EKONOMICKÁ FAKULTA TUL

Diplomová práce

Mobilní aplikace pro terénní sběr dat o odpadech

Studijní program:	N0688A140016 Systémové inženýrství a informatika
Autor práce:	**Bc. Michaela Grusmanová**
Vedoucí práce:	Mgr. Tomáš Žižka, Ph.D. Katedra informatiky

Liberec 2023

EKONOMICKÁ FAKULTA TUL

Zadání diplomové práce

Mobilní aplikace pro terénní sběr dat o odpadech

Jméno a příjmení:	**Bc. Michaela Grusmanová**
Osobní číslo:	E21000603
Studijní program:	N0688A140016 Systémové inženýrství a informatika
Zadávající katedra:	Katedra informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Frameworky pro vývoj mobilních aplikací
2. Výběr vhodného frameworku
3. Návrh UI/UX
4. Vývoj mobilní aplikace se zaměřením na odpadové hospodářství
5. Zhodnocení navržených přístupů a mobilní aplikace

Rozsah grafických prací:
Rozsah pracovní zprávy: 65 normostran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

- VERMEIR, Nico, 2022. *Introducing .NET 6: Getting started with Blazor, MAUI, Windows App SDK, Desktop Development, and Containers*. APress. ISBN 978-14-8427-318-0.
- GIRETTI, Anthony, 2022. *Beginning gRPC with ASP.NET Core 6: Build Applications using ASP.NET Core Razor Pages, Angular, and Best Practices in .NET 6*. APress. ISBN 978-14-8428-007-2.
- LACKO, Ľuboslav, 2017. *Mistrovství – Android*. Brno: Computer Press. Mistrovství. ISBN 978-80-2514-875-4.
- LAURENČÍK, Marek, 2018. *SQL: podrobný průvodce uživatele*. Praha: Grada Publishing. Průvodce. ISBN 978-80-2710-774-2.
- PROQUEST, 2022. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2022-09-30]. Dostupné z: http://knihovna.tul.cz/.

Konzultant: Bc. Vojtěch Starý, vedoucí oddělení Projekty, Inisoft s. r. o.

Vedoucí práce: Mgr. Tomáš Žižka, Ph.D.
Katedra informatiky

Datum zadání práce: 1. listopadu 2022
Předpokládaný termín odevzdání: 31. srpna 2024

L.S.

doc. Ing. Aleš Kocourek, Ph.D.
děkan

Ing. Petr Weinlich, Ph.D.
vedoucí katedry

V Liberci dne 1. listopadu 2022

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracovala samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědoma toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědoma následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. dubna 2023 Bc. Michaela Grusmanová

Mobilní aplikace pro terénní sběr dat o odpadech

Anotace

Cílem diplomové práce je vyvinout mobilní aplikaci pro usnadnění práce spojené s evidencí odpadů zejména v terénu uživatelům programů společnosti INISOFT s.r.o. (IS ENVITA a SKLAD Odpadů 8.). Vzájemnou výměnu dat těchto programů s mobilní aplikací nebo jiné důležité funkce aplikace (např. získání dostupných zařízení atd.) umožňuje komunikační server prostřednictvím protokolu gRPC. Aplikace je vyvíjena v programovacím jazyce C# ve frameworku .NET MAUI a umožňuje uživateli vkládat, ukládat či editovat záznamy o položkách odpadu a následně je odeslat nebo získat z ostatních programů společnosti INISOFT s.r.o.

Klíčová slova

C#, gRPC, MVVM, .NET MAUI, SQLite

Mobile application for field data collection of waste

Annotation

The aim of the thesis is to develop a mobile application for facilitating the work related to waste registration, especially in the field for users of INISOFT s.r.o. software (IS ENVITA, SKLAD Odpadů 8). The mutual data exchange of these programs with the mobile application or other important functions of the application (e.g. obtaining available devices, etc.) is enabled by the communication server via the gRPC protocol. The application is developed in the C# programming language in the .NET MAUI framework and allows the user to enter, save or edit records of waste items and then send or retrieve them from other INISOFT s.r.o. programs.

Keywords

C#, gRPC, MVVM, .NET MAUI, SQLite

Poděkování

Tímto bych chtěla poděkovat vedoucímu mé diplomové práce panu Mgr. Tomáši Žižkovi, Ph.D. za užitečné rady, konzultantům ze společnosti INISOFT s.r.o. za vedení projektu a trpělivost, svým kolegům a přátelům za pomoc při překonávání překážek a čas, který mi věnovali. V neposlední řadě bych chtěla poděkovat i své rodině za neustálou podporu.

Obsah

Seznam obrázků

Seznam použitých zkratek a názvů

AOT	Ahead-of-Time
API	Application Programming Interface
BCL	Base Class Library
CA	Certificate authority
CIL	Common Intermediate Language
CLR	Common Language Runtime
CSS	Cascading Style Sheets
DNS	Domain Name Systém
ES	Embedded Systems
gRPC	Google Remote Procedure Call
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
iniAPKA	název mobilní aplikace
INI SW	software společnosti INISOFT s.r.o.
IoT	Internet of Things
IL	Intermediate Language
IS	Information systém
JIT	Just-In-Time
JSON	JavaScript Object Notation
KS	komunikační server
MAUI	Multi-platform App User Interface
OS	Operating system
PWA	Progressive Web Apps
RZ	registrační značka
SDK	Software Development Kit
SSL	Secure Sockets Layer
SW	Software
TCP	Transmission Control Protocol
UI	User Interface
UID	ID registrace
VM	Virtual Machine

W3C	World Wide Consortium
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

Úvod

V dnešní době jsou mobilní aplikace stále populárnější a každým dnem vznikají nové. Tyto aplikace mohou být vytvořeny pro různé mobilní operační systémy a existuje mnoho technologií, které lze při jejich vývoji použít. Kromě přehledu o mobilních operačních systémech, typech vývoje aplikací a používaných technologiích je důležitá analýza požadavků a návrh vhodného řešení, které zajistí požadovanou podobu a funkčnost výsledné aplikace.

Tato diplomová práce je zaměřena na vývoj mobilní aplikace s názvem iniAPKA, především pro operační systém Android. První kapitoly jsou o mobilních operačních systémech, typech vývoje mobilních aplikací a k tomu používaných technologiích. Na základě této teoretické časti a analýze požadavků je poté rozhodnuto, jakým způsobem se bude výsledná aplikace této diplomové práce vyvíjet. Samotný vývoj iniAPKY je ještě rozdělen na teoretickou a praktickou část. V teoretické části jsou rozebírány důležité pojmy a způsoby vývoje ve vybrané technologii, v praktické části je již samotný popis vývoje a funkčnosti výsledné mobilní aplikace. Poslední kapitola této práce je věnována zhodnocení celého vývoje.

Aplikace, která vzniká společně s touto diplomovou prací, by měla usnadňovat evidenci odpadů uživatelům programů společnosti INISOFT s.r.o.

Současné programy společnosti INISOFT s.r.o. (dále INI SW, jedná se o programy pod obchodní značkou ENVITA a SKLAD Odpadů 8) staví na koncepci klient–server (PC desktop – databázový server MSSQL). Veškerá evidence se odehrává na desktopových klientech. Uživatelé tak prakticky nemají možnost potřebná data zapisovat mobilním zařízením online či offline v terénu a data synchronizovat do jejich aplikační databáze. Společnost INISOFT s.r.o. dosud nemá v nabídce mobilní aplikaci, která by výše uvedenou situaci dokázala řešit.

Mobilní aplikace má umožňovat zaznamenání údajů pro identifikaci odpadů, jejich množství, přebírající či předávající osoby a pořídit fotografii. Všechny zaznamenané údaje se musí uložit lokálně v mobilním zařízení pro ošetření situace, kdy není k dispozici připojení k síti. Následně mobilní aplikace po připojení k lokální síti tyto údaje odešle na server, a to standardizovaným způsobem přes moderní komunikační protokoly.

Výhoda vytvoření iniAPKY tedy spočívá v tom, že se zpracování dat získaných v terénu bude provádět na dotaz uživatele. Ten tak nebude muset pořízené údaje duplicitně zadávat a jeho pracovní proces se tak výrazně optimalizuje.

1 Mobilní operační systémy

Mobilní operační systém je software umožňující chytrým mobilním telefonům, tabletům či jiným chytrým mobilním zařízením spouštět aplikace a programy. Poskytuje mnoho důležitých funkcí jako např. volání, zasílání zpráv, rozhraní pro komunikaci mezi hardwarem a softwarem daného zařízení, správu připojení k mobilní a bezdrátové síti, možnost multitaskingu aj. (Vávrů a Ujbányai 2013)

Operačních systémů je hned několik, ale nejznámější a nejpoužívanější z hlediska mobilních telefonů jsou operační systémy iOS a Android. (Lacko 2018)

1.1 Android

Android je open source[1] mobilní operační systém vyvíjený společností Google založený na jádře Linuxu, díky kterému je zajišťováno zabezpečení systému, správa paměti a procesů, přístup k ovladačům vnitřních komponent a k síti. Aplikace mají přístup k funkcím jádra prostřednictvím Android API. Díky tomu lze vytvářet aplikace, které jsou pro uživatele přínosnější a užitečnější. Kromě linuxového jádra používá Android vlastní virtuální stroj, díky kterému je optimalizována paměť a hardwarové komponenty.

1.1.1 Historie vzniku Androidu

Počátky vývoje mobilních aplikací lze datovat do roku 2003, kdy byla založena společnost Android Inc. v Kalifornii a jejími zakladateli byli: Andy Rubin, Rich Miner, Nick Sears a Chris White. V roce 2005 odkoupila společnost Google celou firmu Android Inc. i s jejími stávajícím zaměstnanci. Díky tomu získala společnost Google nové a zkušené zaměstnance v oblasti mobilních technologií. V roce 2007 vznikl tým pod vedením jednoho ze zakládajících členů Android Inc. Andyho Rubina, který společnými silami vyvinul platformu pro mobilní zařízení na jádře OS Linux. V témže roce bylo tomuto týmu uděleno několik patentů. Oficiálně byla softwarová platforma Android představena 5. listopadu 2007. V tentýž den vzniklo sdružení více než

[1] Zdrojový kód je otevřený (snadno dostupný všem po technické i licenční stránce).

třiceti firem[2] s názvem Open Handset Alliance (zkratka OHA) s cílem vytvořit otevřené standardy mobilních zařízeních. V těchto dnech je toto sdružení tvořeno více než osmdesáti členskými společnostmi. Dne 23. září 2008 byl představen první produkt s mobilní platformou Android s jádrem Linuxu verze 2.6.25, kterým bylo HTC Dream (T-Mobile G1) s verzí Androidu 1.0.

Názvy jednotlivých verzí Androidu jsou pojmenovány dle názvu zákusků, které jdou v abecedě po sobě (Cupcake – verze 1.5, Donut – verze 1.6, Eclair – verze 2.0/2.1, Froyo – verze 2.2/2.2.3, Gingerbread – verze 2.3/2.3.7, Honeycomb – verze 3.0/3.2.6, Ice Cream Sandwich – verze 4.0/4.0.4, Jelly Bean – verze 4.1/4.3.1 atd.). (Vávrů a Ujbányai 2013)

Výrobci různých značek mobilních zařízení mohou operační systém Android upravovat pro svůj hardware při dodržení stanovených podmínek.

1.1.2 Komponenty Androidu

Jak již bylo zmíněno, tak operační systém Android je postaven na jádře Linuxu, díky kterému jsou zajištěny důležité funkce jako např. vytváření vláken, nízkoúrovňová správa paměti, zabezpečení atd. Výhodou implementace linuxového jádra v architektuře Androidu je usnadnění vývoje ovladačů zařízení jejich výrobcům.

Další důležitou částí je vrstva abstrakce hardwaru (Hardware Abstraction Layer – HAL), která poskytuje jednotné rozhraní (API) pro přístup k hardwaru daného zařízení. Je to tedy jakási abstraktní vrstva mezi hardwarem zařízení a jeho softwarem. Je složena z několika modulů knihoven, kde každý z nich implementuje rozhraní pro konkrétní hardware (např. kamera, Bluetooth, Audio, senzory atd.).

Každá aplikace, od Androidu verze 5.0 (úroveň API 21) nebo vyšší, běží ve vlastním procesu a s vlastní instancí Android Runtime – běhové prostředí pro aplikace (zkratka ART). Do verze 5.0 byl používán pouze virtuální stroj s názvem Dalvik. Myšlenka je taková, že se aplikace napsané v Javě, zkompilují do bajtkódu pro Java Virutal Machine, a tento bajtkód je zase přeložen do bajtkódu Dalvika – soubory s příponou

[2] Například HTC, Motorola, Google, T-Mobile, Intel atd.

DEX. Jedná se o speciálně navržený formát souborů pro Android, který je optimalizován pro minimální nároky na paměť a výkon zařízení. ART již oproti Davlikovi umí spustit více virtuálních strojů na zařízeních, a to spouštěním souborů DEX. Hlavními funkcemi ART jsou např.: Ahead-of-Time (AOT) a Just-In-Time (JIT) kompilace, optimalizovaný sběr odpadu (Garbage Collection – GC), od Androidu verze 9.0 lepší a kompaktnější převod DEX souborů na strojový kód a vylepšená podpora ladění. Pokud tedy aplikace běží dobře v Android Runtime, měla by běžet dobře i v Dalviku. Opačně to ale platit nemusí.

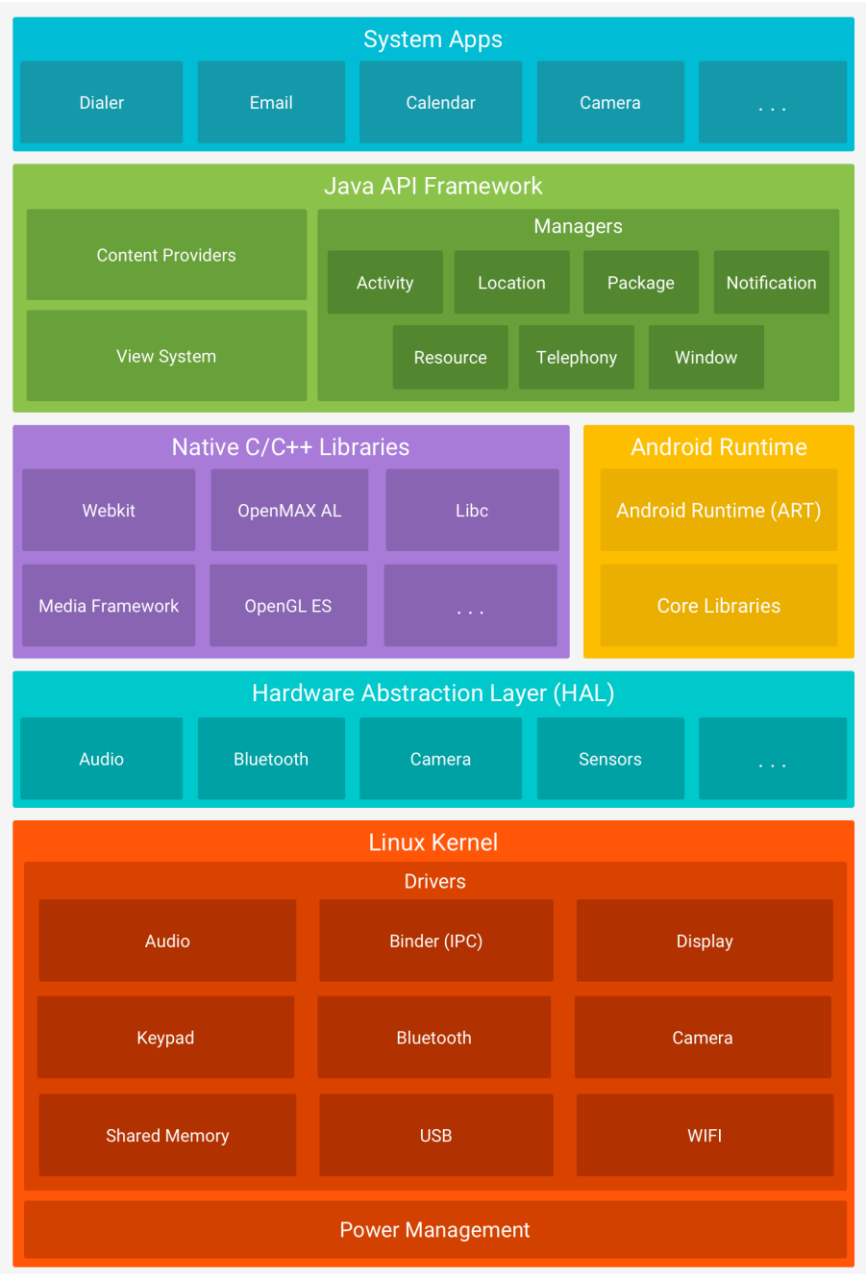
Mnoho hlavních systémových komponent a služeb (jako např. HAL a ART) je vytvořeno z nativního kódu, který využívá nativních knihoven napsaných v programovacích jazycích C a C++. Aplikace napsané pro Android pak tyto knihovny mohou používat, respektive jejich funkce, zavoláním Java API frameworku, který Android poskytuje (př. zavolání některé funkce kreslení z OpenGL ES pomocí rozhraní Java OpenGL API).

Veškeré funkce operačního systému Android jsou dostupné skrz Java API framework. Ten je tvořen z jednotlivých rozhraní pro různé části. Při vývoji aplikací pro Android se využívají právě tyto rozhraní. Java API Framework tedy obsahuje např.:

- **View System:** systém zobrazení, díky kterému je možné vytvořit uživatelské rozhraní aplikace (seznamy, mřížky, textové pole, tlačítka atd.);
- **Resource Manager:** správce zdrojů pro přístup k souborům bez kódu (obrázky, lokalizované soubory atd.);
- **Notification Manager:** správce notifikací pro zobrazení oznámení z aplikace ve stavovém řádku zařízení;
- **Activity Manager:** správce aktivit pro správu životního cyklu aplikace a poskytnutí navigace v aplikaci;
- **Content Providers:** poskytovatelé obsahu pro přístup k datům jiných aplikací (např. kontakty atd.).

Aplikace třetích stran mají možnost stejného přístupu k Java API frameworku jako mají systémové aplikace Androidu.

Android obsahuje balíček svých základních – systémových aplikací (např. e-mailový klient, SMS zprávy, kalendář, webový prohlížeč, kontakty aj.). Tyto systémové aplikace nemají zvláštní nastavení oproti aplikacím třetích stran, a proto je možné, nastavit si jako výchozí aplikaci i některou z nich (samozřejmě jsou zde výjimky, třeba pro Nastavení). Pokud by při vývoji aplikace pro Android byla potřebná funkce pro napsání e-mailu, lze zavolat jakoukoli nainstalovanou aplikaci, a tím odpadá jakákoli práce s implementací vlastního e-mailového klienta ve vyvíjené aplikaci. (Android Developers 2023)



Obrázek 1: Schéma komponent, ze kterých je složen OS Android
Zdroj: Android Developers 2023

1.2 iOS

IOS[3] je mobilní operační systém, který je vyvíjen společností Apple Inc. pro zařízení s názvem iPhone. Z iOSu byly odvozeny další samostatné operační systémy určené pro další zařízení od Apple Inc., a to pro iPad, Apple TV a Apple Watch. IOS je druhým nejpoužívanějším a nejoblíbenějším operačním systémem, prvním je OS Android.

1.2.1 Historie vzniku iOSu

Prvotní verze operačního systému společně s prvním zařízením iPhone byla vydána v roce 2007 a pojmenována jako iOS, což je zkratka pro iPhone operační systém (z angl. iPhone Operating System). Od té doby byla každý rok většinou v září vydána nová verze tohoto operačního systému. Aktuální verzí je iOS 16 (z roku 2022). (Spencer 2022)

1.2.2 Vrstvy iOSu

Architektura iOSu je složena ze čtyř základních vrstev: Cocoa Touch (Application Layer), Media, Core Services a Core OS, které kromě základních funkcí poskytují také API a frameworky pro vývoj aplikací.

Vrstva Cocoa Touch, aplikační vrstva, obsahuje klíčové funkce iOSu (dotykové a pohybové akce, multitasking, notifikace atd.).

Media vrstva obsahuje technologie umožňující práci s grafikou, zvukem a videem. Díky této vrstvě lze vytvářet graficky a zvukově propracované aplikace.

Vrstva Core Services poskytuje systémové služby, které jsou implementovány v jednotlivých rámcových strukturách (např. Adress Book Framework – umožňuje přístup ke kontaktním údajům uživatele, HealthKit Framework – zpracovává infor-

[3] IOS je také označení pro operační systém ve směrovačích a přepínačích od firmy Cisco. (Carthern et al. 2015)

mace o zdraví uživatele, HomeKit Framework – komunikace a ovládání chytrých zařízení v domácnosti atd.).

Poslední vrstvou je vrstva Core OS (např. Core Bluetooth Framework, Accelerate Framework atd.), která poskytuje nízkoúrovňové funkce ostatním technologiím. Je tedy velmi pravděpodobné, že v aplikacích nebudou volány přímo, ale jsou volány jinými frameworky. (Velu 2016)

2 Typy vývoje mobilních aplikací

Dle způsobu vývoje lze mobilní aplikace rozdělit do několika skupin: nativní, multiplatformní, webové a hybridní. (Azure Microsoft 2022)

2.1 Nativní aplikace

Nativní aplikace je vyvíjena pouze pro konkrétní operační systém v programovacím jazyce, který je pro něj definován (např. programovací jazyk Swift pro vývoj aplikací pro iOS, Kotlin pro Android atd.). Pro každou platformu jsou vyhrazené vývojové nástroje, SDK a prvky pro tvorbu UI.

Výhodou tohoto typu aplikace je to, že je vyvinutá přímo na míru pro daný operační systém, a díky tomu dosahuje vyššího výkonu, rychlé odezvy a může maximálně využít potenciál daného zařízení.

Nevýhodou je fakt, že vývoj aplikace pro každou platformu zvlášť je časově náročný a nákladný. Ze všech třech výše uvedených typů je tato varianta nejnákladnější. (Patadiya 2022)

Tento typ aplikace je vhodný pro vývoj výpočetně a graficky náročných projektů (např. hry, 3D modely aj.), nebo pokud je třeba, aby aplikace uměla pracovat i v offline režimu. (Bennett 2018)

Příkladem existujících nativních mobilních aplikací jsou WhatsApp, Spotify nebo Pokémon Go. (Patadiya 2022)

2.2 Multiplatformní aplikace

Multiplatformní aplikace je taková aplikace, která je vyvíjena pro více platforem zároveň. Vše je psáno v jednom kódu a v jednom jazyce frameworku[4], který multiplatformní vývoj podporuje (např. React Native atd.). Aplikace se následně tváří, že je napsána nativně.

[4] Framework je sada knihoven, nástrojů, funkcí atd., díky kterým je vývoj softwaru snazší.

Vývoj je mnohem rychlejší a méně nákladný, než je tomu u vývoje nativní aplikace. Údržba a aktualizace je také snadnější, neboť vše probíhá v jednotném zdrojovém kódu.

Nevýhodou je omezený přístup k hardware daného zařízení anebo kompatibilita použitého frameworku s aktuální verzí operačního systému. (Bennett 2018)

2.3 Webová aplikace

Webová aplikace se od obyčejné webové stránky liší v tom, že je více interaktivní. K vývoji se nejčastěji využívá HTML, CSS a JavaScriptu.

Běží ve všech prohlížečích, tím pádem i na jakémkoli zařízení, které podporuje webové prohlížeče. Uživatel nemusí nic stahovat ani instalovat, takže odpadají požadavky na úložiště. Výhodou je také to, že se zobrazení obsahu webové aplikace může přizpůsobit velikosti displeje daného zařízení, zde záleží na tom, jak je aplikace napsaná. Díky tomu, že je vše napsáno v jednom kódu, tak i jakákoli údržba nebo aktualizace se provádí poměrně snadno. Odpadá zároveň i starost a náklady spojené s nahráváním aplikace v případě aktualizace na App Store, Google Play nebo jiné místo.

Nevýhodou je to, že pro používání webové aplikace je nutný přístup k internetu. Bez toho aplikace fungovat nebude. Interakce s uživatelem není tak plynulá jako tomu je u nativní aplikace. Nevýhodou je také fakt, že nelze plně přistupovat ke každé části hardwaru daného zařízení.

Příklady těchto webových aplikací jsou Netflix nebo Trello (samozřejmě, že existují i stejnojmenné mobilní aplikace). (Patadiya 2022)

2.3.1 Progresivní webová aplikace

Označení progresivní webová aplikace (PWA) použila společnost Google v roce 2015 pro aplikace, které využívají výhody nových funkcí moderních webových prohlížečům, a to konkrétně service workers a webové aplikační manifesty. Díky nim je

totiž možné obohatit webovou aplikaci o funkcionality z nativního operačního systému. PWA mají dle vývojářů Google následující vlastnosti:

- **progresivní:** funguje na všech prohlížečích, neboť obsah aplikace je načítán postupně;
- **responzivní:** optimalizované zobrazení pro všechny druhy zařízení;
- **nezávislé na konektivitě:** možnost využívat aplikaci i offline díky technologii service workers;
- **app-like:** uživatel má při používání aplikace pocit, jako by pracoval přímo s nativní aplikací;
- **fresh:** aktualita dat díky technologii service workers (proces update);
- **zabezpečené:** komunikace probíhá přes HTTPS;
- **instalovatelné:** aplikace může být uchována na domovské obrazovce pro rychlé rozkliknutí bez nutnosti instalace z App Store nebo Google Play;
- **odkazovatelné:** možnost sdílení přes URL;
- **naleznutelné:** vyhledávače je umí najít díky W3C manifestům a registraci service workers;
- **znovuzapojení uživatele:** umožňují znovuzapojení uživatele pomocí např. push notifikace.

Webový aplikační manifest je ve formátu JSON. Díky němu je možné nainstalovat PWA na domovskou obrazovku. Funguje to tak, že se uživateli zobrazí dialogové okno, které se ho zeptá, zda si přeje přidat danou webovou aplikaci na domovskou stránku. Pokud uživatel souhlasí, tak se vše provede, a na tapetě se zobrazí ikona aplikace.

Service workers poskytují programovatelnou síťovou proxy, pomocí které se dají obsluhovat webové (HTTP) požadavky. Používají cachování a je možné používat aplikaci i v offline režimu. Pomocí této technologie jsou zajišťovány např. push notifikace, synchronizace dat na pozadí, zpracování HTTP požadavků nebo přijímání centralizovaných aktualizací.

Výhodou je již zmíněná možnost používání aplikace i v offline režimu, ušetření množství stažených dat, rychlejší přístup k aplikaci díky možnosti přidání ikony aplikace na domovskou obrazovku.

Nevýhodou je skutečnost, že se nedá v aplikaci využit všech dostupných hardwarových prvků daného zařízení. (Wargo 2020)

2.4 Hybridní aplikace

Hybridní aplikace je kombinací webového a nativního přístupu. Aplikaci lze stáhnout a nainstalovat jako nativní, a zároveň aplikace dokáže vykreslit webový obsah. Pro vývoj hybridní aplikace se využívá HTML, CSS a JavaScriptu. (Patadiya 2022) Aplikace běží ve webovém zobrazení – WebView, což je integrovaný prohlížeč, díky kterému je možné zobrazit webový obsah přímo v aplikaci.

Vývoj hybridní aplikace je rychlejší a levnější než u nativní aplikace. Také údržba a aktualizace je snadnější, protože vše probíhá v jednom zdrojovém kódu a aktualizovaná HTML stránka se pouze znovu nahraje na server, tudíž odpadá práce s vydáváním nové verze aplikace.

Hybridní aplikace nemají možnost přímého přístupu k hardwaru zařízení. Ten se ale dá zprostředkovat pomocí pluginů, které musí být kompatibilní s daným operačním systémem. Další nevýhodou je, že hybridní aplikace nedosahují takového výkonu jako nativní aplikace.

Tento typ mobilní aplikace se hodí v okamžiku, kdy je potřeba oslovit více potenciálních uživatelů skrz více platforem. (Cheng 2018)

Příkladem takto napsané již existující aplikace je Instagram, Gmail nebo Evernote. (Patadiya 2022)

3 Technologie pro vývoj mobilních aplikací

V následujících podkapitolách budou uvedeny a rozebrány nejpopulárnější technologie, které jsou v současné době nejvíce používány pro vývoj mobilních aplikací.

3.1 .NET

.NET je vývojářská open-source platforma pro vývoj různých druhů aplikací (např. webových, mobilních, desktopových, nativních pro cloud, hry, IoT[5] atd.) pro různé operační systémy (Android, iOS, Windows, Linux, macOS aj.). Spravuje ho společnost Microsoft a komunita GitHubu.

Každý rok v listopadu jsou vydávány nové verze .NETu. Pokud vyjde nová verze v lichý rok, je dlouhodobě podporována (LTS – Long-Term Support) po dobu tří let, pokud vyjde v sudý rok, je podporována krátkodobě (STS – Short-Term Support) po dobu osmnácti měsíců. (Microsoft 2022c)

V zásadě je .NET tvořen čtyřmi věcmi: jazykem, Visual Studiem, virtuálním strojem (CLR) a knihovnami.

