlans).catch(() => []),
      medicationLogs: await db.select().from(medicationLogs).catch(() => []),
      visitDocuments: await db.select().from(visitDocuments).catch(() => []),
    };

    const jsonString = JSON.stringify(backup);

    const fileUri = fs.documentDirectory + 'ezdravi_zaloha.json';

    await fs.writeAsStringAsync(fileUri, jsonString, { encoding: 'utf8' });

    await Sharing.shareAsync(fileUri, { mimeType: 'application/json', dialogTitle: 'Exportovat zálohu eZdraví', UTI: 'public.json' });
  } catch (e: any) {
    Alert.alert('Chyba Exportu', e.message || 'Nepodařilo se vytvořit zálohu.');
  }
};
```

```
    Alert.alert(
      'Obnova dat',
      'Tato akce SMAŽE VŠECHNA AKTUÁLNÍ DATA v aplikaci a nahradí je daty ze zálohy. Opravdu chcete pokračovat?',
      [
        { text: 'Zrušit', style: 'cancel' },
        { text: 'Ano, přepsat data', style: 'destructive', onPress: async () => {
            try {
              await db.delete(medicationLogs).catch(() => {});
              await db.delete(medicationPlans).catch(() => {});
              await db.delete(inventory).catch(() => {});
              await db.delete(visitDocuments).catch(() => {});
              await db.delete(visits).catch(() => {});
              await db.delete(diseases).catch(() => {});
              await db.delete(users).catch(() => {});

              if (backup.users?.length > 0) await db.insert(users).values(backup.users);
              if (backup.diseases?.length > 0) await db.insert(diseases).values(backup.diseases);
              if (backup.visits?.length > 0) await db.insert(visits).values(backup.visits);
              if (backup.visitDocuments?.length > 0) await db.insert(visitDocuments).values(backup.visitDocuments);
              if (backup.inventory?.length > 0) await db.insert(inventory).values(backup.inventory);
              if (backup.medicationPlans?.length > 0) await db.insert(medicationPlans).values(backup.medicationPlans);
              if (backup.medicationLogs?.length > 0) await db.insert(medicationLogs).values(backup.medicationLogs);

              Alert.alert('Úspěch', 'Data byla úspěšně obnovena ze zálohy.');
              loadUsersAndSettings();
            } catch (e: any) { Alert.alert('Kritická chyba', 'Obnova selhala.'); }
        }}
      ]
    );
  } catch (e) { Alert.alert('Chyba', 'Nepodařilo se načíst soubor.'); }
};
```

```
// Ochranná logika při vyčerpání zásoby (při dalším pokusu o užití)
if (med.remaining_qty <= 0) {
  Alert.alert(
    'Prázdná krabička!',
    `Lék ${med.medication_name} už došel. Chcete prázdnou krabičku rovnou vyřadit do historie a vybrat z lékárničky novou?`,
    [
      { text: 'Zrušit', style: 'cancel' },
      {
        text: 'Vyřadit a změnit zdroj',
        style: 'default',
        onPress: async () => {
          await db.update(inventory).set({ status: 'DEPLETED', depleted_at: new Date().toISOString() }).where(eq(inventory.inventory_id, med.inventory_id));
          router.push({ pathname: '/medication-detail', params: { id: med.plan_id } });
        }
      }
    ]
  );
  return;
}
```

Příloha 10

```
const isFromThisPlan = med.plan_id == fromPlanId;
const isEnded = med.end_date !== null;
return (
  <TouchableOpacity key={index} style={styles.connectionCard} activeOpacity={0.7} disabled={isFromThisPlan} onPress={() => router.push({ pathname: '/medication-detail', params: { id: med.plan_id, ...historyParams } })}>
    <View style={{ width: 40, height: 40, borderRadius: 12, backgroundColor: isEnded ? '#F5F5F5' : '#E8F5E9', justifyContent: 'center', alignItems: 'center' }}>
      <MaterialCommunityIcons name={med.form === 'SYRUP' ? 'medication-outline' : 'pill-multiple'} size={22} color={isEnded ? '#AAA' : colors.fourth} />
    </View>
    <View style={{ marginLeft: 12, flex: 1 }}>
      <Text style={{ fontSize: 16, fontWeight: 'bold', color: isEnded ? '#888' : '#333' }}>{med.medName}</Text>
      <Text style={{ fontSize: 13, color: '#888' }}>{isFromThisPlan ? 'Aktuálně zobrazeno' : (med.isSos ? 'Podle potřeby (SOS)' : 'Pravidelný režim')}{isEnded ? ' • Ukončeno' : ''}</Text>
    </View>
```