V .NETu jsou pro vývoj k dispozici tři jazyky: C#, F# a Visual Basic.

Visual Studio je integrované vývojářské prostředí (IDE – Integrated Development Environment), které umožňuje psaní zdrojového kódu a pomáhá s vývojem. Má rozsáhlou integrovanou funkcionalitu a lze ho spustit v operačním systému Windows, nicméně existuje verze i pro macOS.

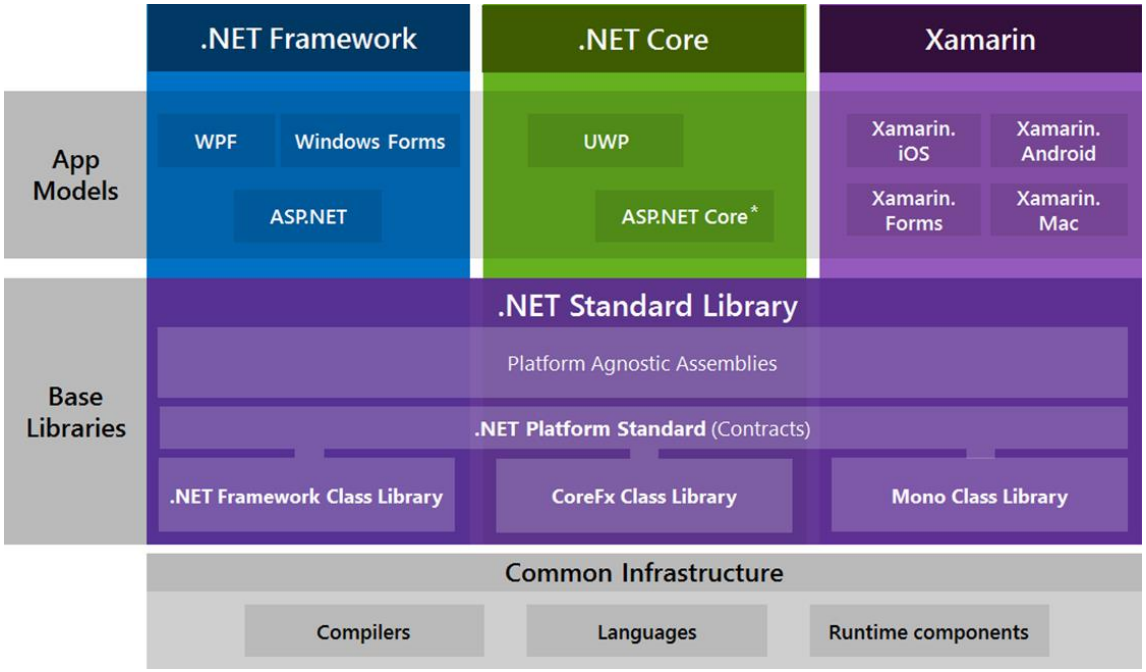
Modul CLR (Common Language Runtime) je virtuální stroj, který interpretuje mezikód, označující Microsoftem jako CIL (Common Intermediate Language), do instrukcí fyzického prostoru.

[5] Internet of Things je označení pro síť chytrých zařízeních, které dokáží automaticky provádět dané úkony (např. chytrý termostat, žárovky atd.). (Pathak a Bhandari 2018)

Největší výhodou .NETu jsou knihovny, ve kterých jsou připraveny řady struktur a komponent, díky kterým je vývoj rychlejší a snazší. (Price 2022)

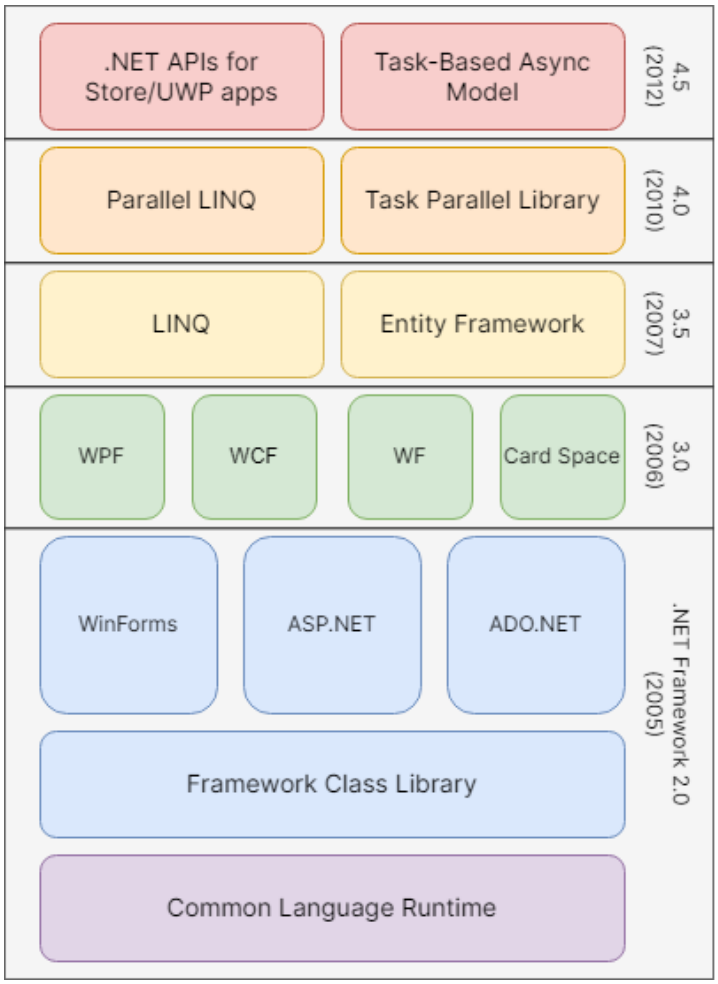
Vytvořená aplikace je ze zdrojového kódu sestavována pomocí MSBuild. (Microsoft 2022c) K tomuto procesu je dále potřeba soubor s koncovkou .csproj (v případě, že je aplikace vyvíjená v C#), .fsproj (pokud v F#) nebo .vbproj (pokud ve Visual Basic). Tento soubor obsahuje MSBuild XML kód, díky kterému při sestavování aplikace IDE importuje veškerá potřebná nastavení tak, aby byl projekt úspěšně sestaven. (Microsoft 2022a)

.NET existuje v několika vydáních a každé z nich podporuje jiný typ aplikace. První vydání je .NET Framework, druhý Mono a třetí .NET (Core). (Microsoft 2022c)



Obrázek 2: Struktura jednotlivých implementací .NET
Zdroj: Torre 2016

.NET Framework již není aktivně vyvíjen, avšak jeho nejnovější verze (verze 4.8 z roku 2019) je stále podporována. Na následujícím obrázku je znázorněna struktura .NET Frameworku. V každé verzi je přidána nová funkcionalita. Ve verzi .NETu 2.0 je pouze CLR a základní knihovna Framework Class Library. S verzí .NET 3.0 přichází nástroje pro tvorbu formulářových aplikací. Verze 3.5 umožňuje pracovat s dotazovacím jazykem LINQ a Entity Frameworkem. Další velká změna byla ve verzi 4.5, která usnadnila psaní asynchronních funkcí, které jsou pro vývoj mobilních aplikací velice důležité.



Obrázek 3: Struktura .NET Frameworku
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

Mono se vyvinulo jako podpora .NETu na operačním systému Linux. Dlouho se ale neudrželo a bylo vytlačeno novějším .NET Core. Dnes je Mono vyvíjeno společností Xamarin (vlastněnou Microsoftem).

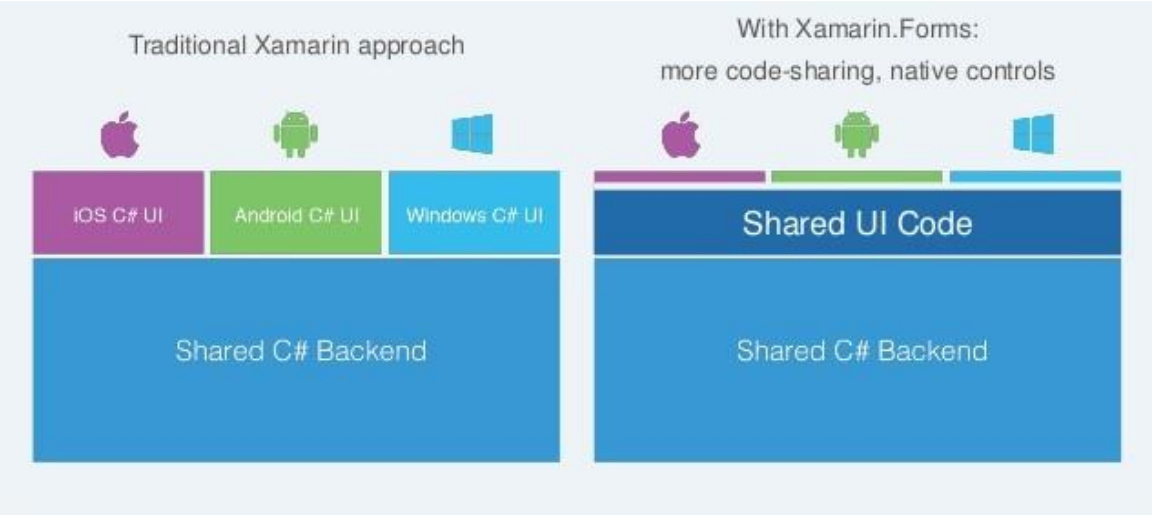
Aktivně vyvíjenou implementací je .NET Core. Nahrazuje výše zmíněný .NET Framework. Od páté verze se používá pouze označení .NET. (Price 2022) Aktuální verzí je .NET 7, která vyšla 8. listopadu 2022. V .NETu je možné vyvíjet aplikace nejen pro OS Windows, ale i pro jiné OS, např. Linux, Android nebo iOS. Mezi další jeho výhody patří kompatibilita s příbuznými technologiemi (.NET Framework a Xamarin – Mono). Tato kompatibilita je umožněna díky .NET Standardu – specifikaci, určující společnou funkcionalitu implementací .NETu. (Price 2022)

3.1.1 Xamarin a Xamarin.Forms

Xamarin je platforma pro nativní vývoj mobilních aplikací pro iOS a Android. (Bennett 2018) Jedná se o open-source framework, který vlastní a spravuje společnost Microsoft od roku 2016. Původně byl totiž vyvíjen společností Xamarin, která byla založena v roce 2011. (Petzold 2015) Pro vývoj v Xamarinu se využívá programovacího jazyka C# a značkovacího jazyka XAML. Aplikace je napsána v jednom kódu, jenž je dále překládán do nativních jazyků jednotlivých operačních systémů *(Android – Kotlin, iOS – Swift).* (Bennett 2018) Je vhodné ho využít pro vývoj aplikací, kde uživatelské rozhraní jednotlivých platforem má vypadat odlišně.

Open-sourcový framework Xamarin.Forms je velmi podobný výše zmíněnému Xamarinu. Je určen pro vývoj nativních aplikací s multiplatformní sadou nástrojů .NET v programovacím jazyce C# a značkovacím jazyce XAML. Umožňuje vytvářet aplikace v jednom sdíleném kódu pro operační systémy: Android, iOS a Windows. (Microsoft 2021) Na rozdíl ale od samotného Xamarinu umožňuje psát uživatelské rozhraní pro všechny platformy dohromady. (Petzold 2015) Rozdíl mezi přístupy k vývoji mezi Xamarinem a Xamarinem.Forms znázorňuje Obrázek 4.

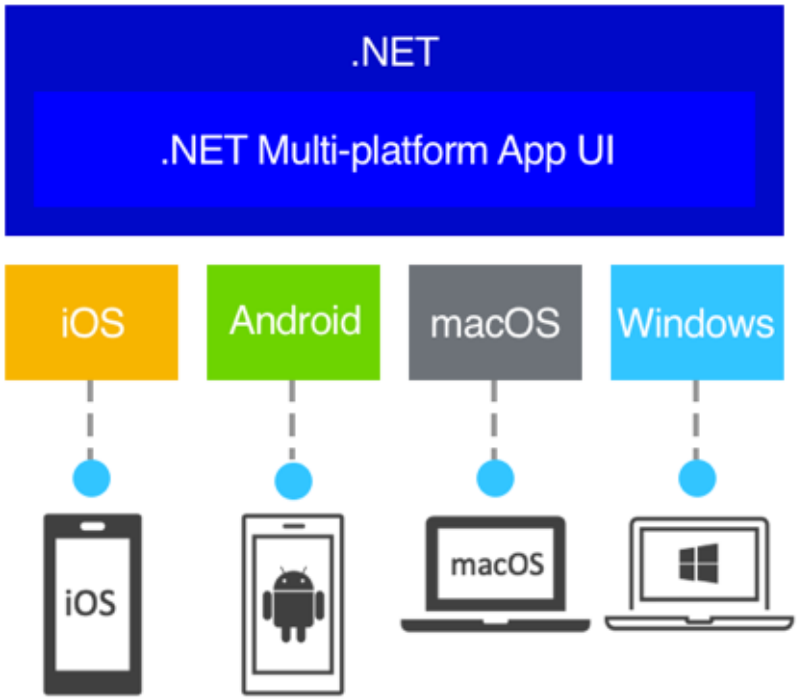
Nevýhodou je, že neumožňuje tzv. hot reload, takže pro aplikaci provedených změn, je potřeba aplikaci opětovně sestavit (Petzold 2015), a další nevýhodou je také skutečnost, že společnost Microsoft oznámila blížící se konec podpory pro Xamarin SDK, a to k datu 1. května 2024. (Microsoft 2022d)



Obrázek 4: Rozdíl mezi Xamarinem a Xamarinem.Forms
Zdroj: Whalen-Dunn 2019

3.1.2 .NET MAUI

.NET MAUI (Multi-platform App User Interface) je multiplatformní open-source framework pro vývoj mobilních aplikací pomocí programovacího jazyka C# a značkovacího jazyka XAML. Vytvářet aplikace lze pro následující OS: Android, iOS, macOS, Windows a Tizen. Je evolucí již zmíněného frameworku Xamarin.Forms a rozšiřuje ho o podporu vývoje desktopových aplikací. Jedním z hlavních cílů .NET MAUI bylo to, aby implementace logiky a uživatelského rozhraní byla co možná největší a pouze v jednom kódu, avšak není vyloučena možnost přidání nebo úprava kódu pro specifickou platformu.



Obrázek 5: .NET a .NET MAUI
Zdroj: Microsoft 2022b

.NET MAUI sjednocuje jednotlivá API všech výše zmíněných platforem do jednoho API. .NET 6 a vyšší verze .NETu poskytují specifické frameworky pro tvorbu aplikací pro konkrétní platformu, a to konkrétně .NET pro Android, .NET pro iOS, .NET pro macOS a WinUI3 (knihovna pro Windows). Všechny tyto frameworky přistupují k sadě základních knihoven zvané .NET BCL. Ta je závislá na .NET runtime, což je tzv. běhové prostředí, které spouští aplikace a usnadňuje vývoj. Pro Android, iOS a macOS je tímto prostředím Mono runtime a pro Windows .NET CoreCLR.

Aplikace pro Android je nejprve zkompilována ze C# do IL, ze kterého je následně JIT kompilací sestavena nativní aplikace. Sestavení aplikace pro iOS zase probíhá pomocí AOT kompilace. Aplikace pro macOS je vytvořena pomocí systému Mac Catalyst od Applu, který iOS aplikaci převede na desktopovou. Windows aplikace je sestavena pomocí knihovny WinUI3. K vytvoření aplikací pro iOS a macOS je potřeba fyzické zařízení Mac.

Výhoda v .NET MAUI je možnost hot reloadu, který umožňuje aplikovat změny v kódu aplikace ihned za běhu, bez nutnosti aplikaci opětovně kompilovat a spouštět, což ušetří mnoho času. Další výhodou je velký seznam UI komponent, které je možné použít. (Microsoft 2022b)

3.2 React Native

React Native je open-source framework určený pro multiplatformní vývoj mobilních aplikací. Lze v něm vyvinout aplikaci pro Android a iOS. Je úzce spojen s javascriptovou knihovnou React, kterou má i ve svém názvu, a využívá stejných zápisů a principů jako tato knihovna. Rozdíl mezi React Nativem a Reactem je v tom, že React Native necílí na vývoj dynamických webových stránek jako React, ale na vývoj mobilních aplikací. React i React Native jsou díla společnosti Facebook (Eisenman 2015) (od roku 2021 nese tato společnost název Meta). (Meta 2021)

React Native je postaven na specifikaci jazyka JSX – JavaScript, překládaný do Javy (v případě Androidu) nebo Objective-C (v případě iOS), a XML. React Native nevyužívá HTML a CSS jako React pro vytváření uživatelského rozhraní aplikace, ale vykreslování jednotlivých komponent probíhá nativně pomocí UI knihoven každé platformy. Stylování prvků uživatelského rozhraní probíhá pomocí stylů definovaných v JavaScriptu. Tyto styly jsou strukturou velmi podobné CSS. (Eisenman 2015)

Typické pro React a React Native je jejich svižnost, což ocení především uživatelé, a znuvupoužitelnost komponent, díky které je vývoj rychlejší a snazší.

Díky tomu, že je aplikace napsána pouze v JavaScriptu, není nutné po provedených změnách ji znovu celou sestavovat. Stačí pouze obnovit aplikaci, podobně jako tomu je např. u webových stránek, což významně šetří čas.

Nevýhodou je skutečnost, že některé části vývoje musí být implementovány zvlášť pro každou platformu. (Eisenman 2015)

3.3 Flutter

Tento open-source framework, vytvořen společností Google v roce 2015 (Zaccagnino 2020), je určen pro vývoj mobilních aplikací pro Android a iOS v programovacím jazyce Dart. První stabilní verze Flutteru – verze 1.0 byla vydána až v roce 2018, aktuální verzí je Flutter 2 z roku 2021. (Flutter 2023)

Výhodou vývoje aplikací ve Flutteru je existence velkého množství knihoven s komponentami, jež se dají stáhnout a upravit, což značně urychluje vývoj, a také to, že aplikace napsané ve Flutteru jsou podporovány i staršími zařízeními. (Zaccagnino 2020) Nevýhodou je skutečnost, že velikost aplikace napsané ve Flutteru je větší než velikost nativní aplikace.

Programovací jazyk Dart byl vyvinut též společností Google. Je objektově orientovaný a syntaxí velmi podobný programovacím jazykům Java a C#, z čehož vyplývá, že pro Java a C# vývojáře není velkým úskalím, naučit se programovat i v Dartu. Díky JIT kompilaci a Dart VM je možný hot reload. Aplikace napsaná v Dartu je do nativního kódu zařízení zkompilována pomocí AOT kompilace. Díky tomuto překladu do nativního kódu je aplikace celkově rychlejší.

Prvky uživatelského rozhraní jsou tvořeny widgety. Každý widget má své definované vlastnosti (velikost, barvu, stav atd.) a jsou vykreslovány 2D grafickým enginem Skia. Flutter poskytuje sadu widgetů dodržujících zásady Material Designu pro Android a Cupertina pro iOS, takže je velmi snadné vytvořit za krátký čas přívětivé grafické rozhraní aplikace. Všechny své dostupné widgety má Flutter ve svém katalogu, v němž lze najít i jejich popis. (Flutter 2023)

3.4 Apache Cordova

Apache Cordova je open-source framework, pomocí kterého je možné vyvíjet multiplatformní aplikace. Byl vytvořený v roce 2009 společností Nitobi a nesl název PhoneGap. (Wargo 2015)

Pro vývoj aplikací je využíváno standardních webových technologií, tedy HTML, CSS a JavaScript. Spuštěná aplikace běží na mobilním zařízení ve WebView jako webová stránka.

Výhodou je, že Cordova má implementované své vlastní pluginy, které vytváří rozhraní mezi nativními komponentami a frameworkem, což poskytuje obousměrnou komunikaci, která se hodí v případě, že je potřeba volat nativní kód z JavaScriptu (např. otevřít galerii, přístup ke kontaktům nebo k fotoaparátu atd.). Další silnou stránkou je možnost testování aplikace rovnou v prohlížeči, ovšem za podmínky, že se nevolá některá nativní funkce mobilního zařízení.

Tento framework neumožňuje pracovat s prvky pro tvorbu uživatelského rozhraní, tzn. komponenty pro tlačítka, navigaci, záložky atd. Lze ho ale rozšířit o widgety, které to dokáží nahradit. Těmito widgety jsou např. Ionic, Onsen UI nebo Framework7. (Apache Cordova 2023)

3.5 Ionic

Ionic je open-source framework pro vytváření mobilních, desktopových a webových aplikací s využitím webových technologií: HTML, CSS a JavaScript. Využívá se hlavně pro vytváření uživatelského rozhraní. Je možné ho přidat do Angular, React či Vue projektů a dalších. Každopádně lze vytvořit i samotný Ionic projekt.

Starší verze Ionicu byly velmi úzce propojeny s frameworkem Angular. Verze 4.x je již navržena tak, aby Ionic mohl fungovat jako samostatná knihovna komponent, a mohl tak být součástí i jiných frameworků. Lze se setkat tedy s označeními jako je Ionic Angular, Ionic React či Ionic Vue. Do budoucna se počítá také i s podporou jiných frameworků.

Výhodou Ionicu je jeho rychlost, kterou dosahuje především díky hardwarové akceleraci a optimalizaci dotykových gest. Další výhoda spočívá v předem připravené sadě různých komponent, typografie a motivů. A největší výhodou Ionicu je jedna kódová základna pro všechny typy aplikací, takže vývoj je mnohem snazší a rychlejší.

Slabší stránkou Ionicu je jeho omezenější přístup k hardwaru zařízení. K nativnímu nasazení mobilní aplikace je nutné integrovat technologii Capacitor (nebo Cordova). (Ionic 2023)

3.6 Nativní jazyky

Od roku 2017 je Kotlin oficiálním jazykem pro nativní vývoj mobilních aplikací pro Android. Je to objektově orientovaný, kompilovaný jazyk s typovou kontrolou, vyvinutý společností JetBrains. Zkompilovaný kód je kompatibilní s Javou, díky tomu je tedy možné Java knihovny využít i v Kotlinu. (Laurence 2021)

Kotlin Multiplatform Mobile je SDK (zatím pouze ve verzi beta), které zjednodušuje vývoj mobilních aplikací a to tak, že je možné psát společný kód pro aplikace určené pro Android a zároveň pro iOS. (Kotlin 2022)

Pro nativní vývoj mobilních aplikací pro operační systém iOS se od roku 2014 používá programovací jazyk Swift, který nahradil programovací jazyk Objective-C. Jedná se opět o objektově orientovaný, kompilovaný jazyk s typovou kontrolou. Nicméně se nepoužívá pouze pro vývoj mobilních aplikací, ale i pro vývoj utilit určených pro MacBook, aplikací pro Apple Watch nebo webových aplikací. (Lacko 2018)

4 Analýza požadavků a návrh řešení

Pro vývoj jakéhokoli softwaru je důležitá analýza požadavků. Cílem je definovat požadavky kladené na vyvíjený software a zároveň popsat jeho funkčnosti. Jelikož se software vyvíjí postupně, tak i požadavky na něj se v průběhu vývoje mění. Dá se tedy říci, že tato analýza je kontinuální proces.

4.1 Funkční požadavky

Funkční požadavky určují chování a služby, které bude budoucí systém nabízet uživatelům.

Ke specifikaci funkčních požadavků je možné využít technik jako např. diagram případů užití, sekvenční diagram, případně diagram aktivit. (Wiegers a Hokanson 2023)

Funkční požadavky pro vývoj iniAPKY byly rozděleny do třech fází z důvodu postupného hodnocení a sladění aplikace s představami společností INISOFT s.r.o.: 1. fáze (Proof of Concept), 2. fáze (Funkční minimum) a 3. fáze (Minimum Viable Product).

V 1. fázi jsou hlavními body:

- vytvoření databáze pro ukládání dat;
- navázání spojení s KS;
- vytvoření, editace a odstranění záznamu.

Ve 2. fázi jsou hlavní body následující:

- autentizace uživatele;
- vyžádání záznamů z KS a jejich vykreslení uživateli s možností úpravy;
- odeslání záznamu z KS.

A v poslední fázi:

- jednodušší vyplnění formuláře pro vytvoření záznamu pomocí např. skenování 1D/2D kódů;
- rozšíření možností formulářových prvků (např. přidání fotografie aj.).

4.2 Nefunkční požadavky

Tento typ požadavků specifikuje, jak bude systém implementován. Vyvíjená aplikace iniAPKA by měla mít jednoduché GUI. Její provoz by neměl být závislý na internetovém připojení (synchronizace s KS by měla probíhat pouze v lokální síti) a měla by být dostupná pouze pro uživatele IS ENVITA.

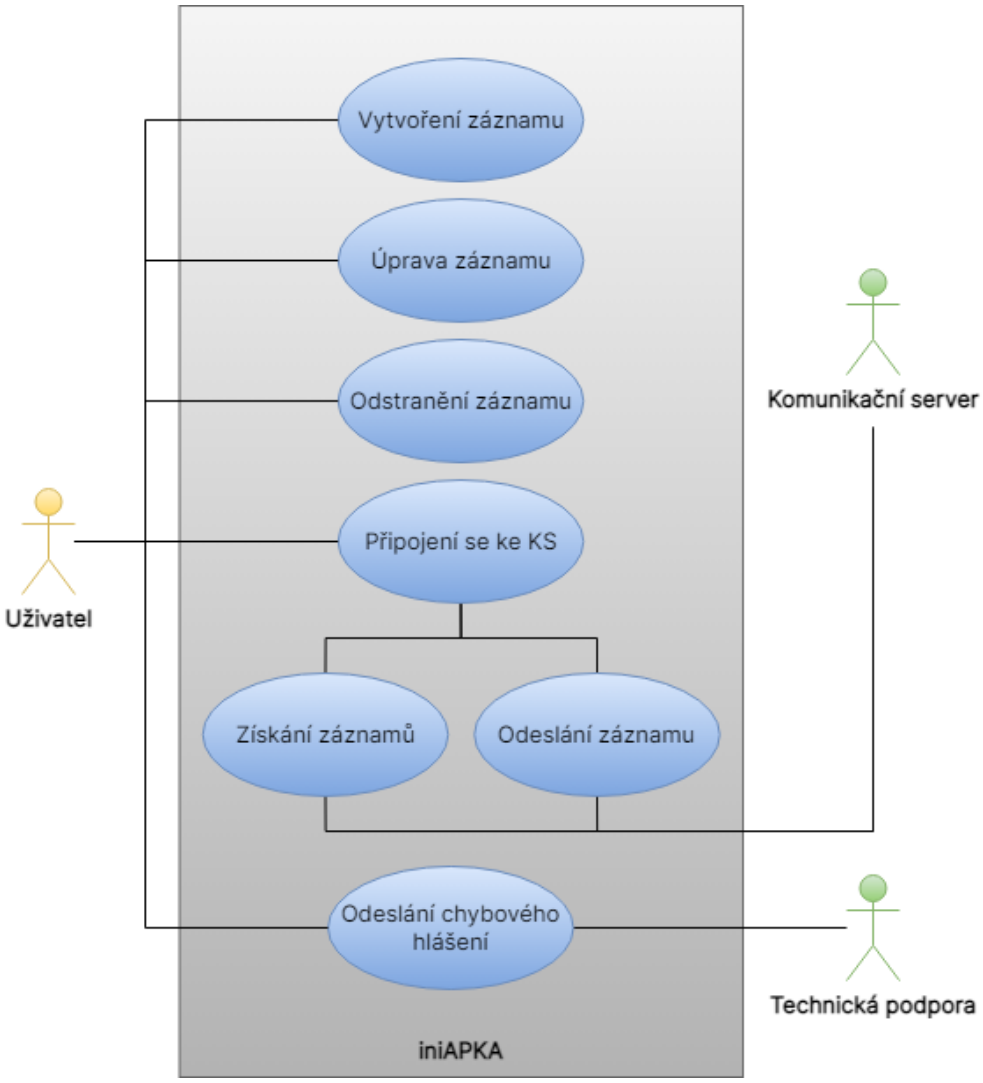
4.3 Use Case Diagram

Use Case Diagram (česky diagram případů užití) znázorňuje chování systému z uživatelského pohledu. Nezobrazuje přímo implementaci dané funkcionality, ale pouze to, jak se má systém chovat. Je tedy zřejmé, že toto je první krok návrhu systému.

Use Case (česky případ užití) je seznam několika akcí vedoucích k dosažení jedné funkcionality (např. registrace nového uživatele, přidání komentáře v diskusi atd.). Další ne tak důležité akce související s dosažením konkrétní funkcionality z pohledu uživatele (např. ověření ještě nepoužitého přihlašovacího jména právě registrovaného uživatele, kontrola znaků ve formuláři atd.) již znázorňovány v diagramu nejsou. V diagramu je Use Case znázorněn pomocí elipsy s názvem funkcionality uvnitř.

Actor (česky aktér) je role komunikující s případy užití. Tuto roli může mít buď uživatel, nebo externí systém (např. administrátor, server atd.). Aktivní aktér zahajuje dané chování – případu užití systému (např. registrace). Pasivní aktér je ten, který je chováním – případem užití systému iniciován (např. server pro posílání notifikací je iniciován případem užití Poslat notifikaci). Aktivní aktér se do diagramu přidává do levé části a pasivní do pravé části. (Seidl 2015)

Obrázek 6 byl vytvořen pomocí informací uvedených výše. Uživatel může v iniAPCE provádět následující akce: vytvořit, upravit a odstranit záznam, připojit se ke KS a pomocí něj odeslat záznamy nebo získat záznamy z INI SW a odeslat chybové hlášení technické podpoře.



Obrázek 6: Use Case Diagram iniAPKY
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

4.4 Shrnutí požadavků a výběr vhodné technologie

Mobilní aplikace, jejíž vytvoření je cílem této diplomové práce, by dle požadavků společnosti INISOFT s.r.o. měla být primárně určena pro operační systém Android s možným budoucím rozšířením pro operační systém iOS. Měla by být také schopna pracovat offline. Pro udržitelnost a budoucí úpravy by měla být aplikace napsána v programovacím jazyce C#. Zvolený framework by měl podporovat komunikaci prostřednictvím gRPC.

Po rešerši a nastudování všech dostupných možností pro vývoj mobilních aplikací, bylo rozhodnuto, že aplikace bude vyvíjena multiplatformně ve frameworku .NET MAUI v programovacím jazyce C#.

5 Vývoj iniAPKY – teoretická část

Tato kapitola obsahuje rešerši potřebných informací pro vývoj mobilní aplikace v .NET MAUI.

5.1 Životní cyklus softwaru

Životní cyklus softwaru se dá představit jako jednotlivé fáze obsahující předem dané procesy, díky kterým je vývoj mnohem kvalitnější. Tyto fáze jsou:

- **zahájení:** seznámení se s projektem, zvážení času, nákladů, výnosů aj.;
- **sběr požadavků:** zjištění potřeb (požadavky se dají sbírat různými metodami např.: průzkumy, dotazníky, myšlenkovými mapami atd.);
- **analýza:** každý požadavek je rozebrán do detailů (vzniká specifikace softwarových požadavků – SRS a specifikace funkčních požadavků – FRS);
- **návrh:** SRS je převeden na specifikaci návrhu, která obsahuje veškeré technické detaily (např. použitou technologii, bezpečnost aj.);
- **vývoj:** kódování aplikace;
- **testování:** kontrola splnění všech požadavků a funkčnosti aplikace;
- **nasazení do produkčního prostředí:** nastává po úspěšném otestování aplikace, kdy je aplikace zpřístupněna skutečným uživatelům;
- **provoz a údržba:** interakce s uživateli za účelem oprav chyb, menších změn aj.

O přechodu do další fáze rozhoduje splnění výstupních kritérií aktuální fáze. Tyto výstupní kritéria jsou obvykle vstupními kritérii fáze následující. (Rose 2022)

5.1.1 Modely životního cyklu softwaru

Vzájemné vazby mezi fázemi životního cyklu softwaru popisuje model životního cyklu softwaru. Modelů existuje více a liší se v používané metodice. Každý model má své výhody i nevýhody a hodí se pro konkrétní situaci. Níže jsou uvedeny příklady modelů, které se používají nejčastěji.

Vodopádový model

Nejstarším a nejznámějším modelem pro vývoj softwaru je vodopádový model. Jedná se o sekvenci po sobě jdoucích fází vývoje (směrem dolů, proto se označuje jako vodopádový). To znamená, že jakákoli fáze vývoje začíná až po tom, co je dokončena předchozí fáze, bez jakékoli možnosti zpětného návratu. Je tedy velmi důležité, aby byla první fáze zpracována co nejlépe, a je na ni kladen velký důraz. Protože se ne moc dobře přizpůsobuje měnícím se podmínkám, je vodopádový model vhodný pro projekty, které je možné předem jasně definovat a naplánovat.

Výhodou tohoto modelu je možnost jeho přesného naplánování (ohledně zdrojů, času atd.), rychlé odhalení chyb nebo nepřesností díky specifikovaným výstupům každé fáze, a jeho jednoduchost, která se hodí např. při komunikaci se zákazníky.

Naopak nevýhodou je již zmíněná nemožnost návratu do předešlých fází, obtížnost a vynaložení vyšších nákladů pro zakomponování jakékoli změny.

Spirálový model

Tento typ modelu kombinuje vlastnosti vodopádového modelu s prototypovým modelem[6]. Využívá se u větších, dražších a komplikovanějších projektů. Prochází postupně téměř stejnými fázemi jako model vodopádový, s rozdílem toho, že je rozdělen na menší části (cykly spirály). Spirála je rozdělena do čtyř kvadrantů: analýza, hodnocení, vývoj a plánování.

Výhodou oproti vodopádovému modelu je větší pravděpodobnost odhalení chyby či nedostatku, což znamená, že odhad práce bude mnohem realističtější, a také obsahuje část, ve které jsou zohledněna rizika.

Čas a vynaložené náklady pro dosažení konečného produktu patří mezi nevýhody tohoto modelu. Omezující je také podmínka určité znalosti spojené s vyhodnocováním rizik.

---

[6] Vytváření neúplných verzí softwaru, tzv. prototypů, z důvodu minimalizace rozdílu představ mezi zákazníkem a vývojovým týmem

Modely agilního vývoje

Modely agilního vývoje se dokáží rychle přizpůsobit nastalým změnám v rámci jednoho cyklu. Dají se využít pro libovolný projekt, avšak podmínkou je aktivní spolupráce zákazníka s vývojovým týmem. Model agilního vývoje je vhodný také v případě, že zákazník potřebuje mít rychle vyhotovené některé funkční požadavky na daný software, ale další požadavky nemá zákazník zatím úplně promyšlené. Mezi příklady modelů agilního vývoje patří Scrum, Extrémní programování (XP), Vývoj řízený vlastnostmi nebo testy atd.

V případě metodiky Scrum se týmy každý den schází a pracují v tzv. sprintech. Sprint je dvou až čtyřtýdenní období, ve kterém jsou obvykle splněny zadané úkoly. Na správné fungování a cestu k cíli dohlíží tzv. Scrum Master. Výhodou je vytvoření kvalitního softwaru v co nejkratším čase a spokojení zákazníci. Ti totiž mají stále pocit, že se něco děje, díky častým konzultacím. Mezi nevýhody je možné zařadit neschopnost zákazníka mnohdy přesně definovat konkrétní požadavky nebo také potřeba speciálních dovedností pro vedení projektu. (Half 2023)

5.2 Logické architektury

Způsob, jakým je aplikace z objektů poskládána, udává logická architektura. Díky ní, je kód přehlednější a lépe se udržuje. Existují čtyři základní typy logických architektur: monolitická, dvouvrstvá, třívrstvá a vícevrstvá.

V monolitické architektuře aplikace funguje vše v jedné vrstvě (skupině objektů). Často jsou takto psány menší aplikace. Tento typ architektury je označován za antipattern a není dobré ho používat.

Pro dvouvrstvou architekturu je typické to, že nereprezentuje data uživateli. Obsahuje dvě vrstvy: datovou a aplikační. Je využívána často pro vývoj API serverů nebo služeb obecně. Tyto služby komunikují často s další aplikací, od které přijímá požadavky a vrací jí odpovědi.

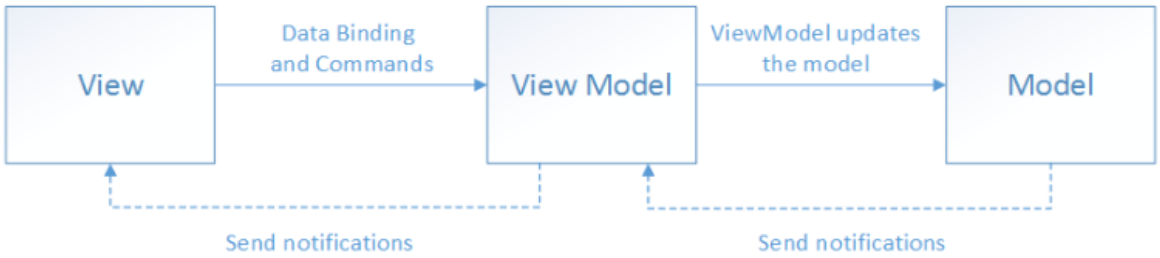
V třívrstvé architektuře (typ vícevrstvé architektury) je již zprostředkovávané zobrazování dat uživateli. Tři vrstvy aplikace tedy jsou: prezenční, aplikační a datová. Důvodem tohoto rozdělení je nezávislost těchto vrstev na sobě. Prezenční vrstva je

část, která je viditelná pro uživatele, aplikační vrstva obsahuje logiku samotné aplikace a datová vrstva zajišťuje práci s daty. Nejznámější třívrstvá architektura je MVC (Models, Views a Controllers), dále pak třeba MVVM (Models, Views a ViewModels) nebo MVP (Models, Views a Presenters). (Raj et al. 2017)

5.2.1 MVVM

Pro vývoj aplikace v .NET MAUI se nejčastěji využívá architektonického vzoru MVVM.

MVVM je architektonický vzor, který kód aplikace rozděluje dle funkčnosti do tří vrstev – Model (model), View (pohled) a View Model (model pohledu). Díky těmto vrstvám je business logika oddělena od uživatelského rozhraní. Výhoda použití vzoru je taková, že se aplikace lépe udržuje, testuje a dále rozvíjí. Na následujícím obrázku je schéma znázorňující vztahy mezi jednotlivými komponentami.



Obrázek 7: Schéma architektonického vzoru MVVM
Zdroj: Stonis 2022

Ve View je definováno to, jak vypadají a jak jsou rozloženy jednotlivé prvky uživatelského rozhraní. Soubor pro View je složen ze dvou částí: UI (uživatelské rozhraní, přípona *.xaml*) a code-behind (přípona *.xaml.cs*). UI se píše ve značkovacím jazyce XAML (eXtensible Application Markup Language) a code-behind v programovacím jazyce C#. V části pro code-behind není doporučováno implementovat business logiku. Naopak je tato část určena pro implementaci vizuálního chování (např. animace), které nelze implementovat v UI části. Akce, která má být vyvolána např. po zmáčknutí tlačítka, výběru položky nebo naplnění dat do zobrazovacích prvků (např. labelů), je definována v příslušném ViewModelu a prováděna pomocí tzv. data-bindingu (navázání dat) k vlastnostem daného UI prvku.

Model zapouzdřuje data, se kterými aplikace pracuje. Příkladem objektu modelu může být objekt pro přenos dat (DTO), proxy objekt aj.

ViewModel je takovým prostředníkem, který propojuje obě výše zmíněné vrstvy View a Model, a stará se o změnu jejich stavů. Obsahuje vlastnosti a funkce, na které jsou navázány prvky View (např. co se má stát po zmáčknutí tlačítka). Pro to, aby aplikace uživateli připadala plynulejší a výkonnější, je nutné udržovat vlákno, starající se o vykreslování uživatelského rozhraní a o interakci s uživatelem, neblokované. To se dá docílit tím, že ve ViewModelu budou definovány asynchronní metody a asynchronní oznamování změn View. Dále ViewModel koordinuje interakce mezi View a všemi potřebnými třídami Modelu. K jednomu ViewModelu může být navázáno více Modelů, jedná se tedy o vztah 1:N. Pokud dojde ke změně vlastnosti, může ViewModel provolat View pomocí události *PropertyChanged* (musí k tomu třída ViewModelu implementovat rozhraní *INotifyPropertyChanged*). Pro kolekci dat je určena událost *ObservableCollection<T>* (s implementací rozhraní *INotifyCollectionChanged*). Pro vykonávání příkazů se využívá vlastnosti *Command* (s implementací rozhraní *ICommand)*.

Pro zjednodušení práce s MVVM vzorem bylo vyvinuto mnoho knihoven. Všechny obsahují stejné podstatné funkce. V .NET MAUI jsou nejčastěji využívány následující: .NET Community MVVM Toolkit, ReactiveUI, Prism Library atd. (Stonis 2022)

5.3 Struktura .NET MAUI projektu

Pro vytvoření aplikace v .NET MAUI je potřeba mít stažené a nainstalované Visual Studio 2022 (verzi 17.3.2 nebo vyšší, u Visual Studia pro Mac verzi 17 nebo vyšší) společně s frameworkem .NET MAUI (rozšíření Visual Studia). Po vytvoření projektu v .NET MAUI je nutné stáhnout i ostatní NuGet balíčky, aby bylo vše funkční. Ke zjištění, jaké NuGet balíčky mají být nainstalovány, slouží část s názvem *Dependencies*. Tato část obsahuje veškeré závislosti, které jsou potřeba pro vytvoření aplikace pro daný operační systém.

Složka s názvem *Properties* obsahuje soubor *launchSettings.json*, který obsahuje nastavení profilu.

Platforms je složka, která v sobě zahrnuje další složky s názvy: *Android, iOS, MacCatalyst, Tizen* a *Windows.* Ty obsahují konkrétní spouštěcí soubory pro každou platformu. Pro Android je spouštěcí soubor pojmenován jako *MainApplication.cs*, pro iOS a Mac OS je to *Program.cs*, pro Tizen *Main.cs* a pro Windows *App.xaml.cs*. Dále tyto podsložky obsahují specifické a potřebné soubory potřebné pro sestavení aplikace.

Ve složce *Resources* jsou složky s názvy dle toho, co obsahují, např. obrázky (*Images*), fonty (*Fonts*), ikona aplikace (*AppIcon*), zobrazovaný obrázek během načítání aplikace (*Splash*), styly (*Styles*) atd.

Soubor s názvem *MauiProgram.cs* spouští celou aplikaci a dají se v něm nakonfigurovat vlastnosti aplikace. Odkazuje na třídu *App*.

Soubor *App* obsahuje dva soubory s názvy *App.xaml* a *App.xaml.cs*. Ve Visual Studiu v Průzkumníku řešení se dá najít *App.xaml.cs* je jako „podsoubor" souboru *App.xaml*. *App.xaml* definuje defaultní styly, barvy nebo šablony nastavením zdrojových XAML souborů a *App.xaml.cs* obsahuje kód pro interakci s *App.xaml* např. inicializuje třídu *AppShell*.

Pod označením *AppShell* jsou opět zahrnuty dva soubory: *AppShell.xaml* a *AppShell.xaml.cs*. V *AppShell.xaml* jsou rozvrženy jednotlivé položky menu.

Pomocí *MainPage.xaml* a *MainPage.xaml.cs* je zobrazována úvodní stránka. *MainPage.xaml* vykresluje uživatelské rozhraní např. tlačítka, popisky atd. *MainPage.xaml.cs* je označován jako code-behind a obsahuje kód např. pro vizuální efekty.

Pokud je takto vygenerovaný „prázdný" projekt pro vytvoření aplikace v .NET MAUI spuštěn, objeví se ukázková aplikace od vývojářů Microsoftu, která počítá stisknutí tlačítka. (George a Britch 2023)

5.4 Vývojářské poznámky

V této kapitole je zmíněno pár důležitých poznámek, které je třeba znát před vývojem aplikace v .NET MAUI.

5.4.1 Dependency injection

Dependency injection, neboli injektáž závislostí, je proces získávání závislostí. Daná třída získá závislosti za běhu aplikace jejich předáním od kontejneru pro vkládání závislostí. Tento kontejner se stará o vytváření instancí objektů rozhraní a následné vkládání do jiných tříd. Výhodou je, že odpadá starost neustále vytvářet instance všech závislostí ve třídách, kde jsou potřeba, a spravovat jejich životní cykly.

V .NET MAUI existuje mnoho kontejnerů pro vkládání závislostí, nejčastěji se využívá kontejner s názvem *Microsoft.Extensions.DependencyInjection*.

Pro použití injektáže závislostí v projektu je potřeba registrovat v kontejneru dané rozhraní a jeho konkrétní implementaci. Tuto registraci závislostí lze provést v souboru s názvem MauiProgram.cs. (Stonis 2022)

5.4.2 NuGet balíčky

NuGet balíček je soubor ve formátu ZIP s příponou .nupkg, obsahující zkompilovaný kód, ostatní související soubory a manifest, ve kterém jsou uloženy informace jako např. číslo verze daného NuGet balíčku atd. Je to jednoduchý způsob sdílení již implementovaných funkcí mezi vývojáři. NuGet balíčky mohou být veřejné nebo soukromé (např. v rámci konkrétní společnosti nebo pracovní skupiny). Vývojářovi, který chce využívat funkcí NuGet balíčku ve svém projektu, stačí pouze stažení a přidání tohoto NuGet balíčku do projektu. (Douglas 2022)

5.4.3 Asynchronní metody

Asynchronní metoda je taková metoda, která ve své deklaraci obsahuje modifikátor async, její návratová hodnota je typu Task, Task<T> (T označuje datový typ) nebo void. To, jestli se jedná o task (úlohu), lze zjistit tak, že synchronní kód je obalen kódem Task.Run(() => …).

Obvykle se na provedení kódu asynchronní metody čeká pomocí klíčového slova await. Pokud se v kódu narazí na příkaz s awaitem, vyčká se na provedení daného kódu, jehož část je uvnitř příkazu Task.Run(() => …). Jeho vykonávání se provádí na

pomocném vlákně. Až teprve co je tento asynchronní kód dokončen, pokračuje se v původním kódu dále. I když obyčejná metoda (tím je myšlena taková metoda, která ve svém těle neobsahuje Task.Run(() => ...)) volá pouze jinou metodu, která je asynchronní, stává se automaticky také asynchronní.

Důležité je, že hlavní vlákno není ničím blokováno v průběhu provádění asynchronní metody, tudíž může zpracovávat jakékoli jiné události.

Implementace asynchronních metod je vhodná na místech, na kterých se provádí např. náročný výpočet, návrat odpovědi od serveru atd. (Wagner 2023)

5.5 gRPC

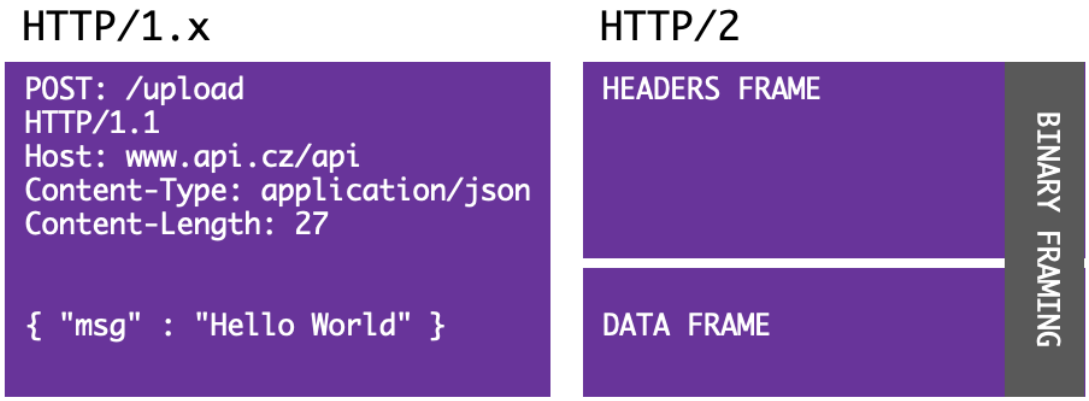
Zkratka gRPC znamená Google Remote Procedure Call. Jedná se o službu, která byla navržena společností Google v roce 2015. (Holec 2022a) Je založena na vzdáleném volání procedur. Využívá se především pro vývoj rychlých mikroslužeb pro synchronní komunikaci v reálném čase za využití protokolu HTTP/2 a binární serializace. Díky tomu je služba velmi rychlá a efektivní. Další výhodou je, že je platformně nezávislá. Jedná se o časem prověřenou technologii, kterou využívají služby jako Spotify nebo Netflix.

Plnou podporu získala služba gRPC v .NET Core 3.0 a ve verzi .NET 6 byl vylepšen výkon. Takže v porovnání s konkurencí gRPC služby dosahují nejlepší výkon právě v .NETu. (Holec 2022a)

5.5.1 Komunikace

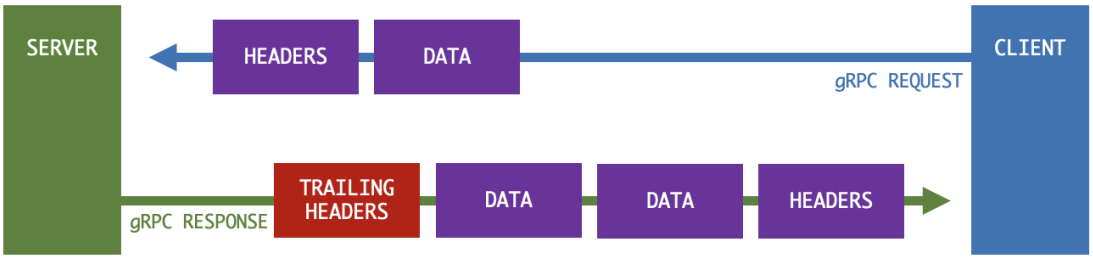
Základními pilíři jsou tzv. framing a využití streamování ze serveru na klienta, z klienta na server nebo obousměrně (Holec 2022f) a bez handshaku jako je tomu např. u HTTP/1.1 (REST) služeb, od kterých se dále liší tím, že jsou orientovány na tzv. volání procedur (a ne datově). (Holec 2022a)

Přenosový protokol HTTP/2 umí efektivně využít TCP spojení a je postaven na binárním framingu, který rozděluje zprávy do tzv. headers a data framů. Díky tomu je komunikace efektivnější než v případě protokolu HTTP/1.x.



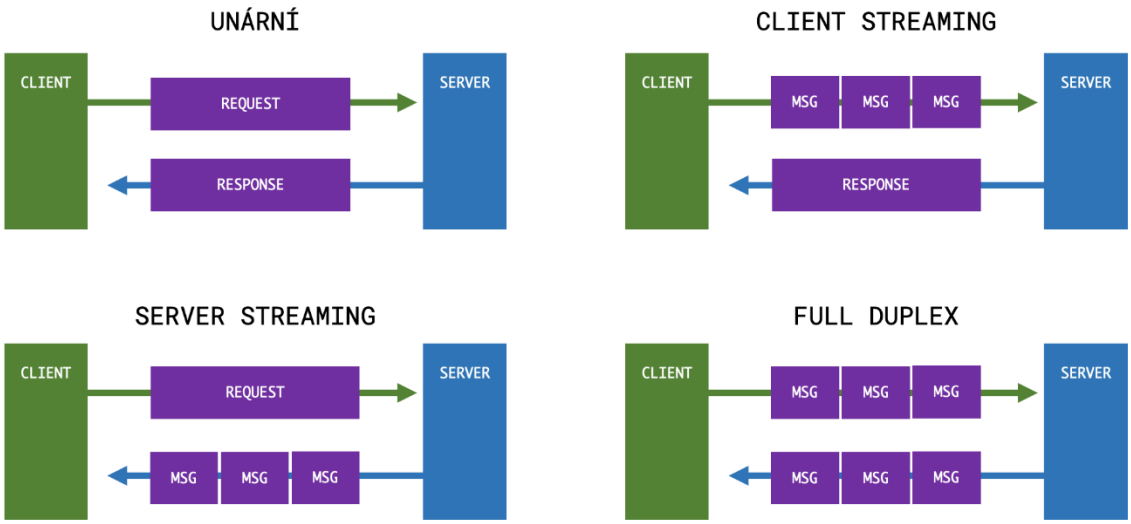
Obrázek 8: Rozdíl struktury požadavku mezi protokoly HTTP/1.x a HTTP/2
Zdroj: Holec 2022b

Po úspěšném odeslání gRPC požadavku (gRPC Request) na server následuje gRPC odpověď (gRPC Response). Na konci komunikace je dodatečný header frame (tzv. trailing header). Ten v sobě obsahuje status gRPC komunikace a běžné prohlížeče s tímto rámcem neumí dále pracovat. Je to tedy menší nevýhoda gRPC služeb a také důvod toho, že se gRPC služby obvykle používají pro komunikaci typu API-2-API.



Obrázek 9: Průběh gRPC komunikace
Zdroj: Holec 2022b

Mimo již zmíněného typu komunikace požadavek – odpověď (unární komunikace) je možné používat tzv. streaming. Existují tři typy: Client Streaming, Server Streaming a Full Duplex. V případě Client Streamingu otevírá spojení klient se serverem a odesílá na něj kontinuálně zprávy. Server na konci odešle odpověď. Pokud to funguje obráceně, jedná se o Server Streaming, a jestliže si zprávy posílá server i klient navzájem, jde o Full Duplex, neboli plně duplexní komunikaci. (Holec 2022b)



Obrázek 10: Jednotlivé typy komunikace
Zdroj: Holec 2022b

5.5.2 Protobuf

Protocol Buffers (zkráceně ProtoBuf) je mechanismus pro binární serializaci dat. Je multiplatformní a jazykově neutrální. Popisuje webovou službu a zprávy, které přijímá a odesílá. Serializace pomocí tohoto způsobu je mnohem rychlejší a úspornější než v případě např. JSON nebo XML. Díky tomuto protokolu je možné využít velkého množství nástrojů, které jsou schopné vygenerovat kód pro klienta i server v různých programovacích jazycích. (Holec 2022a) Soubory s tímto protokolem mají příponu .proto.

```
syntax = "proto3";

message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
  repeated Product products = 4;
}
```

Obrázek 11: Ukázka definice zprávy pomocí protokolu ProtoBuf
Zdroj: Holec 2022d

Na obrázku výše je ukázka definice zprávy pomocí zmiňovaného protokolu. Na prvním řádku je specifikovaná verze syntaxe (aktuální verze ProtoBufu je verze 3). Zpráva je složena ze tří vlastností. Každá z nich obsahuje datový typ, název

a pořadí. Právě pořadí je klíčové při serializaci dat a každý prvek musí mít své unikátní číslo. Zprávu můžou tvořit samostatné vlastnosti (na obrázku výše např. query) nebo kolekce (např. products). Kolekce se označují klíčovým slovem – repeated. Kromě těchto primitivních datových typů může odkazovat na další Protobuf kontrakty.

V .NETu se o kompilaci Protobuf souborů do C# tříd stará ProtoBuf Compiler. Navrhuje i vhodné datové typy pro převádění mezi proto a CLR typem (např.: string → string, sint32 → int, sfixed32 → uint, bytes → ByteString atd.). Některé datové typy ProtoBuf nemá (např.: DateTime, decimal aj.). Z nejčastěji používaných datových typů (tzv. well-known types) vytvořila společnost Google NuGet balíček, který lze zapojit do projektu. (Holec 2022d)

5.5.3 Návrh gRPC služeb

Službu gRPC se dá, kromě formátu Protobuf, navrhnout i pouze pomocí C# a .NET Frameworku. (Holec 2022f) Vývoj je pak snazší, neboť se mezi projekty sdílí jednotlivé kontrakty prostřednictvím NuGet balíčků.

Návrh gRPC služeb je složený z návrhu služeb a procedur. Každá procedura má svou vstupní a výstupní zprávu.

```
service1
    procedure1 (requestMessage + responseMessage)
    procedure2 (requestMessage + responseMessage)
service2
    procedureX (requestMessage + responseMessage)
```

Obrázek 12: Ukázka struktury gRPC služby skládající se ze služeb a procedur
Zdroj: Holec 2022b

Výsledná podoba gRPC služby je ve formě vygenerovaného C# kódu, skládajícího se ze tříd (služby), metod (procedury) a tříd pro zprávy (zprávy). (Holec 2022c)

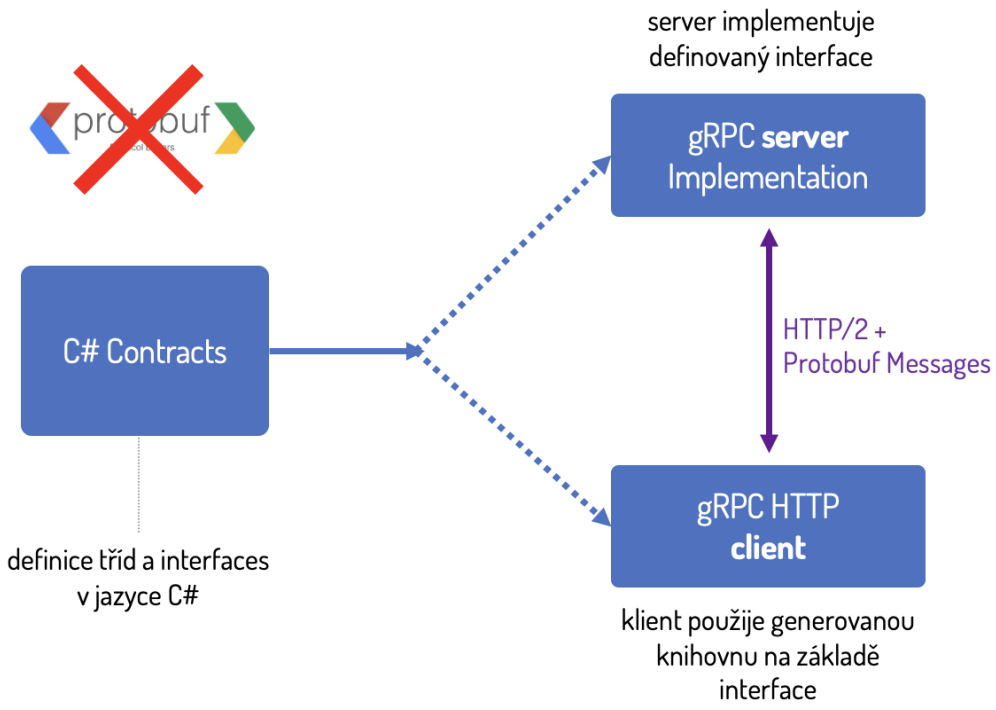
5.5.4 Metody vývoje gRPC služeb

Existují dvě možnosti, jakým mohou být gRPC služby vyvíjeny.

První metoda se označuje jako tradiční, nebo také Contract-First. (Holec 2022e) V tomto přístupu se nejdříve definuje Protobuf zpráva, a až poté probíhá kompilace do C# kódu. (Holec 2022d)

Druhou možností je Code-First, kdy se rovnou píše C# kód.

Obě tyto metody mají odlišný NuGet balíček a nastavení. (Holec 2022e)



Obrázek 13: Schéma přístupu Code-First
Zdroj: Holec 2022b

5.6 SQLite

SQLite je relační databáze, tzn. že je založena na tabulkách, a udává vztahy mezi těmito tabulkami nebo mezi entitami v dané tabulce. Jedna tabulka ukládá data o položkách jednoho typu (např. o uživatelích, zápisech odpadu atd.). Položka je uložena v daném řádku a jednotlivé sloupce představují atributy (vlastnosti) položek (např. uživatelské jméno, kód odpadu apod.). SQLite databáze není typovaná, což znamená, že není nutné udávat datové typy atributů.

Spíše, než označení databáze se uvádí označení RDBMS (Relation DataBase Management System) – systém řízení báze dat, a to z důvodu toho, že SQLite nabízí

mnohem více funkcí než obyčejné ukládání dat do souborů. Jedná se tedy spíše o nástroj, který nabízí komunikaci s databází jazykem SQL, zabezpečení nebo optimalizaci výkonu, řešení situace současného zápisu nebo editace stejné položky více uživateli ve stejném čase, spojení několika dotazů do transakcí atd.

Jak je z názvu patrné, SQLite je odlehčená SQL databáze. Takže SQLite oproti klasické SQL databázi neobsahuje např. uživatelské oprávnění, konfiguraci aj. SQLite je dobré využít v případě vývoje středních nebo menších webových stránek, menších desktopových aplikací nebo mobilních aplikací. Pro větší projekty se využívají velké databáze jako např. MySQL nebo PostgreSQL.

5.6.1 ACID

ACID je akronym slov Atomicity, Consistency, Isolation a Durability. Jedná se o vlastnosti výše zmiňovaného RDBMS.

Atomicity neboli nedělitelnost, znamená, že se operace v transakci provedou celé jako jedna (nedělitelná) operace. Když se nepovede provést část operace, tak se databáze vrátí do stavu před prováděním této operace a žádné části transakce nebudou uloženy. Je to ochrana před nekonzistentním stavem databáze.

Consistency neboli validita, označuje to, že stav databáze je po dokončení dané transakce vždy konzistentní (validní).

Isolation neboli izolace, operace jsou navzájem izolované a v průběhu se vzájemně nijak neovlivňují. V případě, že se shromáždí více dotazů na jeden řádek, jsou tyto dotazy prováděny postupně.

Durability neboli trvanlivost, data, která jsou při zápisu ihned ukládána i na trvanlivá úložiště. (Feiler 2015)

5.7 Certifikační autorita a digitální certifikát

O to, aby data nebyla žádným způsobem zachycena, přečtena a zneužita podvodníky (zejména např. u online nakupování nebo bankovních služeb), se starají certifikační autority ověřováním digitálních certifikátů.

5.7.1 Certifikační autorita

Certifikační autorita (zkratka CA) je společnost či organizace, která vydává digitální certifikáty třetí straně. Jejich úkolem je ověření totožnosti subjektu (např. jednotlivce, společnosti, webové stránky, e-mailové adresy atd.) a jeho navázání na kryptografický klíč pomocí digitálního certifikátu. CA je tak velkým pomocníkem při udržování bezpečnějšího internetu.

5.7.2 Digitální certifikát

Digitální certifikát je elektronický dokument, který poskytuje ověřování totožnosti subjektu, šifrování (zabezpečení komunikace přes nezabezpečené sítě) a integritu dokumentů (díky šifrování digitálním certifikátem není možné po síti posílané dokumenty žádným způsobem číst ani měnit třetí stranou).

Je nejčastěji ve formátu Standard X.509 a obsahuje informace o verzi (v závislosti na Standard X.509), sériovém čísle, platnosti, o totožnosti majitele certifikátu a certifikační autoritě, veřejný klíč majitele certifikátu, účel certifikátu a digitální podpis certifikační autority. Formátů pro ukládání a distribuci je mnoho, avšak nejčastěji se využívá formát PEM (známý též jako CER).

Certifikát má svou dobu platnosti, a to zejména kvůli bezpečnosti. Životnost certifikátu by měla být kratší než známá doba prolomení šifrovacího klíče v okamžiku vydání certifikátu.

Možností využití certifikátů je celá řada, např. identifikace firewallů, webového serveru při navazování HTTPS komunikace, autentizace uživatele atd.

5.7.3 Vydání digitálního certifikátu

Pro vydání digitálního certifikátu je potřeba, aby žadatel vygeneroval pár kryptografických klíčů (soukromý a veřejný) a vyplnil žádost o certifikát. Tato žádost je podepsána soukromým klíčem žadatele, a to kvůli ověření, že žadatel vlastní soukromý klíč k budoucímu certifikátu. Žádost obsahuje veřejný klíč a identifikaci žadatele, účel vydání certifikátu a již zmíněný podpis soukromým klíčem.

Vygenerování digitálního certifikátu může probíhat dvěma způsoby. První způsob je vytvoření certifikátu důvěryhodnou certifikační autoritou. Ta ověří žadatele, a pokud uzná, že je předložená žádost v pořádku, vystaví mu podepsaný certifikát. Ten se dá použít pro zabezpečenou komunikaci na internetu. Druhý způsob je vytvoření tzv. self-signed certifikátu. Což znamená, že veřejný klíč je podepsán vlastním privátním klíčem. Nevýhodou je, že takto vytvořený certifikát není důvěryhodný.

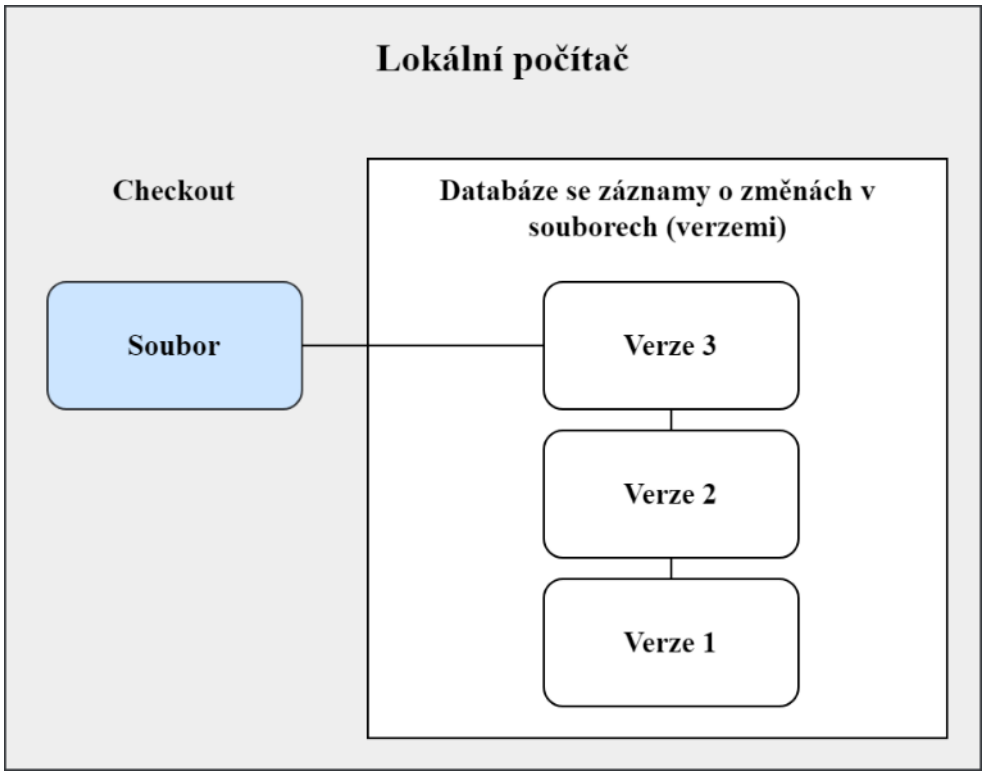
Po tom, co certifikační autorita vydá certifikát, je potřeba spojit veřejný a soukromý klíč. (Hughes 2022)

5.8 Verzování

Verzovací systém (zkratka VCS z anglického Version Control System) slouží k tomu, aby změny, provedené v kódu, byly zaznamenané, a to z důvodů např. možného návratu aktuálního kódu do předchozího stavu, porovnání změn v průběhu času, zjištění autora dané části kódu, zálohy atd. Existují tři typy VCS pro správu verzí – lokální, centralizované a distribuované.

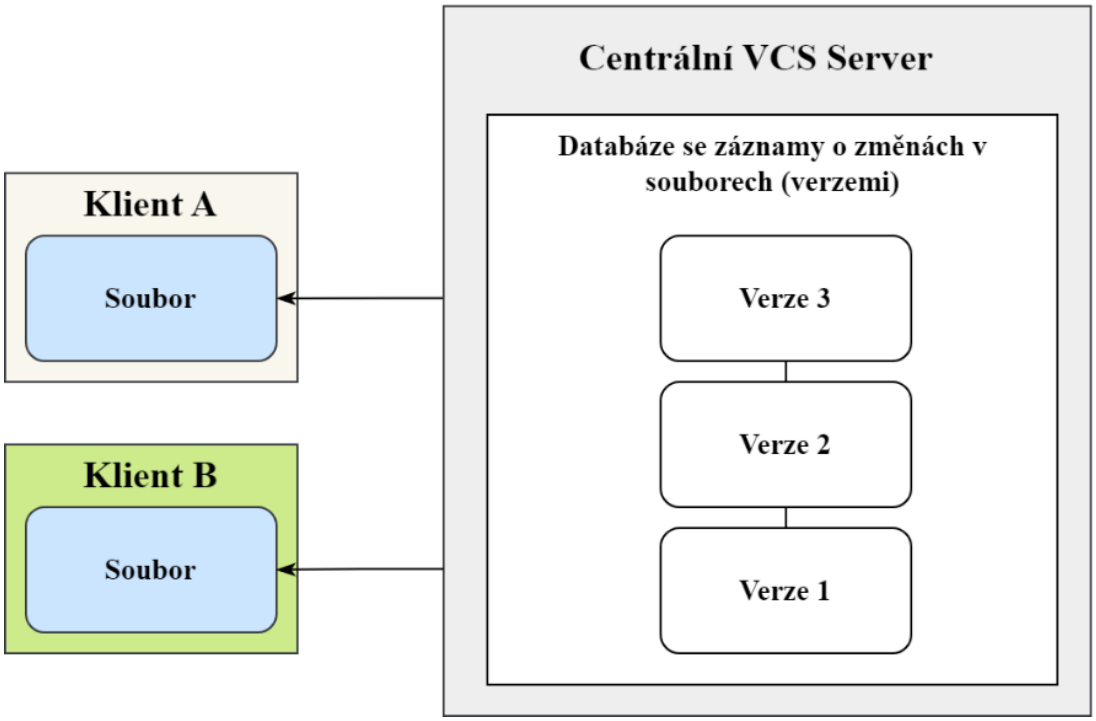
Nejméně efektivní, ale zato nejsnazší a nejrychlejší, metoda pro správu verzí je obyčejné překopírování souborů do jiných adresářů. Nevýhoda tohoto přístupu spočívá v tom, že je jednoduché zapomenout, kam tyto soubory byly překopírovány nebo na místě, kde tyto soubory mají být, nejsou, protože byly omylem uloženy na jiné místo, anebo byly překopírovány úplně jiné soubory, než měly být. Proto byl vyvinut lokální VCS (zkratka LVCS), který obsahoval databázi obsahující záznamy o změnách v jednotlivých souborech. Asi nejznámějším příkladem nástroje lokálního VCS je RCS (zkratka pro Revision Control System), který je stále distribuován a funguje na prin-

cipu uchovávání sady patchů (tj. rozdílů mezi soubory), a sečtení těchto rozdílů pak může obnovit to, jak vypadal dříve jakýkoli verzovaný soubor.



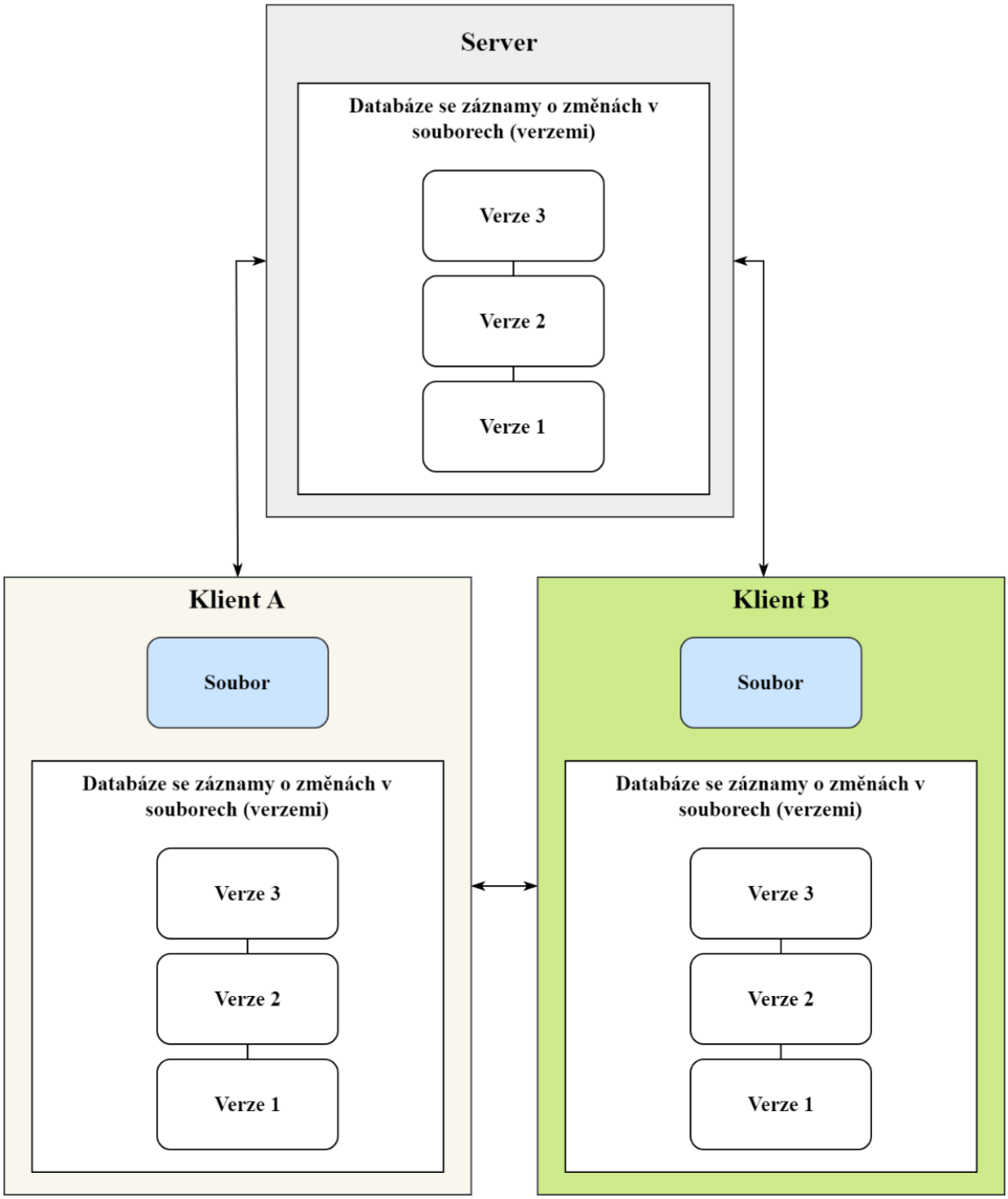
Obrázek 14: LVCS
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

Centralizovaný VCS (zkratka CVCS) řeší problém potřeby spolupráce s ostatními vývojáři na více systémech. Prakticky to vypadá tak, že na jednom serveru jsou uloženy všechny soubory (i předcházející verze těchto souborů), a k němu může přistupovat řada klientů. Spoustu let byl tento přístup standardem pro správu verzí. Velmi populárními CVCS jsou např. CVS, Subversion nebo Perforce. Nevýhoda tohoto přístupu je v tom, že pokud server, na kterém jsou uloženy soubory, vypadne, nikdo nemůže ukládat ani kontrolovat změny v souborech. Poškození úložiště serveru znamená ztrátu všech verzí uložených souborů. To samé platí samozřejmě i pro LVCS.



Obrázek 15: CVCS
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

Posledním typem je distribuovaný verzovací systém (zkratka DVCS). Funguje podobně jako CVCS až na to, že klienti mají u sebe kompletní historii změn, ke které se mohou v případě potřeby vrátit. V případě poškození serveru nejsou všechny soubory ztraceny. Příkladem tohoto typu VCS jsou Git, Mercurial, Bazaar nebo Darcs.



Obrázek 16: DVCS
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

5.8.1 Git

Git je distribuovaný verzovací systém, který se liší od jiných VCS způsobem, jakým přemýšlí o svých datech. Neukládá je jako seznam změn založených na souborech, jako to dělají ostatní VCS, ale jako sadu snímků, tzv. snapshotů, miniaturního souborového systému. V okamžiku, kdy vývojář chce uložit svou práci, Git udělá jakýsi

obrázek toho, jak soubory v daném okamžiku vypadají, a uloží na něj odkaz. Pokud není soubor pozměněn, Git uloží pouze odkaz na předchozí identický soubor, který má již uložený.

Git má tři hlavní stavy, ve kterých mohou být soubory projektu: zapsaný (committed), změněný (modified) a připravený k zapsání (staged). Stav „zapsaný" znamená, že jsou data uložena v lokální databázi, stav „změněný" informuje o tom, že pozměněný soubor ještě není zapsán v žádném snapshotu, a poslední stav „připravený k zapsání" znamená, že změněný soubor v aktuální verzi má být zapsán do dalšího snímku.

Samotný Git je rozdělen také do tří částí: adresář systému Git (Git repository), oblast, ve které jsou připravené změny (staging area) a pracovní adresář (working directory). V adresáři systému Git jsou uložena metadata, databáze objektů a samotná struktura a soubory daného projektu. Oblast připravovaných změn je soubor, ve kterém jsou informace o tom, co bude obsahovat další commit (to, co má být zapamatováno – změny v kódu). Pracovní adresář je lokální kopií jedné verze projektu.

Příkazů pro práci s Gitem je celá řada, níže je vypsáno pár základních:

- git init *název repozitáře* (vytvoří adresář .git);
- git clone *URL* (stáhne repozitář);
- git add . (změní stav „změněný" všem souborům na stav „připravený k zapsání");
- git commit -m *„zpráva"* (uloží změny souborů ve stavu „připravený k zapsání" s krátkou zprávou o daných změnách);
- git pull (stáhne všechny změny z Git repozitáře do lokálního pracovního adresáře);
- git push (odešle připravené změny z lokálního pracovního adresáře do Git repozitáře).

Nejznámějšími webovými Git repozitáři jsou GitHub a GitLab. (Chacon 2014)

5.9 Testování

Testování je nedílnou součástí vývoje softwaru. Je to proces spouštění daného programu nebo aplikace s cílem najít softwarové chyby nebo také ověření toho, že program nebo aplikace splňuje veškeré požadavky, které má splňovat. Díky testování lze např. předejít chybám, snížit náklady na vývoj nebo také zlepšit výkon. (IBM 2022)

Existuje mnoho přístupů k testování, ale podle způsobu realizace testů existují tři typy: manuální, automatizované a exploratory testy.

Manuální testování provádí sám tester ručně dle testovacích případů (tzv. testcase). Je ale časově náročné, a tak se hodí spíše pro menší projekty.

Automatizované testování probíhá na základě napsaných skriptů, které slouží k ověření předem známého scénáře. V posledních letech je tento typ testování velice populární a využívaný.

Posledním typem jsou tzv. exploratory testy. Ty jsou velmi podobné manuálnímu testování, tzn. že je zde opět potřeba tester, který bude testy provádět ručně, ale liší se v tom, že zde není žádný předpřipravený scénář. Cílem exploratory testu je popis chování daného softwaru. Často tento typ testování bývá mezi prvními kroky agilního cyklu. (Kitner 2017)

5.10 Vytvoření spustitelné aplikace

Aplikaci je možné sestavit pomocí Visual Studia. Podmínky a způsoby pro sestavení aplikace pro různé operační systémy jsou odlišné. Aplikace v .NET MAUI může být napsána pro Android 5.0, iOS 10, macOS 10.15 a Windows 10 verze 1809 a všechny vyšší verze uvedených operačních systémů.

Pro sestavení aplikace pro OS Windows postačí samotné Visual Studio. Spustitelný soubor má příponu .exe.

V případě OS Android je nutné mít fyzické zařízení s tímto OS. Druhou možností je stažení emulátoru ve Visual Studiu, který umožňuje simulaci zařízení s Androidem.

Výhodou použití emulátoru je možnost vyzkoušení aplikace na různých zařízeních s rozdílnými verzemi OS Android. Pro zrychlení výkonu emulátoru je důležité povolení hardwarové akcelerace. Na debugování a testování aplikace na fyzickém zařízení je nutné povolit vývojářské možnosti a možnost USB ladění. Aplikaci je možné distribuovat v souboru s příponou .apk (Android Package) nebo .aab (Android App Bundle). Soubor s příponou .apk je využíván k instalaci do zařízení a soubor s příponou .aab k publikování na Google Play. (Saseendran 2022)

Pro sestavení aplikace pro macOS je potřeba mít k dispozici fyzické zařízení s tímto OS, stažené Visual Studio pro macOS a Xcode. Instalační soubor pro distribuci má příponu .pkg (Teheran 2022) a soubor, který poběží bez instalace, má příponu .app.

Aplikaci pro iOS nelze sestavit bez fyzického zařízení s macOS. Lze vyvíjet přímo na tomto zařízení nebo na zařízení s OS Windows. V případě vývoje na zařízení s OS Windows je třeba se spárovat se zařízením s macOS. I v tomto případě lze stáhnout iOS emulátor. Vývojář si musí vytvořit účet na portálu Apple Developer, stáhnout si Xcode, do kterého se přihlásí pomocí tohoto účtu, a pomocí Xcode stáhnout tzv. provisioning profil. Provisioning profil obsahuje vývojářské certifikáty, unikátní identifikátory zařízení (na kterých může být aplikace spuštěna) a App ID (identifikátor aplikace). Zařízení s iOS lze připojit k zařízení s macOS pomocí USB kabelu nebo bezdrátově. Během sestavování je aplikace podepsána pomocí certifikátu. Pro distribuci iOS aplikace je vhodný soubor s příponou .ipa. (Britch 2023)

5.11 Publikace aplikace na Google Play a App Store

Pro publikaci aplikace na Google Play je třeba si založit účet vývojáře. Služba, která slouží pro publikování a správě aplikací se nazývá Play Console. Při registraci je nutné zvolit typ účtu (osobní nebo firemní), informace o vývojáři nebo společnosti (kontakt, zkušenosti, reprezentativní webové stránky, preferovaný jazyk atd.) a odsouhlasí po přečtení smluvní podmínky. Pro založení účtu je požadován jednorázový poplatek ve výši 25 $, což je v přepočtu zhruba 557 Kč (ke dni 7. února 2023, kurzy.cz). Aplikaci je možné nahrát na Google Play v souboru s příponou .aab. Nahrávání souborů s příponou .apk již není podporováno. To přináší hlavně výhodu v tom, že soubor s příponou .aab je více optimalizovaný, takže zabírá méně místa v úložišti. Podepsání aplikace probíhá vlastním vytvořeným klíčem ve formátu

.keystore nebo .jks. Je velmi důležité si tento klíč uložit, aby jím mohly být podepsány i budoucí verze aplikace.

Publikování aplikace na App Store je náročnější a dražší. I zde je potřeba vytvoření vývojářského účtu v programu s názvem Apple Developer. K tomu je nutné mít Apple ID a zaplatit poplatek 99 $, což je v přepočtu 2 205 Kč (ke dni 7. února 2023, kurzy.cz), a to pouze na rok. Navíc na rozdíl od téměř okamžitého ověření uvedených informací ve služba Play Store, trvá registrace do programu Apple Developer několik dní. Ověření probíhá prostřednictvím hovoru od Applu. Na App Store je možné nahrát soubory s příponou .ipa. Je také potřeba mít zařízení s macOS se staženým Xcodem. Na webu Apple Developer je potřeba aplikaci zaregistrovat a vytvořit si zřizovací profil (tzv. provisioning profile). Podepsanou aplikaci je možné nahrát na App Store.

Před publikováním aplikace na Google Play nebo App Store je nutné zadat mnoho informací jako např.:

- **název:** pro každý jazyk lze přidat název v daném jazyce, maximální počet znaků, které může název obsahovat je třicet;
- **bundle ID:** identifikační kód aplikace, nelze ho už nikdy změnit;
- **popis:** shrnutí obsahu a funkcí aplikace, na Google Play lze zadat stručný popis na osmdesát znaků a úplný popis na čtyři tisíce znaků, v případě App Store lze zadat podnadpis o délce třiceti znaků a tzv. promovací text, sloužící pro sdělení důležité zprávy uživatelům bez publikování nové verze aplikace;
- **další texty:** k vydání nové verze aplikace lze napsat stručný popis změn, v případě vydání nové verze na App Store je možné aktualizovat pouze již zmíněný popis aplikace;
- **kategorie:** zařazení aplikace dle jejího zaměření do kategorie pro snadnější vyhledávání;
- **štítky nebo klíčová slova:** pro větší specifičnost aplikace lze přidat štítky, kterých může být na Google Play až pět, nebo klíčová slova na App Store, kterých může být až sto;
- **zásady ochrany osobních údajů:** informace o tom, jaká data o uživatelích jsou v aplikaci shromažďována, využívána a sdílena a s kým;

- **kontaktní údaje na vývojáře nebo podporu:** jméno a příjmení, e-mailová adresa a telefon;
- **ikona:** nejlépe ve formátu 32bitového PNG s alfa kanálem (kvůli podpoře průhlednosti) s rozměry 1024 × 1024 px;
- **snímky obrazovky (screenshoty):** na Google Play nebo v App Store je vidět, jak aplikace vypadá a funguje, hodí se obrázky ve formátu JPEG nebo 24bitového PNG bez alfa kanálu s maximální velikostí 8 MB, Google Play umožňuje nahrát až osm screenshotů, a to pro každý z podporovaných typů zařízení: mobil, tablet se sedmi palci a deseti, na App Store je potřeba vložit více variant snímků obrazovky odpovídajících displejům všech podporovaných zařízení. (Think Easy 2023)

5.12 Dokumentace

Dokumentace je jednou z důležitých částí vývoje softwaru. Jedná se o text, jehož hlavním úkolem je popsat daný software, ve smyslu toho, jak se používá nebo jak funguje. Často je dokumentace začleněna do nápovědy určené pro uživatele. Může díky tomu ušetřit čas a námahu např. týmu z oddělení podpory. Dokumentace se postupně vyvíjí společně se softwarem, který popisuje. Měla by být napsána srozumitelně, jasně a měla by být také pravidelně aktualizována v závislosti na zpětné vazbě od koncových uživatelů a na vydání nových verzí.

Existují dva typy softwarové dokumentace: interní a externí. Jak z názvu těchto typů vyplývá, tak interní typ dokumentace se používá uvnitř firmy a externí mezi koncovými uživateli. Interní dokumentace se obvykle skládá z administrativní části, která obsahuje např. požadavky na daný software, informace o vývoji atd., dále vývojářskou část, ta je zaměřena na implementaci daného softwaru (popis tříd, funkcí aj.). Externí dokumentace obsahuje pokyny pro nasazení a používání daného programu. Často má tyto části: dokumentace pro koncového uživatele, kde jsou různé návody nebo výukové programy pro práci se softwarem, dokumentace pro IT oddělení, kde jsou rady pro nasazení softwar v rámci podniku, a poslední část je Just-In-Time dokumentace, která je poskytována uživatelům ve chvíli, kdy vyjde nová verze softwaru a s ní i nové funkce, které jsou určeny pro zákazníky.

Příkladů dokumentace může být celá řada: systémová dokumentace (obsahuje diagramy pro popis struktury softwaru nebo jeho technické provedení), dokumentace API (definuje volání API), README soubory (soubor, který většinou popisuje zdrojový kód, členění projektu atd.), poznámky k vydání verze (obsahuje seznam nových funkcí nebo chyb, které byly opraveny ke každé verzi programu), tutoriály (video nebo text, který vyučuje uživatele, jak mají používat daný program) atd.

Existuje také mnoho nástrojů, které dokáží usnadnit celý proces tvorby a sdílení dokumentace, např. Apiary, APIMatic, GitHub Pages, ReadMe, Stoplight nebo Swagger. (Lutkevich 2022)

6 Vývoj iniAPKY – praktická část

V této kapitole bude popsána praktická část diplomové práce, tedy vývoj iniAPKY. Celý proces vývoje je rozdělen do samostatných podkapitol.

6.1 Legislativa odpadového hospodářství

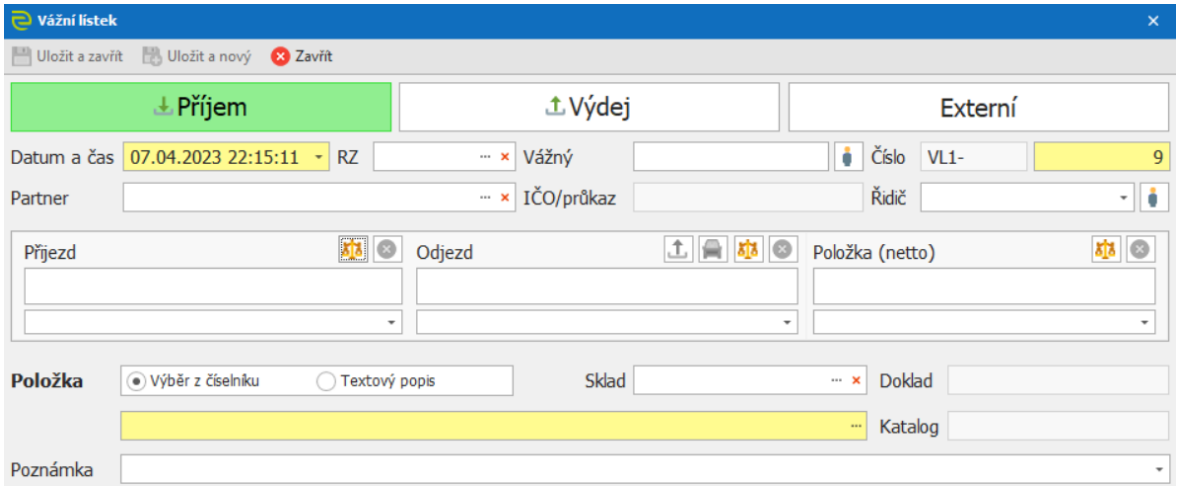
Z pravidel pro průběžnou evidenci stanovených ve vyhlášce č. 273/2021 Sb. o podrobnostech a nakládání s odpady, stejně tak ve vyhlášce č. 16/2022 Sb. o podrobnostech nakládání s některými výrobky s ukončenou životností a ve vyhlášce č. 345/2021 Sb. o podrobnostech nakládání s vozidly s ukončenou životností vyplývá, že je nutné zapisovat v každém záznamu druh odpadu (katalogové číslo), kategorii vyjadřující, zda odpad obsahuje nebezpečné látky, respektive vlastnosti, množství v tunách, údaje o předávající osobě v rozsahu jméno a příjmení, číslo občanského průkazu nebo IČO a název právnické osoby či fyzické osoby oprávněné k podnikání, místo trvalého pobytu nebo sídla provozovny společnosti a místa vzniku odpadu. U každého záznamu je také nutné uvést datum a čas převzetí či předání a u některých druhů odpadu je dokonce nutné provést identifikaci prostřednictvím fotografie. Výše uvedené hodnoty je tak nutné prostřednictvím iniAPKY zaznamenat a uložit do databáze pro následné odeslání do INI SW prostřednictvím komunikačního serveru.

6.2 Vážní lístek

Pro celou aplikaci je stěžejní práce s vážními lístky. Vážní lístek je pomůcka pro snadnější a rychlejší evidenci dat především při vážení odpadu během přijímání nebo předávání. Rychlost evidence spočívá v tom, že data ve vážních lístkách nejsou tak striktně kontrolovány, a pokud si je uživatel následně opravdu jist, že záznam o odpadu z vážního lístku chce vložit do své stávající evidence s odpady, tak data ve vážním lístku musí doplnit o další náležitosti, které vyplývají ze zákona.

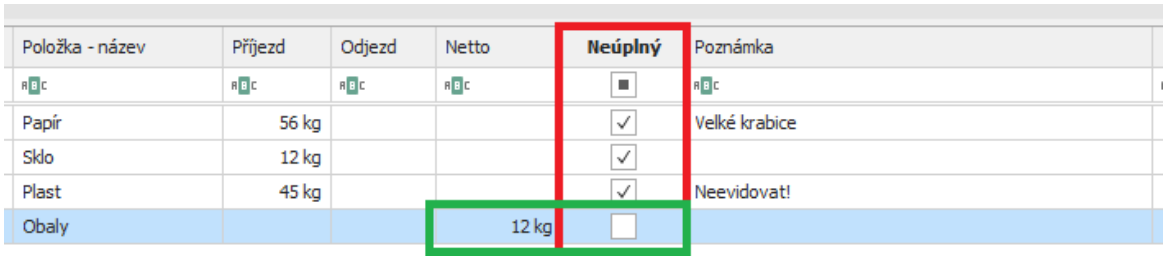
V nastavení vážních lístků jsou informace o místě vážení (např. konkrétní váha atd.) a o propojení se skladovým místem v INI SW. To, aby každé místo vážení mělo svou evidenci kvůli přehlednosti.

Samotný vážní lístek obsahuje informace o tom, jestli je odpad přijímán, vydáván nebo se jedná o externí vážení, datum a čas, registrační značku vozidla, číslo (identifikátor daného vážního lístku), partnera, IČO nebo číslo průkazu partnera, řidiče, příjezdovou/odjezdovou/čistou hmotnost, informace o položce, skladovém místě, dokladu, katalogu a lze přidat i poznámku a přílohu ve formě fotografie.



Obrázek 17: Snímek z IS ENVITY – formulář pro vytvoření vážního lístku
Zdroj: IS ENVITA

Důležité je zmínit, že v iniAPCE je možné získat a upravovat pouze neúplné vážní lístky (případně úplné vážní lístky, které ještě nebyly odeslány z iniAPKY do INI SW). To, že jsou neúplné znamená, že jim chybí údaj o čisté hmotnosti. Pokud tedy např. v INI SW obsahuje vážní lístek čistou hmotnost, v iniAPCE se tento vážní lístek nezobrazí, a tudíž ho nebude ani možné nějak upravit. Na obrázku níže je snímek z aplikace IS ENVITA, ve které je detailní přehled všech vážních lístků. V červeném rámečku je checkbox u každého vážního lístku, který informuje právě o tom, zda je vážní lístek úplný nebo neúplný (zelený rámeček potvrzuje to, že pokud vážní lístek obsahuje čistou hmotnost – Netto, tak je vážní lístek úplný).

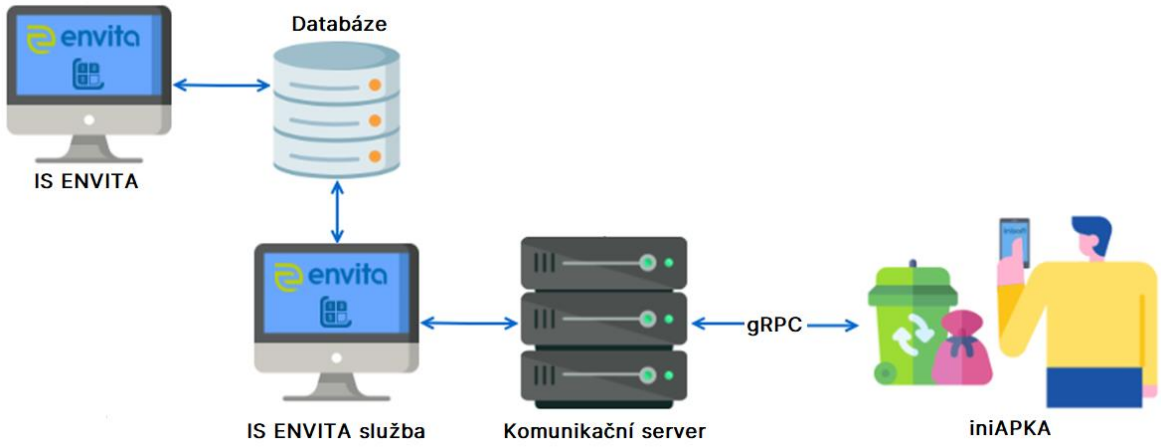


Obrázek 18: Neúplné vážní lístky v IS ENVITĚ
Zdroj: IS ENVITA

Do iniAPKY zatím není možné zapsat všechny údaje o vážním lístku, které lze zaznamenat např. v IS ENVITĚ. A to především kvůli existujícím vazbám některých údajů na číselníky, které v IS ENVITĚ má uživatel uložené.

Některé údaje vážního lístku se po odeslání z iniAPKY do INI SW mohou zahodit, a to z důvodu již zmíněných číselníků. Takovým typickým údajem, který se zahodí, pokud ještě neexistuje jeho záznam v číselníku z některého INI SW, je registrační značka (RZ).

6.3 Schéma komunikace



Obrázek 19: Schéma komunikace iniAPKY s KS a INI SW
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

Na obrázku výše je schéma komunikace iniAPKY s komunikačním serverem a INI SW. IniAPKA pomocí gRPC komunikuje s již zmíněným komunikačním serverem, díky kterému jsou data přijímána nebo odesílána iniAPKOU.

O akce spojené s databází (vytvoření, čtení, editace a mazání záznamů) se v případě IS ENVITY stará IS ENVITA služba, nebo lze data v databázi měnit přímo v IS ENVITĚ.

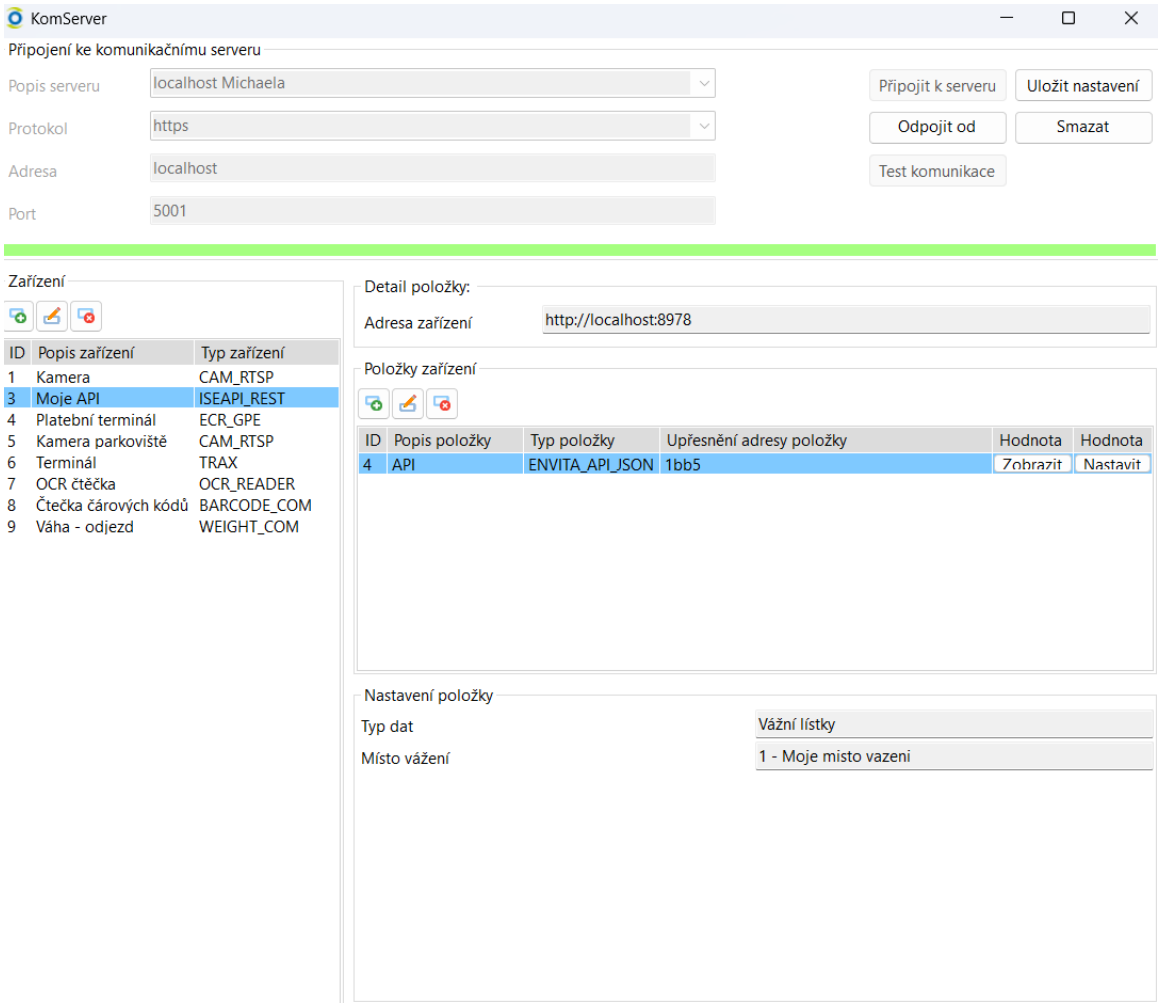
6.4 Komunikační server

Komunikační server (zkratka KS) je speciální program, který zajišťuje komunikaci s periferiemi (např. váhy, platební terminály, semafory, čtečky čipů atd.). S tímto

serverem komunikují i jiné programy společnosti INISOFT s.r.o. jako IS ENVITA a SKLAD Odpadů 8. IniAPKA s tímto komunikačním serverem komunikuje také, a to pomocí gRPC.

Pro uživatelské připojení ke KS je určena aplikace s názvem KomServer, která umožňuje uživateli nakonfigurovat periferie a zobrazit jejich stav.

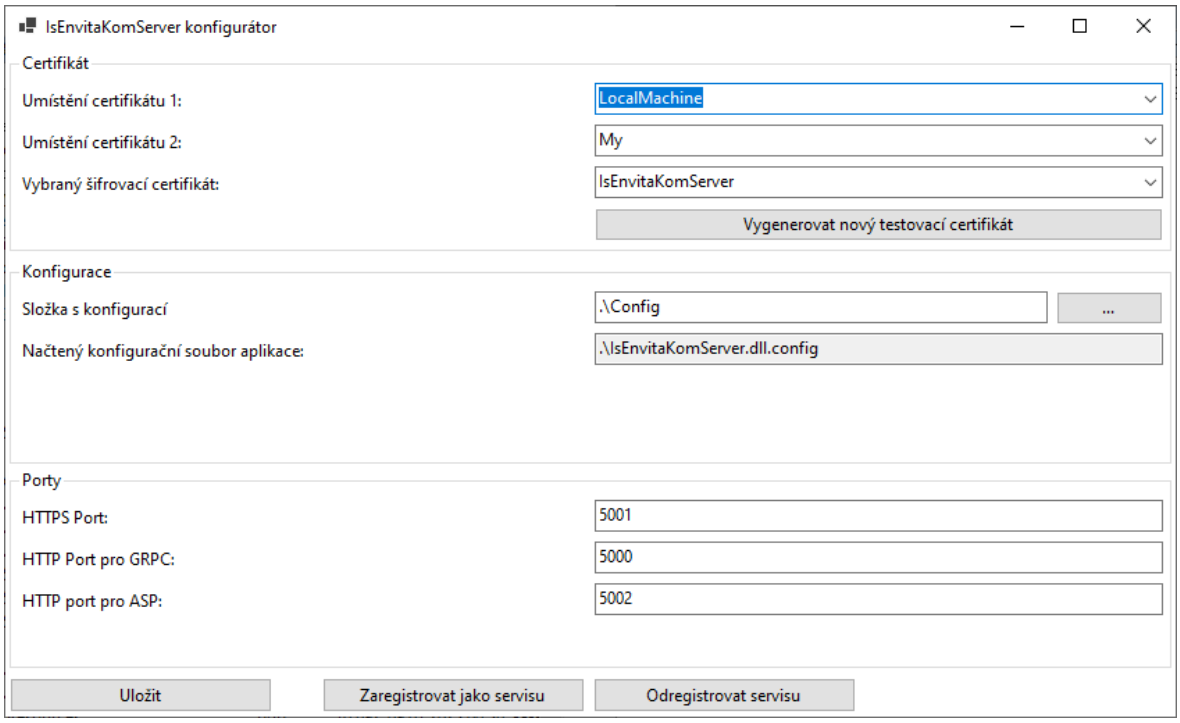
Každý KS má svůj konfigurační soubor, ve kterém jsou základní informace např. o portech, logování atd. Serverovou konfiguraci je možné upravit v aplikaci IsEnvitaKomServer konfigurátor. Uživatel si zde může zvolit umístění konfiguračního souboru, vygenerovat certifikát nebo si nastavit porty, na kterých poběží dané služby.



Obrázek 20: Snímek z aplikace KomServer (produkt společnosti INISOFT s.r.o.)
Zdroj: KomServer

Na obrázcích: Obrázek 20 a Obrázek 21 jsou snímky ze zmiňovaných aplikací. V případě aplikace KomServer se uživatel musí nejprve připojit ke komunikačnímu

serveru, a poté si může přidat potřebná zařízení. V horní části okna je nastavení připojení ke komunikačnímu serveru, vlevo jsou možnosti přidání, úpravy nebo odebrání zařízení a v pravé části je nastavení položek daného zařízení. Pro komunikaci KS s iniAPKOU byl přidán nový typ zařízení pod názvem ISEAPI_REST (pro testovací účely komunikace s iniAPKOU existoval zprvu ještě jeden typ zařízení s názvem INIAPP_GRPC).

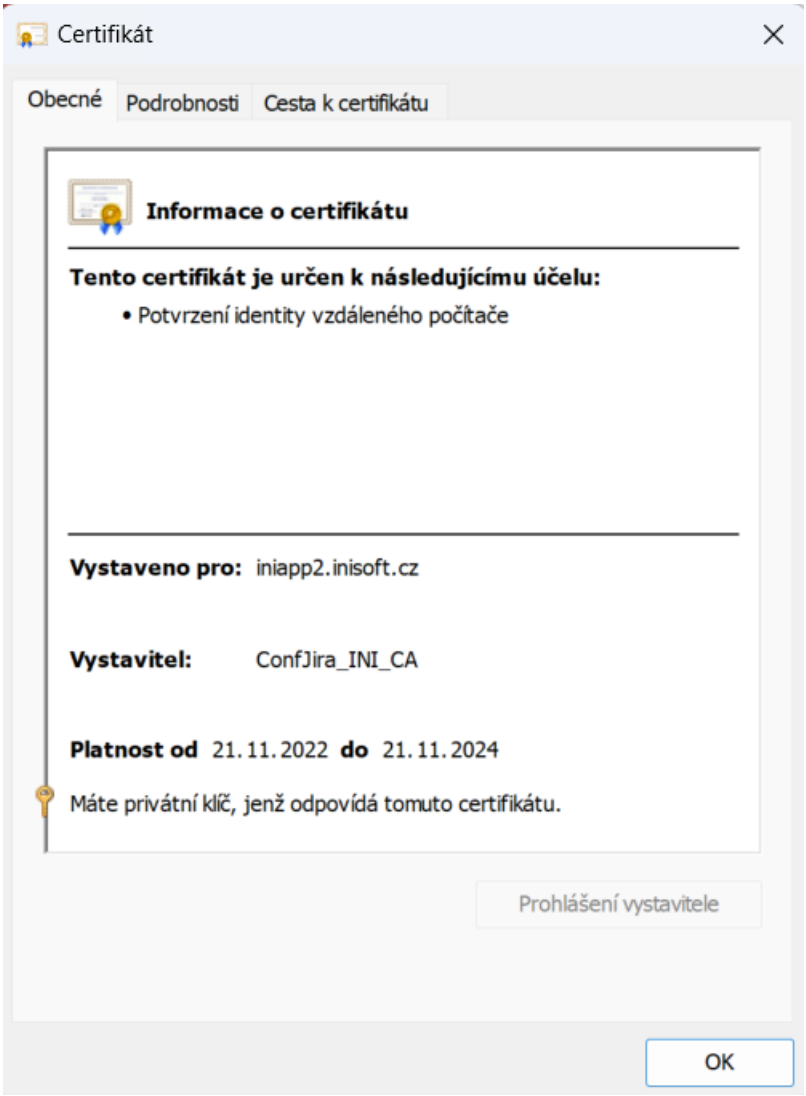


Obrázek 21: Snímek z aplikace IsEnvitaKomServer konfigurátor
Zdroj: IsEnvitaKomServer konfigurátor

6.5 Podmínky pro fungování

Jak již bylo zmíněno, aplikace funguje na mobilních zařízeních s OS Android. Dalšími prvky jsou komunikační server a router. Komunikační server je program od společnosti INISOFT s.r.o., který je nainstalován na fyzické zařízení uživatele (např. počítač atd.). Router vytváří izolovanou lokální síť, řeší DNS a pevnou adresu pro iniapp2.inisoft.cz. Tato adresa byla vytvořena speciálně pro připojení iniAPKY ke KS. Na zařízení, kde běží komunikační server, musí být zároveň nainstalován digitální certifikát vydaný veřejně známou certifikační autoritou, aby bylo možné navázat s KS i šifrovanou komunikaci.

Nicméně je ale možné aplikaci používat i bez nainstalování certifikátu do zařízení a pořizování routeru. Ovšem za podmínky, že uživateli nevadí nešifrovaná komunikace s KS.



Obrázek 22: Nainstalovaný digitální certifikát na zařízení, kde běží KS
Zdroj: Správa certifikátů a uživatelů – Windows 11

6.6 Popis fungování iniAPKY

Nejprve je nutné stáhnout a nainstalovat si iniAPKU z Google Play. Po otevření aplikace se jako první zobrazí stránka pro připojení se ke komunikačnímu serveru. Společně s adresou pro připojení je nutné zadat i přihlašovací jméno a heslo, které je prozatím libovolné (jedná se pouze o předpřípravu pro budoucí funkce aplikace). Pokud tyto údaje uživatel nevyplní, tak může provádět pouze akce, ke kterým není nutné připojení ke KS, tzn. přidávat, odstraňovat a upravovat vážní lístky a odeslat

chybové hlášení. Pokud se uživatel bude snažit provádět jakékoli akce, které připojení ke KS vyžadují, iniAPKA ho o tom informuje zobrazením dialogového okna s upozorněním o tom, že není připojen ke KS.

Pokud se uživatel ke KS úspěšně připojí, objeví se mu stránka pro výběr vážního místa a položky vybraného vážního místa. Pokud uživatel nevybere nic, aplikace se chová tak, jako když neexistuje připojení ke KS, viz předchozí odstavec. Pokud je obojí vybráno, uživatel je přesměrován na stránku Vaše záznamy.

Na téměř všech stránkách iniAPKY v levém horním rohu vlevo je menu, kde se po jeho rozkliknutí může uživatel v aplikaci přesouvat. A pokud tam není, je tam šipka směřující doleva, která vrací uživatele na předešlou stránku.

Na stránce Vaše záznamy v horní liště (toolbaru) má uživatel možnost smazat všechny lokálně uložené vážní lístky (ikona popelnice), odeslat všechny vážní lístky do INI SW (ikona šipky směřující nahoru), získat neúplné vážní lístky z INI SW (ikona šipky směřující dolů) a vytvořit nový vážní lístek (ikona plus). Pokud uživatel není připojen ke KS, může pouze vytvářet vážní lístky, editovat je a všechny je následně smazat. Pokud uživatel získá neúplné vážní lístky z INI SW, bude u takto získaných vážních lístků vlevo nahoře ikona iniAPKY a zároveň se pod výpisem dat vážního lístku zobrazí pouze tlačítko Upravit. V případě existence vážních lístků z INI SW v lokální databázi a kliknutí na ikonu pro získání vážních lístků z INI SW, budou všechny úpravy již existujících vážních lístků získaných z INI SW nahrazeny novými vážními lístky z INI SW, ale vlastní lokálně vytvořené vážní lístky nebudou nijak ovlivněny. U záznamu vážního lístku založeného v iniAPCE žádná ikona nebude, ale u tohoto záznamu má uživatel možnost ho lokálně odstranit kliknutím na tlačítko Smazat. V případě odeslání vážních lístků do INI SW se odešlou úplně všechny vážní lístky a zároveň budou vymazány z databáze iniAPKY, takže se již uživateli nebudou zobrazovat.

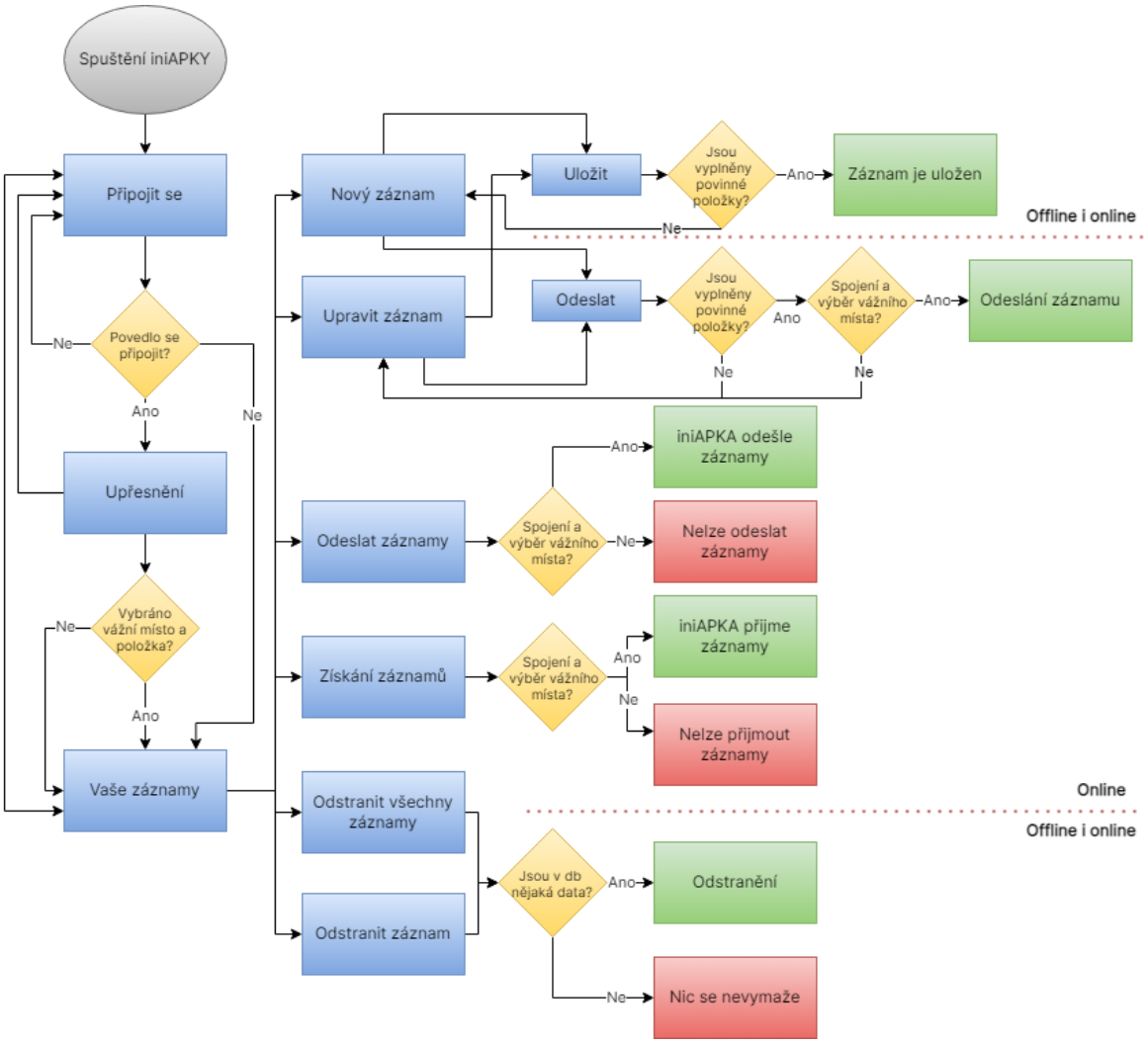
Pro vytvoření nového vážního lístku klikne uživatel na ikonu plus nebo v menu klikne na položku Nový záznam. Na stránce Nový záznam musí uživatel vybrat Datum (automaticky je předvoleno aktuální datum), vyplnit Kód odpadu, který nemusí být pouze číselný, a Hmotnost – netto v kilogramech. Kód odpadu má možnost uživatel také naskenovat kliknutím na tlačítko Naskenovat kód. Dále může vyplnit také Vozidlo RZ, Číslo průkazu (také možnost naskenování kliknutím na tlačítko Naskenovat prů-

kaz), Poznámku a vybrat nebo pořídit fotografii. Po vyplnění může být vážní lístek uložen lokálně (především v případě, kdy chybí připojení ke KS) nebo být ihned na KS odeslán. Pokud je vážní lístek okamžitě odeslán, tak už se lokálně neuloží, protože pro další úpravy tohoto vážního lístku je potřeba stejná úprava i v databázi, ve které se po přijetí vážního lístku vytvoří jeho ID, které je potřeba pro editaci, jenže lokálně není toto ID nijak zjistitelné. Tlačítko Vaše záznamy vrátí uživatele na stejně pojmenovanou již zmiňovanou stránku. V případě, že uživatel nevyplní povinné položky, iniAPKA ho o tom informuje zobrazením dialogového okna s chybou. Pokud si uživatel rozmyslí vložení obrázku při vytváření nového vážního lístku, může obrázek jednoduše smazat kliknutím na tlačítko Odstranit foto.

V případě úpravy záznamu se uživateli zobrazí stejná stránka jako při vytváření záznamu, akorát se do jednotlivých polí předvyplní stávající data.

Stránka O aplikaci zobrazuje krátký popis účelu iniAPKY, nasměrování uživatele na sledování společnosti INISOFT s.r.o. na různých sociálních sítích a její webové stránky, a umožňuje přesměrování na stránku Chybové hlášení.

Na stránce Chybové hlášení je uživatel vyzván, aby popsal chybu, ke které v aplikaci došlo trochu podrobněji. Po kliknutí na tlačítko Odeslat se otevře e-mailová aplikace, ve které je předvyplněna e-mailová adresa, na kterou má být chybové hlášení odesláno, předmět e-mailu, obsah e-mailové zprávy (podrobnější poznámka o chybě, kterou uživatel psal ještě v iniAPCE), a pokud existuje záznam chyby v logovacím souboru, tak se tento logovací soubor zazipuje a vloží se jako příloha do e-mailu.



Obrázek 23: Vývojový diagram iniAPKY
Zdroj: vlastní, vytvořeno pomocí https://www.draw.io

6.7 Formát údajů sbíraných v iniAPCE

V této kapitole budou uvedeny všechny údaje zaznamenávané v aplikaci iniAPKA pro přenos do INI SW. Při samotném připojení ke KS uživatel zadává:

- **přihlašovací jméno:** zatím se nikam neodesílá a tento údaj je předpřipraven pro budoucí vývoj (datový typ String);
- **heslo:** opět se s tímto údajem v aktuální verzi iniAPKY nepracuje a je zde pouze předpřipraven jako v případě přihlašovacího jména (datový typ String);
- **URL:** adresa pro připojení k danému KS (datový typ String);
- **vážní místo:** ID zařízení, ke kterému se má iniAPKA připojit (datový typ Int);
- **položku:** ID položky zařízení (datový typ Int).

Při vytváření nebo úpravě vážního lístku se sbírají následující data:

- **datum:** datum a čas (datový typ DateTime);
- **kód odpadu:** katalogové číslo podle katalogu odpadů, tj. vyhlášky č. 8/2021 Sb. (datový typ String);
- **hmotnost – netto:** numerický údaj v kilogramech (datový typ Decimal);
- **vozidlo RZ:** registrační značka vozidla (datový typ String);
- **číslo průkazu:** pro identifikaci původce odpadu např. občanský průkaz atd. (datový typ String);
- **poznámka:** místo pro informaci, která se nehodí do výše zmíněných polí (datový typ String);
- **fotografie:** fotografie odpadu (datový typ String, formát base64).

Dále iniAPKA může přijmout od KS a zobrazit další informace o vážním lístku, které ovšem uživatel zatím v současné verzi aplikace zadat nebo změnit nemůže. Těmito údaji jsou:

- **číslo:** jedná se spíše o identifikaci vážního lístku ve tvaru např. VL1-2, kde zkratka VL označuje vážní lístek, číslo za touto zkratkou je číselná řada, kterou si uživatel v INI SW může nastavit dle místa vážení, a číslo za pomlčkou označuje pořadí přidání vážního lístku (v tomto případě byl vážní lístek přidán jako druhý), (datový typ String);
- **hmotnost – příjezd:** udává se např. při příjezdu automobilu s odpadem, které je zváženo před vyložením odpadu (datový typ Decimal);
- **hmotnost – odjezd:** zapíše se např. při odjezdu automobilu (datový typ Decimal);
- **název partnera:** jedná se o fyzickou nebo právnickou osobu, která odpad přivezla nebo odvezla (datový typ String);
- **obec partnera:** sídlo fyzické nebo právnické osoby (datový typ String);
- **typ vážního lístku:** označuje to, jestli se jedná o příjem, výdej nebo externí vážení (datový typ Int).

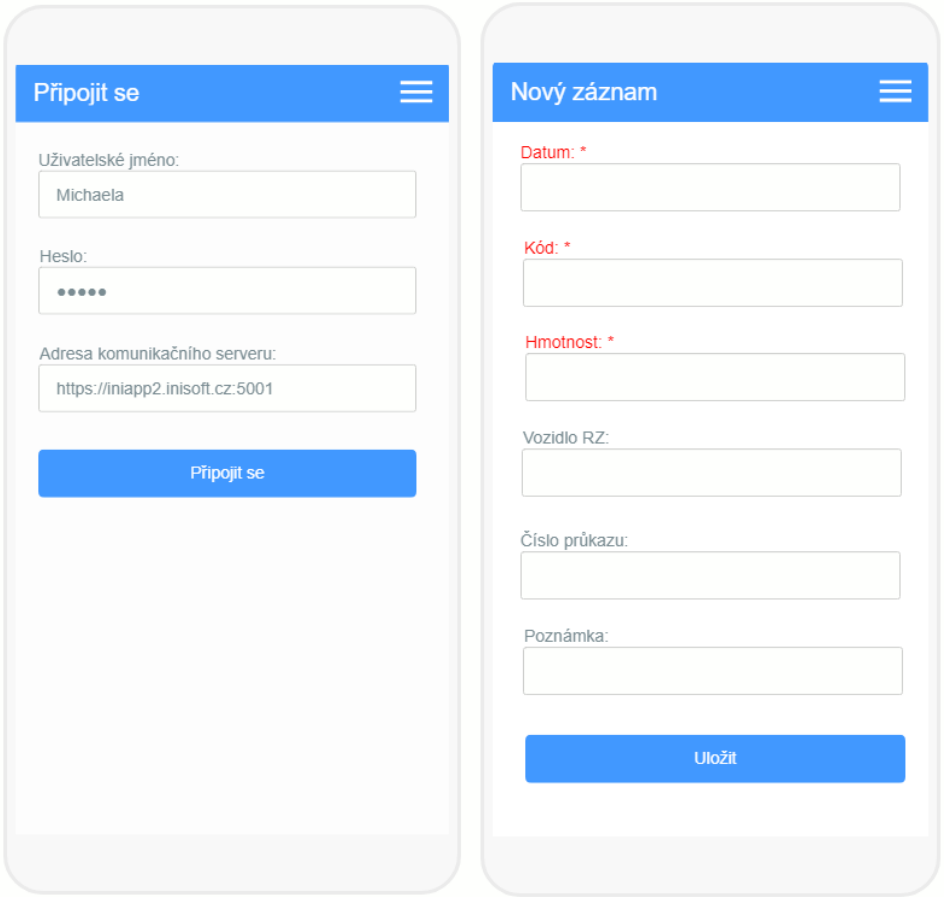
To, že úprava těchto dat prozatím není možná, je z toho důvodu, že pro vyplnění některých položek jsou potřeba číselníky, a pokud by se do mobilní aplikace vložila veškerá potřebná data, kladlo by to na mobilní zařízení větší paměťové nároky. Dále by se muselo hlídat to, aby např. seděly hmotnosti při příjezdu, odjezdu a netto. Platí tedy, že netto hmotnost je hmotnost při odjezdu odečtená od hmotnosti při příjezdu.

V budoucích verzích iniAPKY je ale velmi pravděpodobné, že bude možná úprava i těchto výše zmíněných dat, a nejen těch. Vážní lístek v IS ENVITĚ obsahuje totiž ještě pár dalších údajů. Otázkou však je, jestli pro rychlou evidenci je vyplňování všech dat žádoucí. To se zjistí, až při nasazení aplikace do produkčního prostředí a zpětné odezvy od uživatelů.

6.8 Grafický návrh uživatelského rozhraní

Zpočátku jsem vytvářela grafické návrhy rozložení jednotlivých elementů, ale v průběhu vývoje se od tohoto způsobu návrhu ustoupilo s tím, že bude dopředu vždy dohodnuto, co a kam se má přidat, upravit nebo odebrat, a jak to má vypadat.

K tvorbě grafických návrhů jsem používala online nástroj s názvem FluidUI (https://www.fluidui.com). Je v něm mnoho předpřipravených elementů, které se dají použít, a díky tomu tak vytvořit jednoduše a rychle layout jednotlivých stránek.



Obrázek 24: Příklad prvotních grafických návrhů z online nástroje FluidUI
Zdroj: vlastní zpracování, vytvořeno pomocí https://www.fluidui.com

Grafické rozhraní aktuální verze iniAPKY obsahuje barvy, které jsou typické pro INI SW, tzn. modrou, zelenou a bílou barvu. V levé části je hamburger menu pro rychlejší přístup k jednotlivým akcím (možnostem). IniAPKA komunikuje s uživatelem pomocí dialogových oken nebo zobrazením hlášek s informacemi o průběhu požadované akce. Ovládání iniAPKY je jednoduché a intuitivní, aby budoucí potencionální uživatelé neměli problém iniAPKU používat.

6.9 Tvorba zdrojového kódu

Projekt iniAPKA je tvořen ze dvou menších projektů: GrpcShared a INIApp.

V projektu GrpcShared je třída DataCollector.proto, která definuje všechny metody a třídy pro komunikaci s KS, které jsou podporovány, a závislosti ve formě NuGet balíčků (Google.Protobuf – 3.21.10, Grpc.Net.Client – 2.50.0, Grpc.Tools – 2.51.0-pre1).

Ve druhém projektu s názvem INIApp je kód samotné aplikace. Kromě složek typických pro .NET MAUI projekt jsou zde dle architektonického návrhu MVVM složky: Models, Views a ViewModels.

Ve složce Models jsou modely konkrétních objektů, se kterými se v iniAPCE pracuje, např. ConnectionModel.cs – třída s definicí dat pro připojení se ke komunikačnímu serveru, WasteItemModel.cs – třída s definicí dat pro vážní lístek, který je zobrazován uživateli atd.

Ve složce Views jsou tzv. .NET MAUI ContentPage (XAML) soubory, které jsou tvořeny dvěma částmi: část pro definici rozložení prvků (soubor s příponou .xaml) a část pro kód (soubor s příponou .xaml.cs). Tato druhá část pro kód by měla např. navázat první část s příslušným ViewModelem nebo pomoci např. s animacemi, které by se obtížně definovaly v části pro samotné rozložení prvků. Příkladem je AboutAppPage.xaml (AboutAppPage.xaml.cs) – view pro zobrazení stránky O aplikaci nebo CreateRecordPage.xaml (CreateRecordPage.xaml.cs) – view pro zobrazení stránky pro vytvoření nového vážního lístku.

Ve složce ViewModels jsou C# třídy pro propojení Views a Models a samotná logika požadovaných akcí, např. WasteListViewModel.cs – třída pro získání a manipulaci se

záznamy o vážních lístcích, ReportErrorViewModel.cs – třída pro odeslání chybového hlášení.

Dále projekt INIApp obsahuje složky jako Services a Converters. Services jsou služby, komunikující např. s DataCollectorem z GrpcShared pro komunikaci s KS, ukládání dat do DB atd. Důvod zařazení těchto metod do jednotlivých tříd služeb je takový, že je potřeba některé stejné funkce volat z více míst. Spíše než aby se tedy do těchto míst, ve kterých jsou potřebné kopírovaly, tak se volají právě z těchto služeb. Ve složce Converters je pouze jedna třída s názvem DateTimeJsonConverter.cs, která pomáhá s převodem času např. při příchodu údaje o datu a času v JSONu z KS, který by bez tohoto konvertoru nemohl být rozparsován.

Ve složce Resources je soubor typu ResourceFile s názvem SharedResource.resx. V tomto souboru jsou uloženy veškeré texty (popisky, placeholdery aj.), které se v iniAPCE zobrazují. Díky tomu je následná úprava textů jednodušší a zároveň je to takový základ pro budoucí podporu cizích jazyků (v případě společnosti INISOFT s.r.o. třeba slovenského jazyka). V této složce je podsložka s názvem Fonts, ve které jsou vloženy fonty, které se v aplikaci používají. Nicméně je v ní také soubor s názvem fa-solid.ttf, díky kterému jsou v iniAPCE zobrazovány ikonky. Jedná se o soubor z knihovny Font Awesome. Pro přehled toho, jak ikonky vypadají jsem využívala online konvertoru (https://andreinitescu.github.io/IconFont2Code/), který dokázal soubor přečíst a ikonky vykreslit. Tudíž následný výběr potřebné ikonky byl snadnější.

Důležitým souborem je appsettings.json, kde jsou uloženy důležité údaje, které jsou využívány v iniAPCE:

- název databáze pro ukládání dat z iniAPKY;
- definice údajů pro odeslání chybového hlášení (e-mailové adresy pro odeslání, předmět e-mailu, název souboru, který má být odeslán v příloze).

Databáze, do které se ukládají vážní lístky a přihlašovací údaje, je typu SQLite a je nazvaná jako INIAppDB.db. Obsahuje dvě tabulky: wasteList a connections. Názvy samy o sobě vypovídají o tom, jaká data jsou v nich uložena.

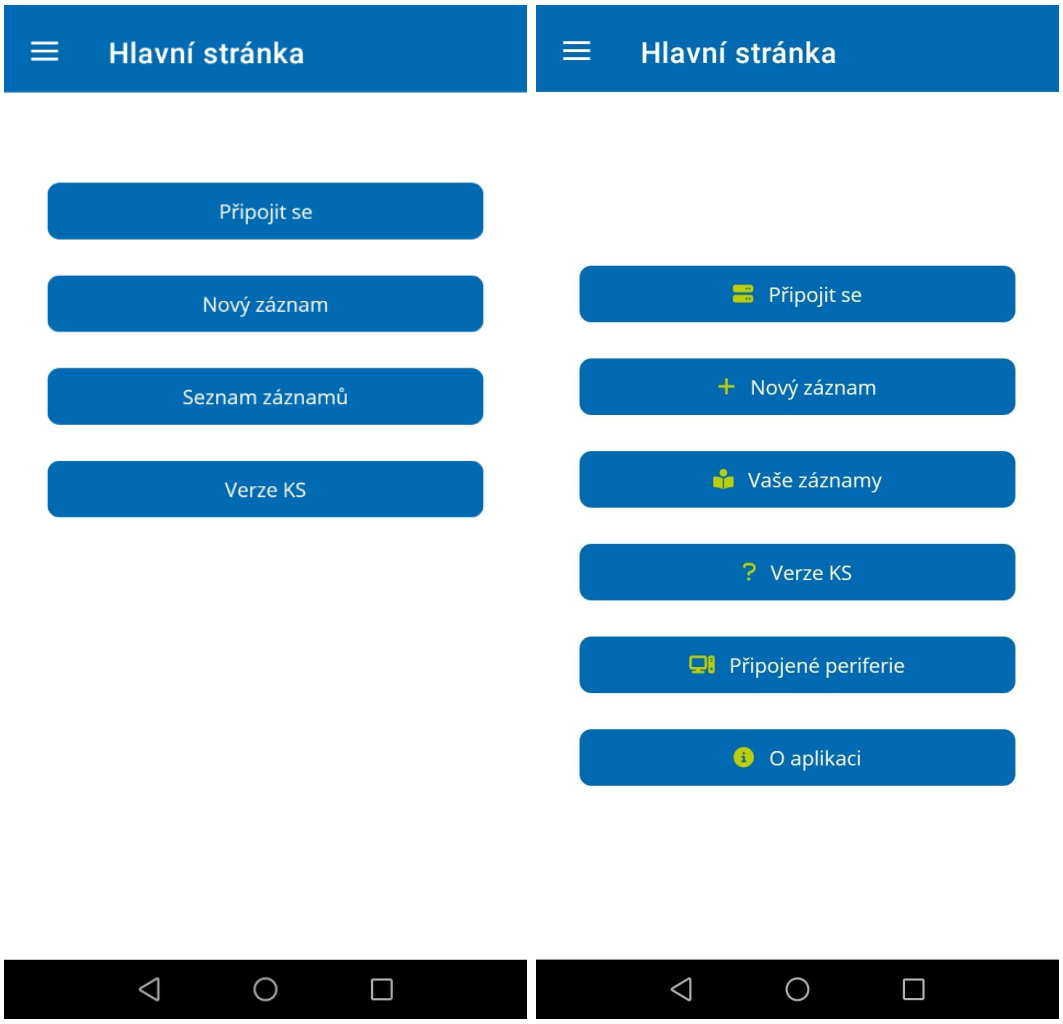
Dokumentační komentáře metod v kódu jsou umístěny vždy nad samotné metody. Jiná dokumentace zatím neexistuje.

6.10 Průběh vývoje iniAPKY

IniAPKA, i přes to, že je to nově vznikající aplikace, si již prošla řadou změn. Od změn grafického uživatelského rozhraní po samotné fungování aplikace (např. přepisování zdrojového kódu pro vytvoření přehlednějšího celku atd.).

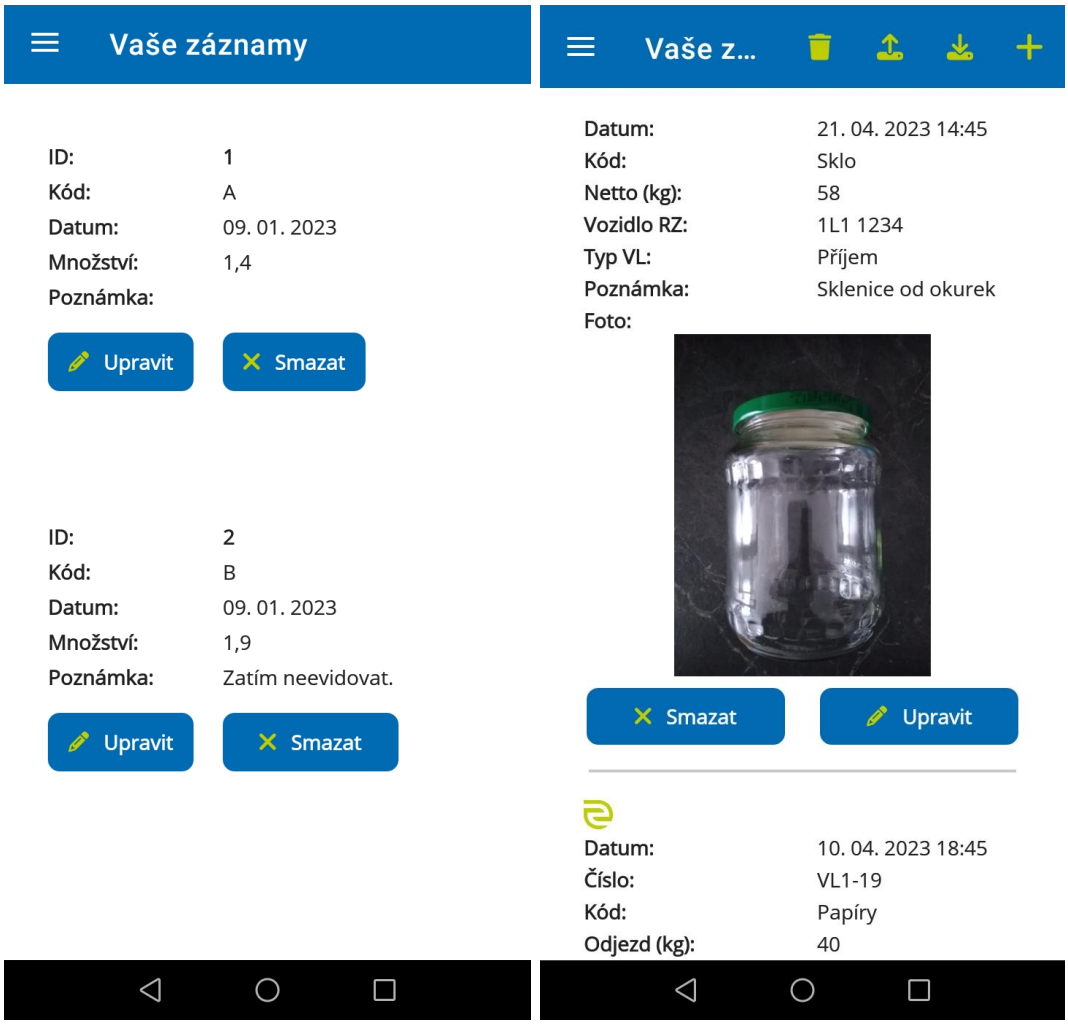
Hlavní změna ve vývoji grafického uživatelského rozhraní a ovládání aplikace byla v tom, že se odstranila původně zamýšlená hlavní stránka pro navigaci uživatelů, která byla nahrazená tím, že se nyní uživateli po spuštění aplikace zobrazí stránka s připojením ke KS. Hlavní změna z hlediska uspořádání zdrojového kódu je ve vytvoření Services pro volání stejných metod na více místech a snaha o dodržení MVVM architektury, která také vznikala postupně s vývojem.

Na obrázcích níže je vývoj hlavní (domovské) stránky s menu. Ta byla v aplikaci obsažena, jak již bylo zmíněno, pouze v pár prvních verzích, pak jsme se dohodli se společností INISOFT s.r.o. na jejím odebrání kvůli jednoduššímu a intuitivnějšímu ovládání. Hlavní stránka (menu) byla tvořena jednoduchým rozložením tlačítek pro jednotlivé uživatelské akce. V levé části obrázku je snímek této hlavní stránky z jedné z prvních verzí aplikace, v pravé části je snímek stejné stránky z novější verze iniAPKY. Novější verze má oproti té starší u názvů možností ikonky, a navíc dvě možnosti (zobrazení připojených zařízení a zobrazení stránky O aplikaci).



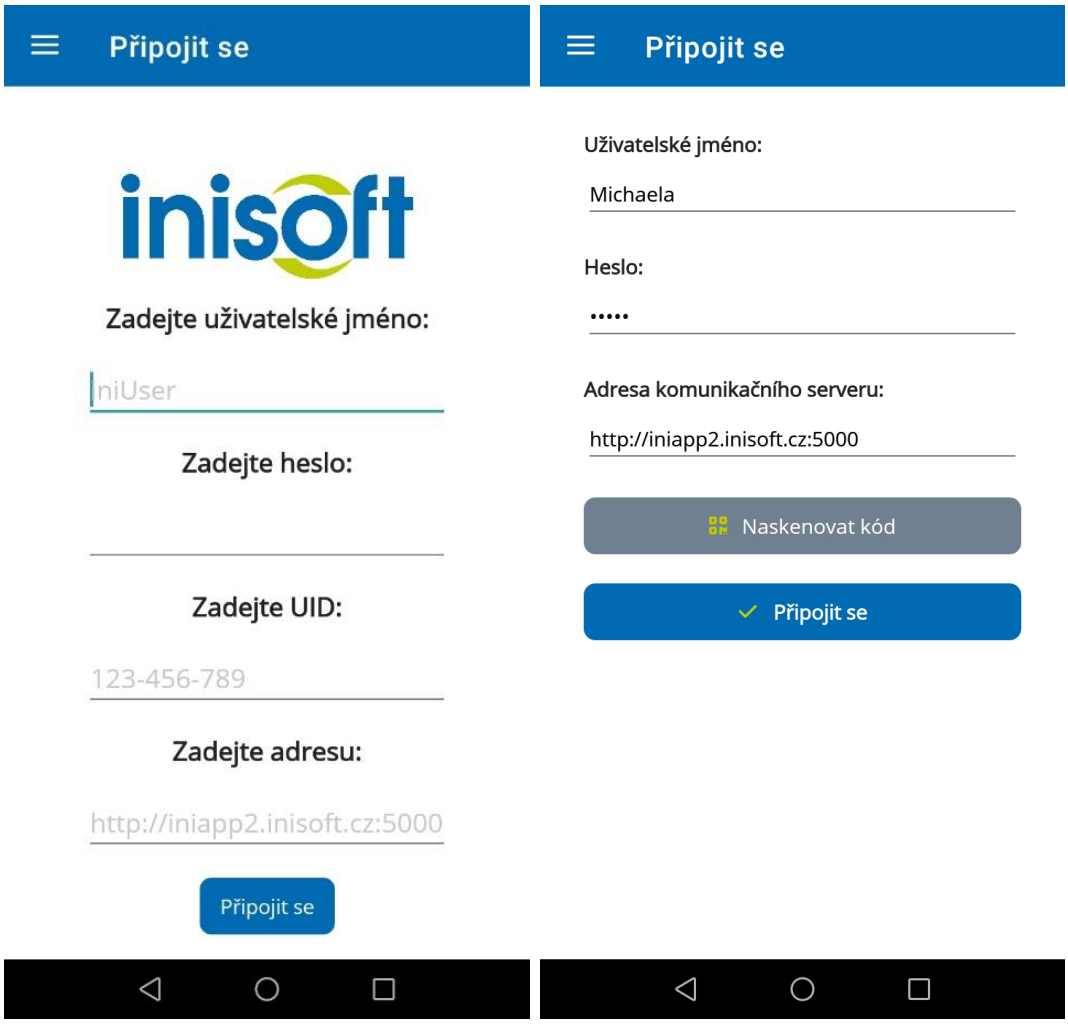
Obrázek 25: Vývoj hlavní stránky
Zdroj: iniAPKA

Opět na obrázcích níže je vidět postupný vývoj layoutu stránky Vaše záznamy. Kdy vlevo je snímek ze starší verze iniAPKY, kde se na této stránce zobrazují pouze lokálně uložené záznamy se striktním zobrazením všech vlastností vážního lístku (i když některé byly prázdné). Vpravo je snímek stejné stránky z novější verze iniAPKY, ve které je již možné přijmout a zobrazit záznamy z KS, odstranit všechny lokálně uložené vážní lístky, přidat nový vážní lístek, dále se zobrazují jen ty vlastnosti vážního lístku, jejichž hodnota není prázdná. Rozlišuje se také, jestli daný vážní lístek pochází z KS nebo jestli byl založen pouze v iniAPCE. Pokud vážní lístek přišel z KS, tak ho uživatel v současné verzi nemůže smazat, ale může upravit. Pokud si ale daný záznam vytvořil uživatel lokálně, a ještě ho neodeslal, tak je i smazání možné.



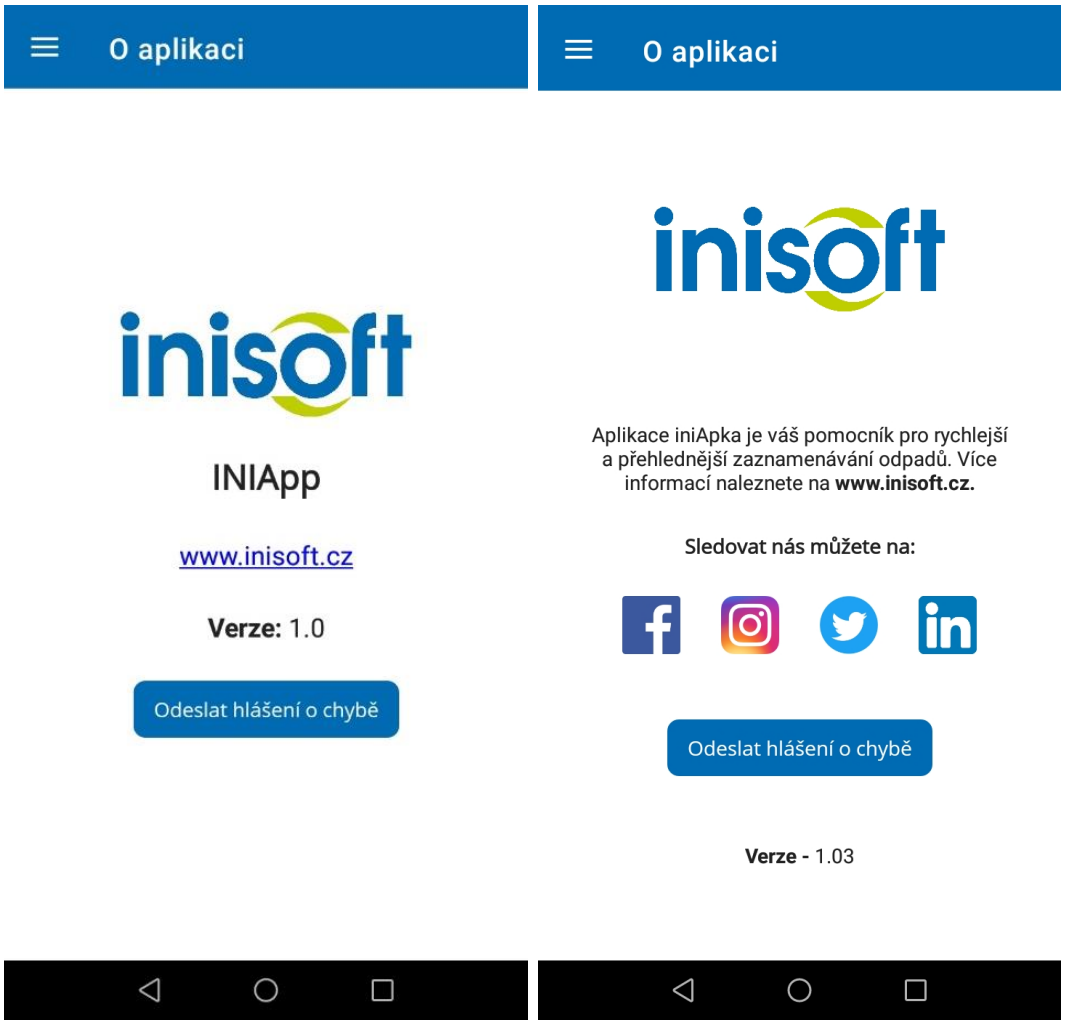
Obrázek 26: Vývoj stránky Vaše záznamy
Zdroj: iniAPKA

Vývoj stránky Připojit se je vidět na obrázcích níže. Vlevo je snímek ze starší verze, kde bylo logo společnosti INISOFT s.r.o., pole pro vyplnění uživatelského jména, hesla, UID (ID registrace) a URL pro připojení ke KS. Vpravo je snímek z novější verze iniAPKY, kde již není logo a pole pro vyplnění UID. Navíc je zde možnost načíst 1D/2D kód pro vyplnění pole Adresa komunikačního serveru. Je nutné zmínit, že v minulých verzích a ani v současné verzi nemá uživatelské jméno a heslo žádnou vazbu na jakékoli volání nebo sběr dat o uživateli. Jedná se o předpřípravu pro budoucí funkce, zejména pro kontrolu uživatelů, případně pro umožnění některých speciálních funkcí.



Obrázek 27: Vývoj stránky Připojit se
Zdroj: iniAPKA

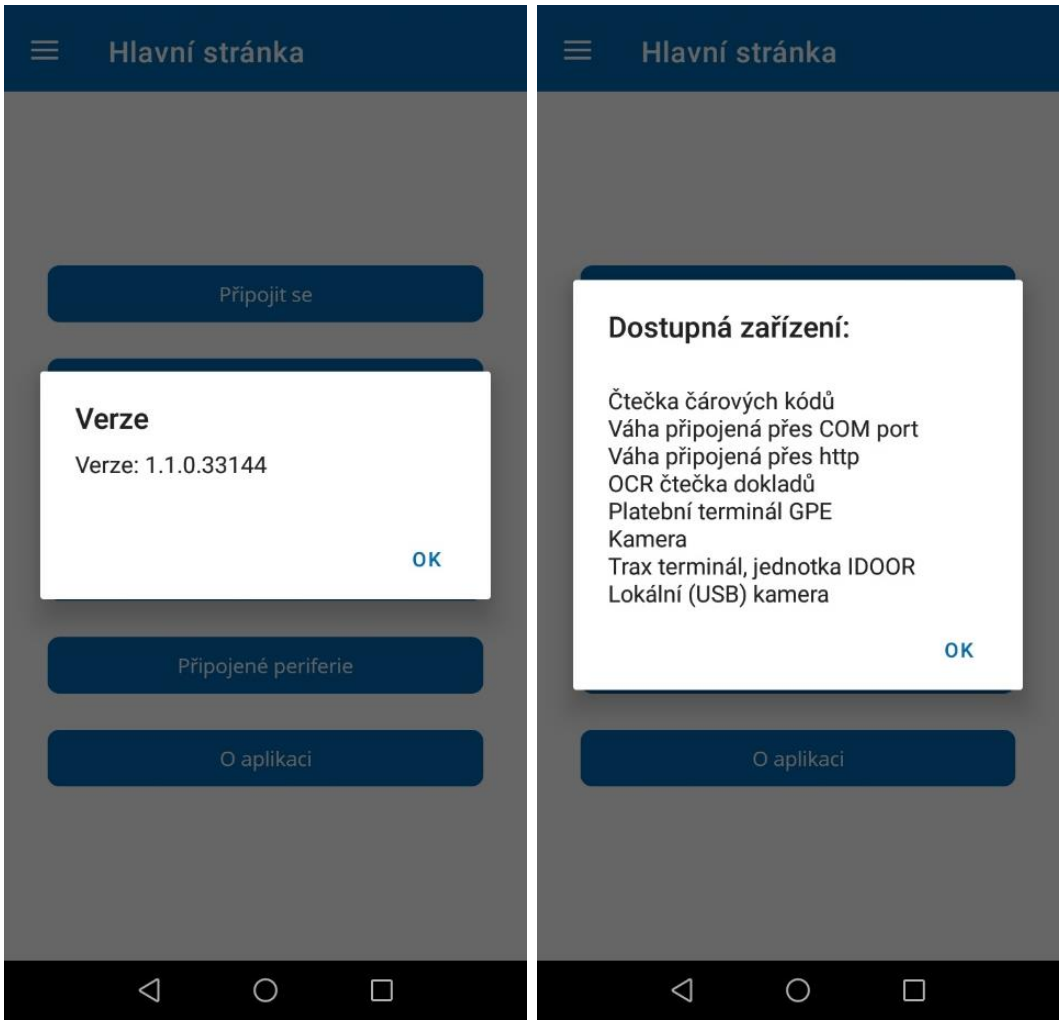
Stránka O aplikaci se moc nezměnila. V novější verzi přibyly obrázky log Facebooku, Instagramu, Twitteru a LinkedInu pro přesměrování uživatele na profil společnosti INISOFT s.r.o. na dané sociální síti. A od publikace aplikace na Google Play je v aplikaci postupně zvyšována minor verze iniAPKY.



Obrázek 28: Vývoj stránky O aplikaci
Zdroj: iniAPKA

Aplikace v předchozích verzích umožňovala po úspěšném připojení ke KS uživateli zobrazit verzi KS – k vidění na obrázku níže vlevo a seznam dostupných zařízení vpravo.

Nyní již uživatel nemá možnost zjistit, jakou verzi KS má. Na druhou stranu je pro uživatele téměř povinností vybrat si zařízení ihned po připojení se ke KS. IniAPKA sice uživatele pustí jinam, pokud nevybere žádné zařízení, ale bude se chovat tak, jako by nebyla připojena ke KS.



Obrázek 29: Verze KS a seznam dostupných zařízení
Zdroj: iniAPKA

6.11 Vytvoření komunikace s KS prostřednictvím gRPC

Důležitým požadavkem bylo, aby iniAPKA podporovala šifrovanou komunikaci s KS kvůli bezpečnosti přenosu dat. I přes fakt, že .NET MAUI podporuje šifrovanou i nešifrovanou komunikaci prostřednictvím gRPC, tak cesta k dosažení a prokopnutí šifrované komunikace nebyla úplně jednoduchá a zabrala opravdu hodně času.

6.11.1 Nešifrované připojení ke KS

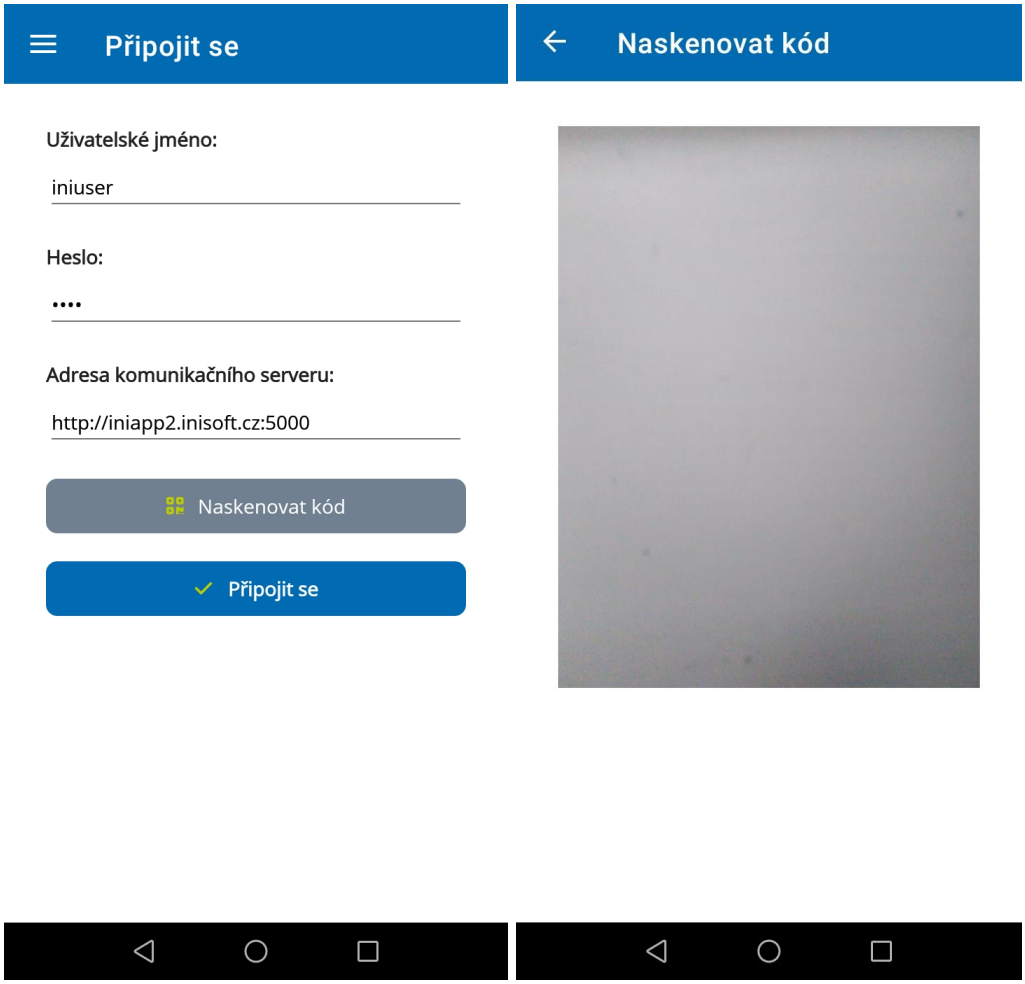
V nešifrovaném připojení ke KS se nevyskytl žádný problém. Stačilo do projektu INIApp přidat následující NuGet balíčky: Google.Protobuf, Grpc.Net.Client a Grpc.Tools a také soubor ProtoBuf, který obsahuje popis funkcí a zpráv, a přes

který je možné s KS komunikovat. Dále stačilo v místě, ve kterém se zahajuje komunikace s KS zavolat metodu z NuGet balíčku s názvem Grpc.Net.Client s parametrem address, což je URL, pro kterou se má gRPC kanál vytvořit:

```
GrpcChannel.ForAddress(address),
```

společně s vytvořením klienta zavoláním metody z kódu, který byl vygenerován z ProtoBuf souboru.

Na obrázku níže v levé části je prázdný formulář, který je určen k zadání dat potřebných k připojení ke KS (uživatelské jméno, heslo, adresa – URL KS). Vpravo stránka pro naskenování 1D/2D kódu.



Obrázek 30: Formulář pro vytvoření připojení ke KS
Zdroj: iniAPKA

6.11.2 Šifrované připojení ke KS

Implementace šifrovaného připojení ke KS již nebyla tak jednoduchá. První vyzkoušená cesta byla prakticky stejná s implementací nešifrovaného připojení. Bohužel, při jejím otestování se vyskytla následující chyba: *Error starting gRPC call*.

Byly tedy přidány třídy *HttpClient* a *HttpClientHandler*, které se používají pro připojení k serveru s certifikátem s nutným ověřením. *HttpClientHandler* může nastavit vlastnost *ServerCertificateCustomValidationCallback* na to, aby vždy vracela *true*, tzn. že certifikát projde ověřením. Avšak ani po této úpravě nebylo možné navázat spojení s KS, a vyskytla se následující chyba: *Error starting gRPC call. WebException: Hostname XXX not verified*.

Do mobilního zařízení byl nainstalován SSL certifikát, byla pozměněna konfigurace KS a přidal se router, který má za úkol simulovat autonomní samostatnou síť. Za těchto podmínek bylo otestováno řešení popsané v odstavci výše. Spojení s KS opět navázáno nebylo a skončilo chybou: *HTTP/1.1 is not supported.* Vznikající iniAPKA byla vyzkoušena na různých mobilních zařízeních s různými verzemi OS Android, avšak také bezúspěšně.

Po tom, co bylo technicky vše přizpůsobeno k tomu, aby šifrované připojení mohlo být realizováno – bylo téměř jasné, že důvod neúspěšného navázání komunikace bude v implementaci. Vyzkoušených cest byla opravdu celá řada např. přidání vyexportovaného certifikátu přímo do projektu (do části pro OS Android), různé nastavování *ChannelCredentials* nebo verze 2.0 pro třídu *HttpClient*, vyzkoušení třídy *SocketsHttpHandler*, využití implementace a úprava *Http2Handler* speciálně pro OS Android atd. Všechny pokusy skončily s různými chybami.

Průlom nastal až v úpravě serverové části KS (napsána v .NET Core) přidáním NuGet balíčku Grpc.AspNetCore.Web a jeho dvou metod do souboru *Startup.cs* do části *Configure:*

```
app.UseGrpcWeb(new GrpcWebOptions { DefaultEnabled = true });

app.UseEndpoints(endpoints =>
 { endpoints.MapGrpcService<název služby>().EnableGrpcWeb(); });
```

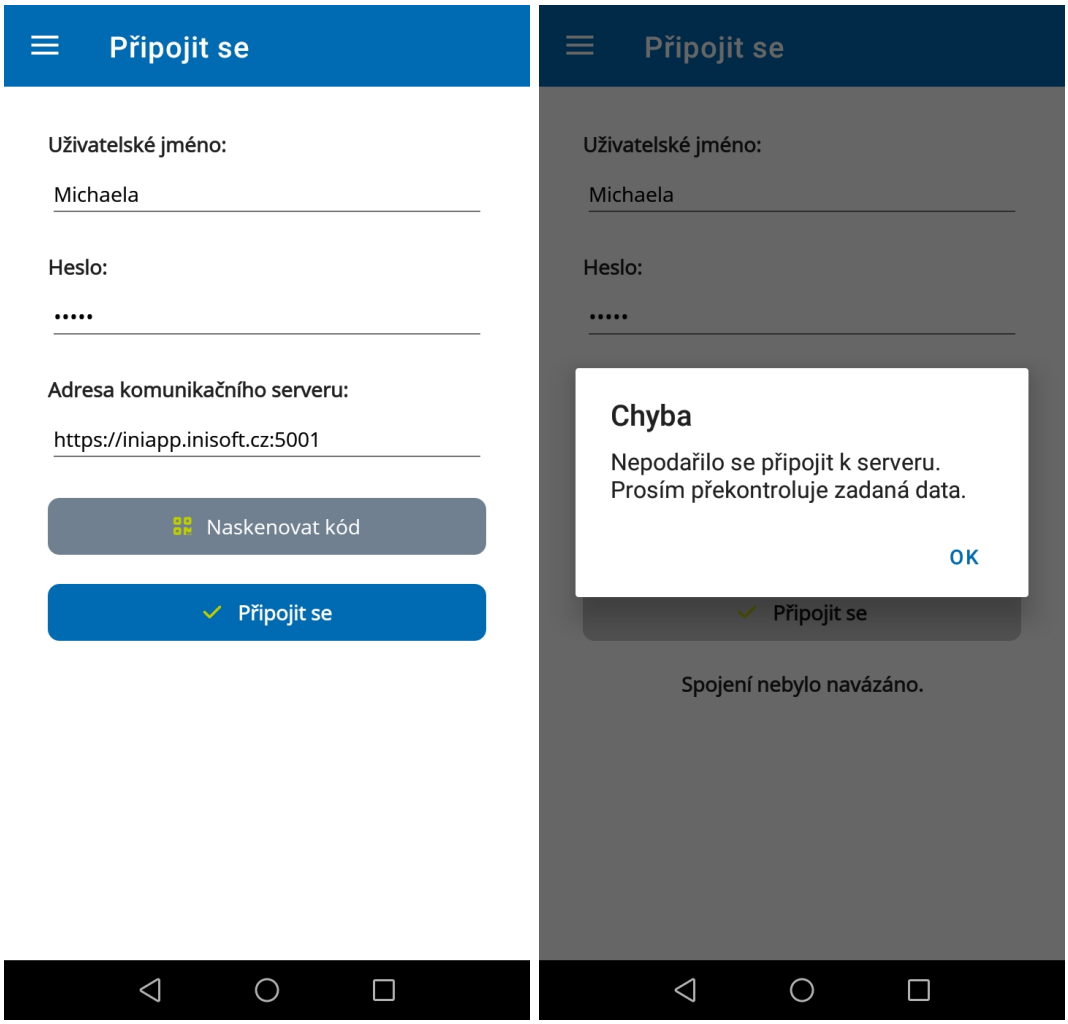
Jde vlastně o přidání serverové podpory pro HTTP/2.0 gRPC volání. Do iniAPKY byl přidán NuGet balíček s názvem Grpc.Net.Client.Web, který obsahuje metodu:

```
GrpcWebHandler(GrpcWebMode mode, HttpMessageHandler innerHandler),
```

kde *mode* je nastaven na *GrpcWebMode.GrpcWeb*, což je nastavení *content-type*, a *innerHandler* na *HttpClientHandler*, kterému se se přidává již zmíněná metoda *ServerCertificateCustomValidationCallback*. *GrpcWebHandler* je dále předán k vytvoření instance třídy *HttpClient* a další postup je stejný jako u nešifrovaného připojení.

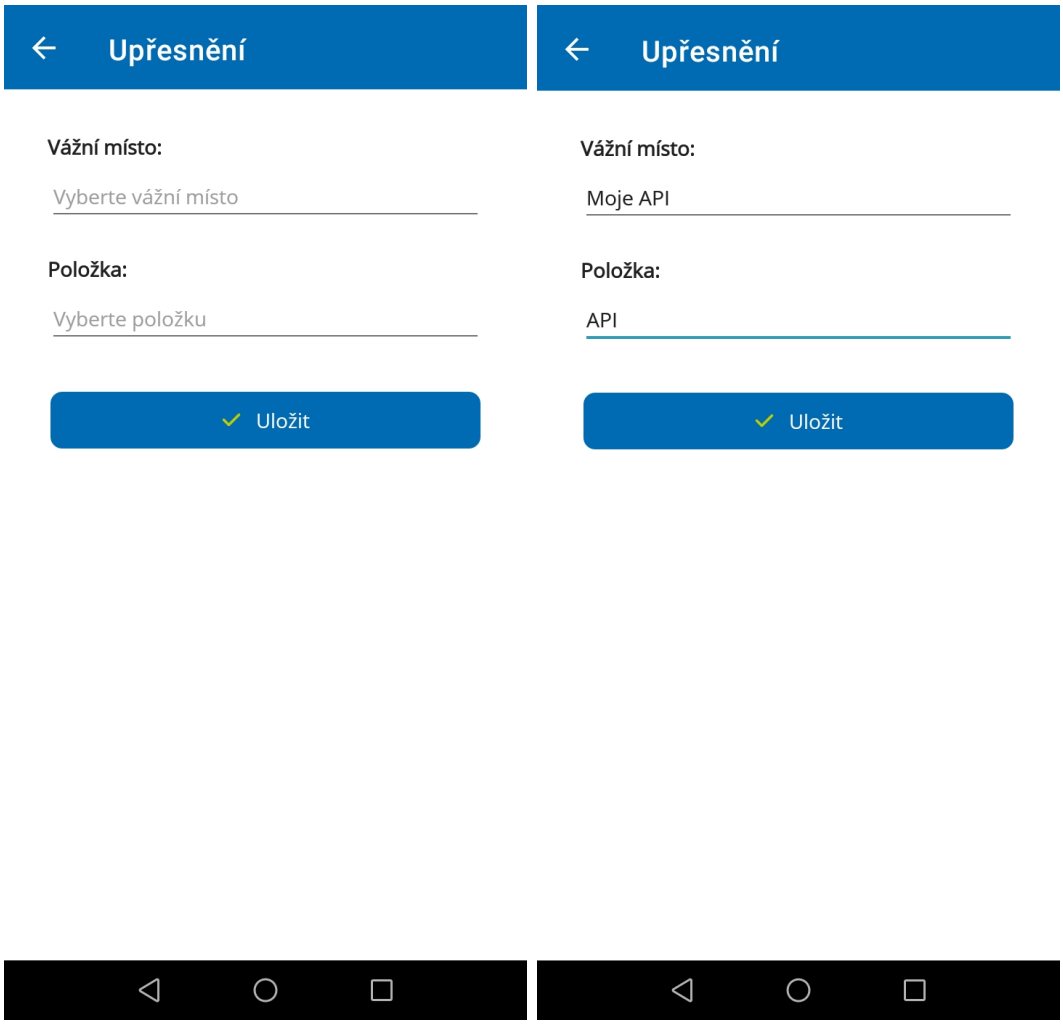
Původně byla jako *GrpcWebMode* nastavena hodnota *GrpcWebMode.GrpcWebText*, díky které se ale někdy nepodařilo odeslat vážní lístek na KS, a skončilo to vždy chybou: *Unexpected end of data when reading base64 content.* Po bádání toho, jak tuto chybu opravit, jsem změnila tuto hodnotu na *GrpcWebMode.GrpcWeb*, a odeslání bylo úspěšné.

V levé části na obrázku níže je vyplněný formulář s URL pro šifrované připojení ke KS (lze si povšimnout, že tato komunikace by měla běžet na portu 5001, nešifrovaná na portu 5000). V pravé části je dialogové okno s informací pro uživatele, že se nepodařilo připojit ke KS např. v případě zadání špatné adresy pro připojení ke KS atd.



Obrázek 31: Formulář pro vytvoření připojení ke KS a dialogové okno
Zdroj: iniAPKA

Po úspěšném připojení ke KS je nutné vybrat vážní místo a položku vážního místa. V tomto případě musí uživatel vybrat vážní místo typu ISEAPI_REST a typ položky ENVITA_API_JSON (samozřejmě si uživatelé mohou tyto místa pojmenovávat pro přehlednost různě, takže na obrázcích níže je název vážního místa Moje API a název položky API).



Obrázek 32: Výběr vážního místa a položky
Zdroj: iniAPKA

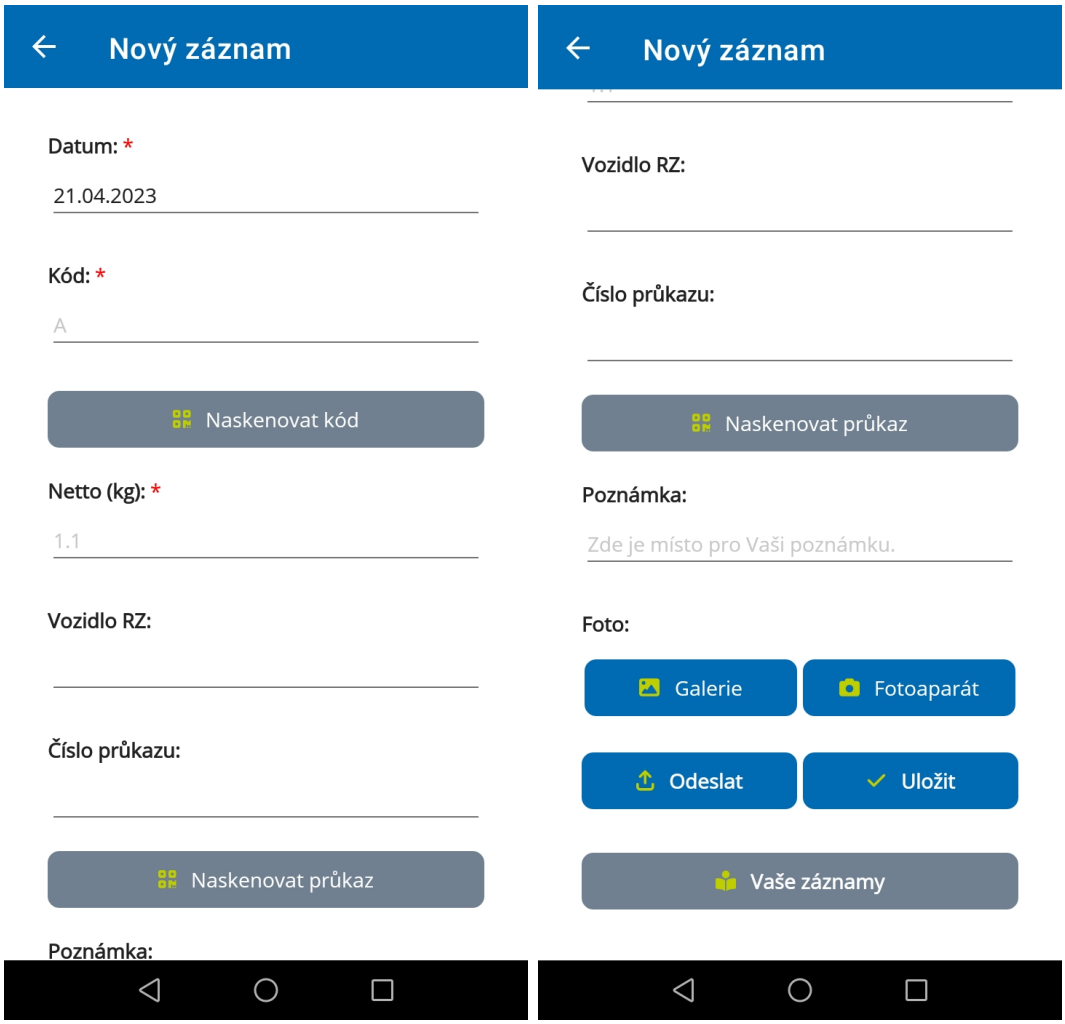
6.12 Práce s vážními lístky

IniAPKA umožňuje uživateli vložit, upravit nebo smazat záznam. Uživatel si také může prohlídnout seznam svých záznamů a získat nebo odeslat záznamy prostřednictvím KS z nebo do INI SW. Data jsou v iniAPCE ukládána do lokální SQLite databáze.

Na obrázku níže v levé části je první část formuláře, která se vešla do snímku, pro vytvoření nového záznamu o dávce odpadu (vážního lístku). V pravé části je zbytek formuláře. Lze vidět, že formulář obsahuje povinné položky, kterými jsou kód odpadu, množství odpadu v kilogramech a datum. Tyto položky jsou označeny červenou hvězdičkou. Pokud uživatel tyto položky nevyplní, záznam nebude možné uložit ani odeslat, a uživatel bude upozorněn dialogovým oknem o tom, aby vyplnil povinná data. V opačném případě budou data uložena nebo odeslána a uživatel o tom bude

také informován. K vážnímu lístku lze dále zadat další nepovinné atributy: RZ vozidla, číslo průkazu, poznámku a fotografii. Kód odpadu a číslo průkazu je možné zadat naskenováním 1D/2D kódu. Fotografie může být vybrána z galerie nebo vyfocena fotoaparátem mobilu přímo v aplikaci. V případě, že si uživatel rozmyslí vložení fotografie, tak stačí, aby klikl na tlačítko Odstranit foto, které se zobrazí pouze v případě, že je fotografie vložena.

Pokud uživatel chce vážní lístek rovnou odeslat (zmáčkne tlačítko Odeslat), tak se v případě, že je iniAPKA připojena ke KS, vážní lístek odešle a zároveň nebude uložen do lokální databáze. To je z důvodu toho, že když je vážní lístek vkládán do INI databáze, tak se vytvoří ID záznamu. Díky tomuto ID lze pak záznam v INI databázi editovat nebo vymazat. IniAPKA po odeslání vytvořeného vážního lístku nemá k dispozici toto ID z INI databáze, takže pokud by uživatel chtěl dále tento vážní lístek upravovat, tak se v INI databázi změny u konkrétního vážního lístku nepropíší. Pokud uživatel chce mít vytvořený vážní lístek ještě u sebe, protože ví, že ho bude potřebovat upravit nebo si jeho vyplněním není jist, může si ho uložit lokálně (tlačítko Uložit).

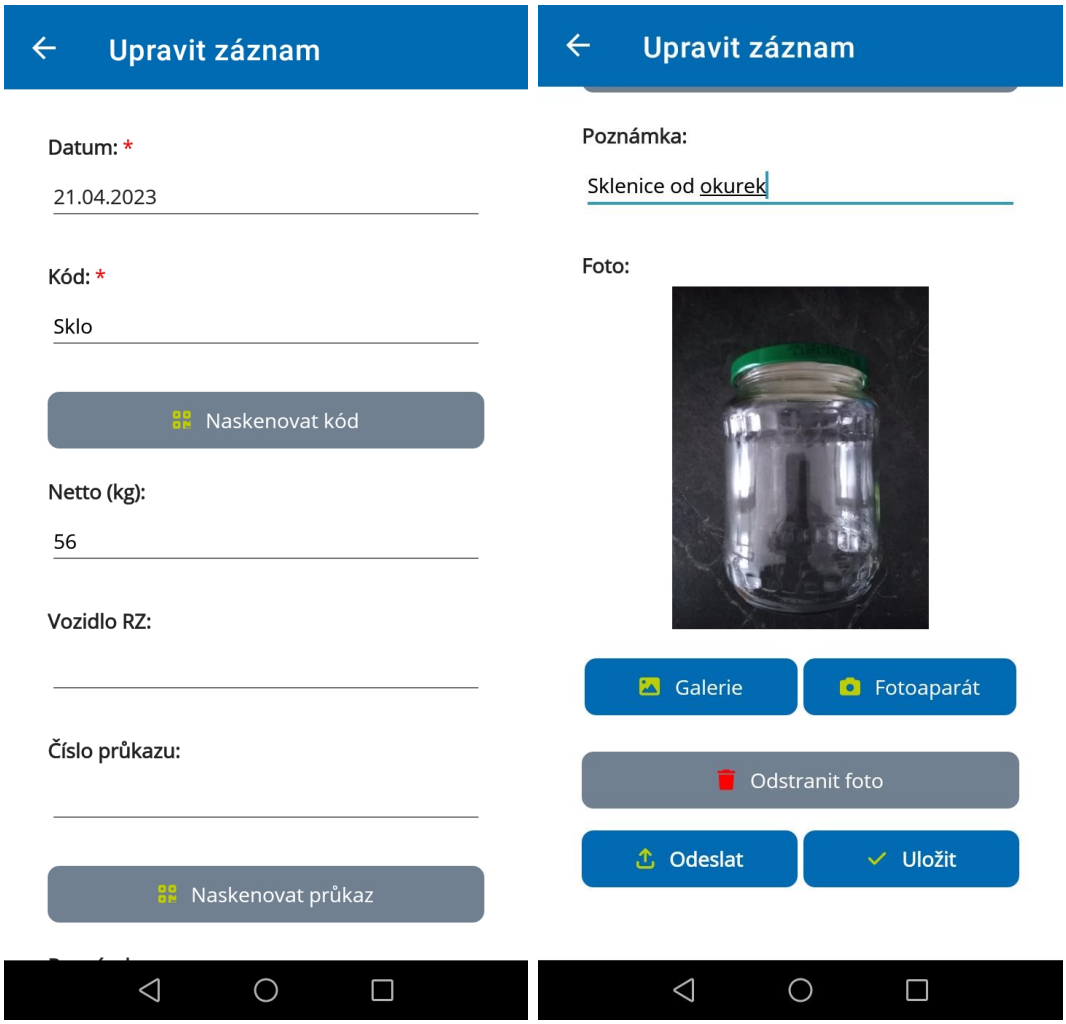


Obrázek 33: Formulář pro založení nového záznamu (vážního lístku)
Zdroj: iniAPKA

Pokud vše proběhne správně, v INI SW se zobrazí vytvořený vážní lístek. V INI SW je možné vážní lístek číst, upravit a smazat. Níže jsou obrázky, které více přibližují celý proces odeslání vyplněného formuláře do IS ENVITY.

V případě úpravy vážního lístku se uživateli zobrazí stejná stránka jako při vytváření nového záznamu s tím rozdílem, že jsou ve formuláři předvyplněna data, která jsou u konkrétního vážního lístku uložena. Uživatel může upravený vážní lístek uložit pouze lokálně nebo ho odeslat do IS ENVITY. Pokud upravuje vážní lístek, tak to bude vždy ten, který byl získán z IS ENVITY nebo ten, který je lokálně uložen, ale není ještě do IS ENVITY odeslán.

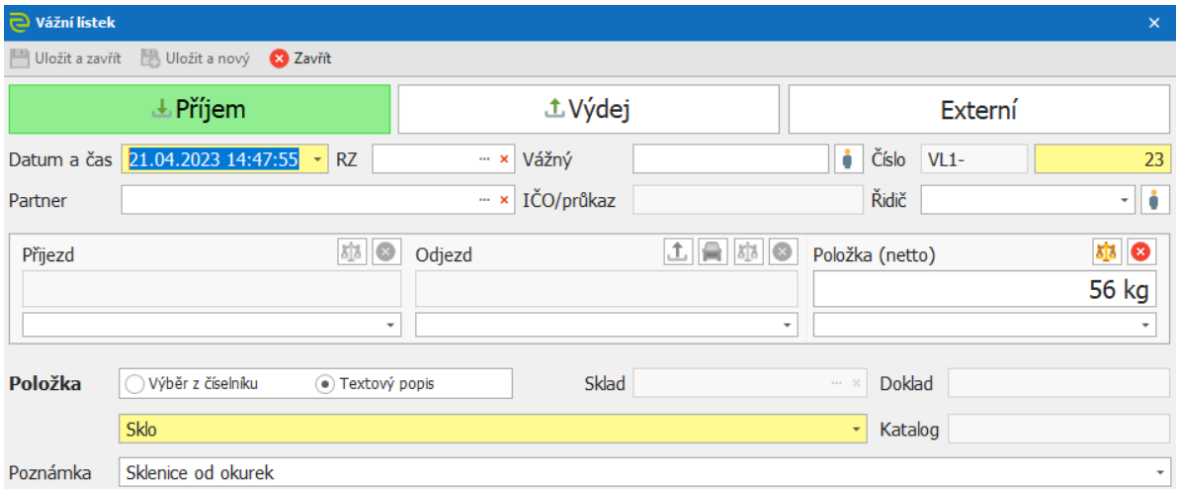
Smazání vážního lístku lze na stránce Vaše záznamy, a mazat lze pouze takové vážní lístky, které jsou lokálně uložené, nikoli ty, které jsou uloženy v INI databázi.



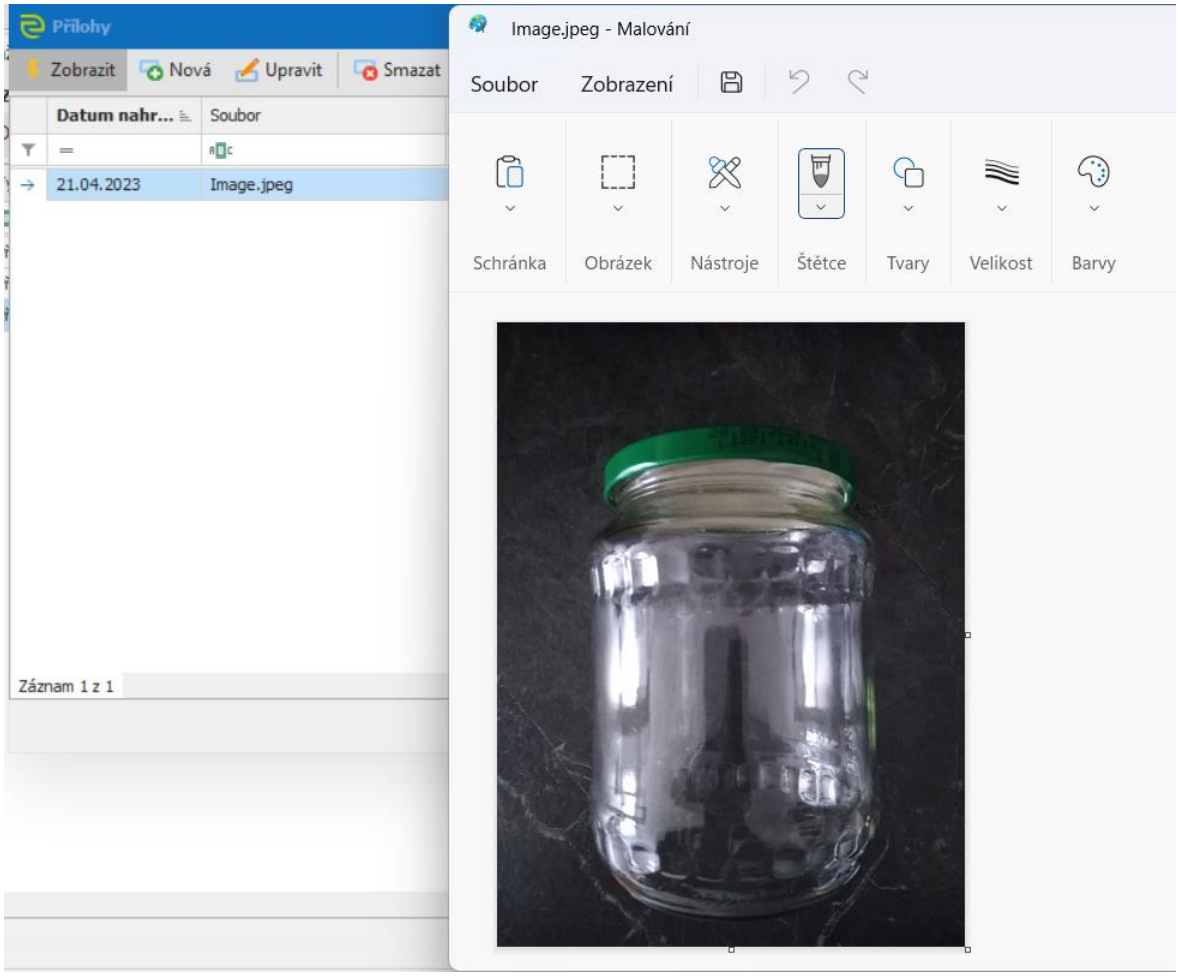
Obrázek 34: Vyplněný formulář pro vytvoření nového záznamu
Zdroj: iniAPKA

Datum a čas	Číslo	Typ	RZ	Partner - n...	Partner -...	Partner - adresa	Položka - kód	Položka - název	Příjezd	Odje...	Netto	Neúplný	Poznámka
10.04.2023 18:45	VL1-19	Příjem	1L1 1234	INISOFT s...	25417657	Kút, Želechovice nad ...		Papíry		40 kg		☑	Velké krabice
10.04.2023 21:10	VL1-21	Příjem	1L1 1234					Plasty	125 kg			☑	Neevidovat!
21.04.2023 14:47	VL1-23	Příjem						Sklo			56 kg	☐	Sklenice od okurek

Obrázek 35: Přehled všech vážních lístků v IS ENVITĚ
Zdroj: IS ENVITA



Obrázek 36: Detail přijatého vážního lístku z iniAPKY v IS ENVITĚ
Zdroj: IS ENVITA

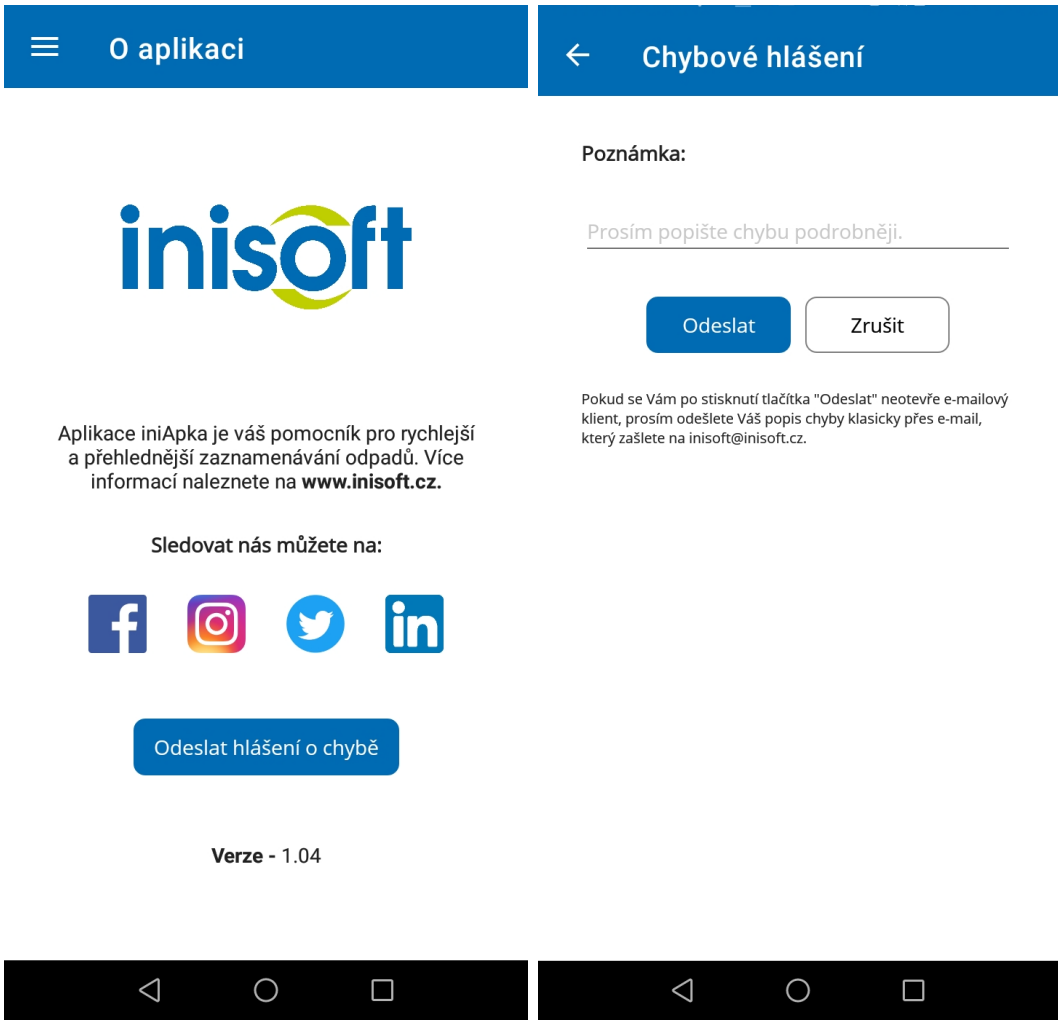


Obrázek 37: Obrázek připnutý k vážnímu lístku z iniAPKY v IS ENVITĚ
Zdroj: IS ENVITA a program Malování

6.13 O aplikaci a odeslání chybového hlášení

Na stránce O aplikaci (na obrázku níže vlevo) mají uživatelé možnost přečíst si krátký popis aplikace, mohou se dostat na webovou stránku společnosti INISOFT s.r.o. a na různé profily společnosti INISOFT s.r.o. na sociálních sítích. Mají možnost kliknout na tlačítko Odeslat chybové hlášení a v dolní části stránky je možné zjistit verzi nainstalované aplikace.

Pokud uživatel klikne na tlačítko Odeslat chybové hlášení, otevře se mu stránka s názvem Chybové hlášení (na obrázku níže v pravé části). Uživatel je nejprve vyzván, aby popsal nastalou chybu podrobněji, každopádně není to povinné. Pokud stiskne tlačítko Odeslat, otevře se uživatelův e-mailový klient, ve kterém bude předvyplněn příjemce, odesílatel, předmět, podrobnější popis chyby, který byl zapsán přímo v aplikaci, a jako příloha je soubor ve formátu zip obsahující logy z předchozích dvou dnů používání aplikace.



Obrázek 38: O aplikaci a Chybové hlášení
Zdroj: iniAPKA

6.14 Logování, testování a verzování

K logování je využita knihovna MetroLog. Zaznamenávají se zatím pouze chyby, ke kterým v aplikaci dojde. Zapisují se do souboru s názvem Log – YYYYMMDD.log. Ukládají se logy ze dvou dnů používání aplikace. V případě odesílání chybového hlášení se tyto soubory zazipují do souboru s názvem log.

```
43|2023-03-13T09:39:24.2066206+00:00|ERROR|20|BaseService|ConnectionService.ReadDeviceSnapshot() -
Nepovedlo se získat data z KS. Chyba: Cannot access a disposed object.
Object name: 'GrpcCall'.
46|2023-03-13T09:40:20.7433979+00:00|ERROR|26|BaseService|ConnectionService.ReadDeviceSnapshot() -
Nepovedlo se získat data z KS. Chyba: One or more errors occurred. (Can't read the next message because the previous read is still in progress.)
```

Obrázek 39: Ukázka záznamu v logovacím souboru
Zdroj: iniAPKA

IniAPKU testuji sama ihned po nějaké menší nebo větší úpravě. Dále je manuálně testována společností INISOFT s.r.o. dle předem připravených scénářů.

Kód iniAPKY je postupně verzován pomocí gitu na GitHubu. V projektu jsou vytvořeny dvě hlavní větve: devel a master. Ve větvi devel je aktuálně vyvíjený kód, který ještě není schválen a otestován. V master větvi je kód aplikace, která je nahrána na Google Play.

6.15 Možné scénáře výměny dat iniAPKY s INI SW

První iterace iniAPKY vychází z ideálního stavu, ve kterém je ke komunikačnímu serveru připojeno pouze jedno zařízení s nainstalovanou aplikací, vše je správně nakonfigurováno a nedochází ve stejném okamžiku k úpravě vážních lístků (tj. v INI SW a zároveň v aplikaci).

V budoucích iteracích vývoje iniAPKY bude potřeba vymyslet způsob, jakým se bude s daty nakládat v některých případech, ke kterým bude v průběhu provozního užívání iniAPKY docházet poměrně často.

Prvním případem může být to, že aplikace nebude připojena ke KS a uživatel bude chtít data odeslat do INI SW. V aktuální verzi iniAPKY existují dva způsoby, jak data do INI SW odeslat: odesláním nově vytvořeného vážního lístku a kliknutím na ikonu šipky nahoru v horní liště v sekci Vaše záznamy. Bylo by tedy velmi nekomfortní, kdyby aplikace uživatele při ukládání nového vážního lístku neustále upozorňovala na to, že se jeho záznam nepovedlo odeslat do IS ENVITY. Řešením by mohl být indikátor stavu připojení, který by byl viditelný na každé stránce aplikace, na které může uživatel provádět akce spojené s potřebou připojení ke KS, anebo by tyto akce mohly být zašedlé, a tím pádem by na ně uživatel neměl možnost kliknout.

Druhým případem je stav, kdy uživatel dostane v aplikaci vážní lístky z INI SW, přeruší se spojení aplikace s KS, a tyto vážní lístky jsou upravovány v iniAPCE a zároveň v INI SW. Problémem je tedy rozhodnout, jaká data při budoucí vzájemné výměně mezi iniAPKOU a INI SW vzít, aby nebyla narušena jejich integrita, popř. aby o ně uživatel nepřišel, a jakým způsobem to provádět. Řešením by mohlo být to, aby uživatel sám vybral to, co chce ponechat. Pokud se iniAPKA nebo INI SW dotáží KS na data, tak se v případě shodného vážního lístku (záznamy se shodným ID z INI databáze v odpovědi od KS a v lokální databázi) s různými časovými značkami (tzv. timestampy) zobrazí dialogové okno vyzývající uživatele, aby vybral, jaký vážní

lístek se má uložit, a jaký zahodit. Bylo by dobré, aby uživatel viděl oba vyplněné vážní lístky, aby věděl, co je kde navíc, a dle toho se mohl rozhodnout.

Podobná situace jako ta předchozí nastává, pokud je v praxi využíváno více zařízení s iniAPKOU. To se dá řešit téměř stejně, až na to, že někdy těch verzí vážních lístků může být více. V aplikaci, kde je potřeba rychlého zaznamenávání dat, by bylo celkem nepříjemné zdlouhavě porovnávat a vybírat jednu platnou verzi jednoho vážního lístku. V tom případě by měl mít možnost uživatel v nastavení aplikace vypnout tuto funkci s upozorněním, že všechny jeho záznamy, které získal z IS ENVITY a upravoval je, mohou být nahrazeny novějšími. Nemůže se tedy stát, že by přišel o své původní uložené vážní lístky.

6.16 Komunikace v týmu

První schůzka se společností INISOFT s.r.o. se uskutečnila 10. října 2022, kdy bylo dohodnuto zahájení vývoje iniAPKY. Další schůzky se konaly vždy každý týden, neboť byl souběžně s vývojem iniAPKY upravován i INI SW a KS pro podporu společné výměny dat, a tudíž bylo zapotřebí dohodnout se, co a jak má fungovat.

Shrnutí toho, co se na schůzkách řešilo, a přehled úkolů na další týden, bylo pravidelně zapisováno do aplikace Confluence od společnosti Atlassian. Dále jsme si v Confluence sdíleli potřebné informace, např. stav vývoje, popis potřebných tříd ke komunikaci, nalezené chyby atd.

V případě problémů nebo potřeb aktualizací INI SW probíhala komunikace prostřednictvím aplikací Skype a AnyDesk.

Na obrázku níže je příklad zápisu ze schůzky ze 14. února 2023. V hlavičce zápisu jsou obecné informace o schůzce, tzn. datum, způsob jednání, kdo se jí ne/účastnil, autor zápisu a stav uskutečnění schůzky. V sekci Program jsou uvedeny body, které se mají na schůzce probrat. V sekci Poznámky jsou sepsány věci, které se na schůzce řešily (aktuální stav a plán vývoje, překážky, nápady, termíny atd.).

2023-02-14 - Zápis ke konzultaci vývoje

Vytvořil(a) František Tomeš, naposledy změnil(a) 20.02.2023, viewed 20 times

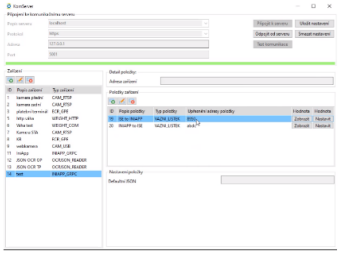
Zápis z jednání

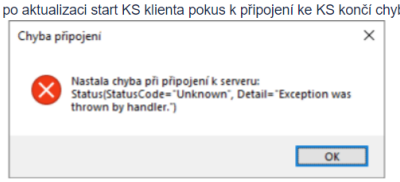
Téma	**Konzultace vývoje**
Datum	14.2.2023
Způsob jednání	FYZICKY
Účastníci	@Vojtěch Starý , @Michal Štrick , @František Tomeš , @Michaela Grusmanová
Nepřítomní / Na vědomí	@Pavel Šír
Zapsal	@František Tomeš
Stav	REALIZOVÁNO

Program

- plán vývoje s ohledem na úpravy na straně INISOFT
- návrhy GRU pro rozšíření teoretické i praktické části
- plán další komunikace a osobní schůzky
- instalace ISE - v některém z dalších termínů

Poznámky

- plán vývoje s ohledem na úpravy na straně INISOFT
  - instalace KS 34386
  - pomocí klienta vytvořit nové zařízení iniapp
  - přidat položky vážní lístky, dvě položky iniapp (směr do envita a zpět)
  - 
  - KS bude odesílat json namísto ENVITA - vážní lístky příklad json (Jira komserver-27)
  - do položky envita-json vyplnit json dle vzoru, aby nám ho posílal
  - aplikace si musí zapamatovat id zařízení a položek
  - vyxvětlení toku dat
  - načíst json
    - umět ho zapsat do logu
    - načíst do objektu
    - příp. načíst do formluáře
  - STR doplní kuchařku na snapshot, deserializaci dat pro objekt Kuchařka pro získání snapshotu
  - v další fázi se bude řešit odeslání dat z formuláře přes json do KS (opačný směr)
- návrhy GRU pro rozšíření teoretické i praktické části
- plán další komunikace a osobní schůzky
  - 16.2.2023 update po skype
- instalace ISE - v některém z dalších termínů
- aktualizace a konfigurace KS - dnes
  - po aktualizaci start KS klienta pokus k připojení ke KS končí chybou



Obrázek 40: Ukázka zápisu ze schůzky konané dne 14. února 2023
Zdroj: https://confluence.inisoft.cz

6.17 Distribuce iniAPKY

Nejrychlejším a nejsnadnějším způsobem distribuce iniAPKY mezi uživatele je její umístění na Google Play.

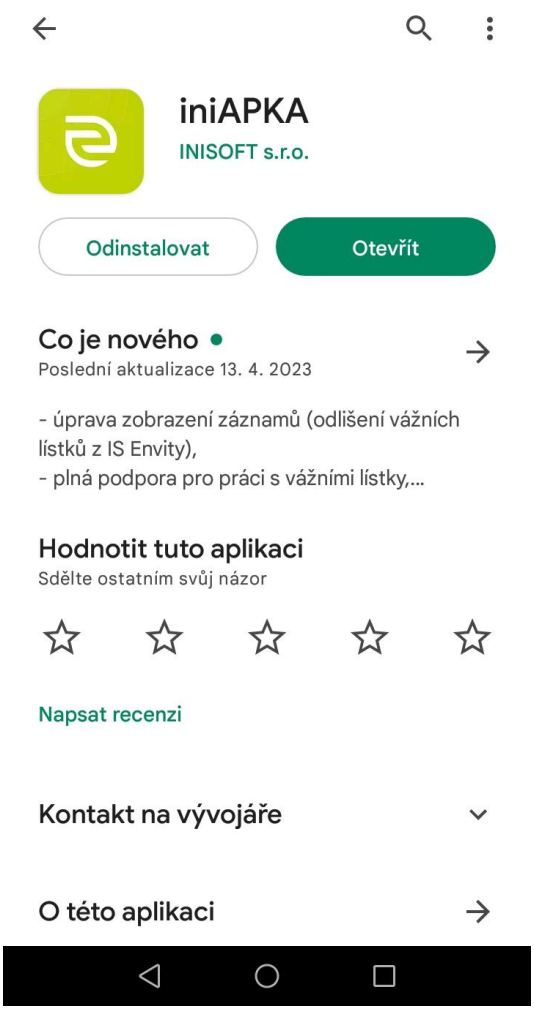
Pro publikování aplikace na toto místo jsem si musela založit vývojářský účet, za který jsem zaplatila jednorázový poplatek ve výši 25 amerických dolarů, tedy 583 korun českých (ke dni 20. března 2023, kurz Komerční banky).

Dále společnost Google musela ověřit mou identitu, což trvalo cca den a půl. Musela jsem zadat své jméno a příjmení, datum narození, bydliště, e-mailovou adresu, telefonní číslo a vyfocený nebo naskenovaný obrázek občanského průkazu.

Ve službě Google Play Console jsem vytvořila novou aplikaci, kde jsem uvedla její název, výchozí jazyk, informaci, jestli se jedná o hru nebo aplikaci, a také informaci o tom, jestli je aplikace placená nebo zdarma, potvrdit, že aplikace splňuje programové zásady pro vývojáře, a přijmout vývozní zákony Spojených států amerických. To ale nebylo všechno. Musela jsem dále upřesnit např. to, jak se k aplikaci dá přistupovat (přihlašování, potřeba specifického hardwaru atd.), jestli aplikace obsahuje reklamy nebo citlivý obsah, je vhodná pro děti, sleduje kontakty kvůli pandemii COVID-19, shromažďuje a sbírá data od uživatele, přidat zásady ochrany osobních údajů (odkaz: https://docs.google.com/document/d/1x26XcSNTBPzaQHc7ns1rVAOXNZKRp2nl), ikonu aplikace, grafiku (tzn. alespoň dva snímky toho, jak aplikace vypadá), krátký a dlouhý popis, vybrat země, ve kterých má být aplikace dostupná ke stažení, vložit podepsanou aplikaci s příponou .aab.

Byl to poměrně zdlouhavý proces. Společnost Google cca po třech dnech schválila vydání iniAPKY, takže aplikace je nyní v České republice dostupná ke stažení na Google Play úplně všem.

S vydáváním nové verze aplikace už je to mnohem jednodušší. Stačí nahrát podepsanou aplikaci s příponou .aab s vyšším číslem verze, než tomu bylo u předchozí verze aplikace. Zkontrolování a publikování této nové verze aplikace trvá většinou jeden den.

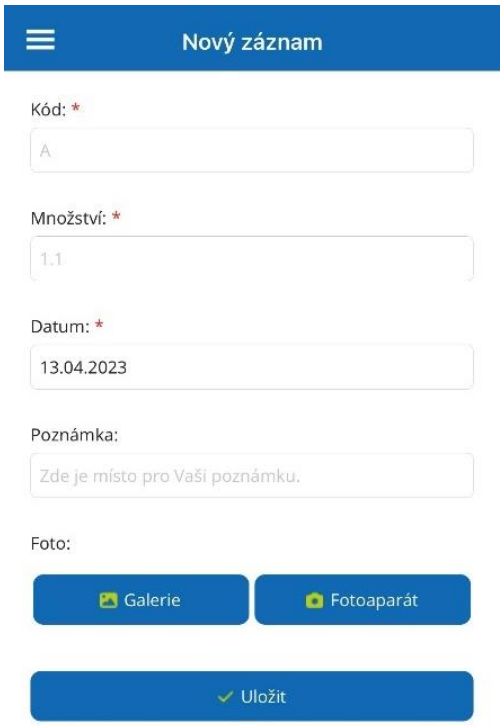


Obrázek 41: Detail iniAPKY na Google Play
Zdroj: Google Play

6.18 Multiplatformní aplikace

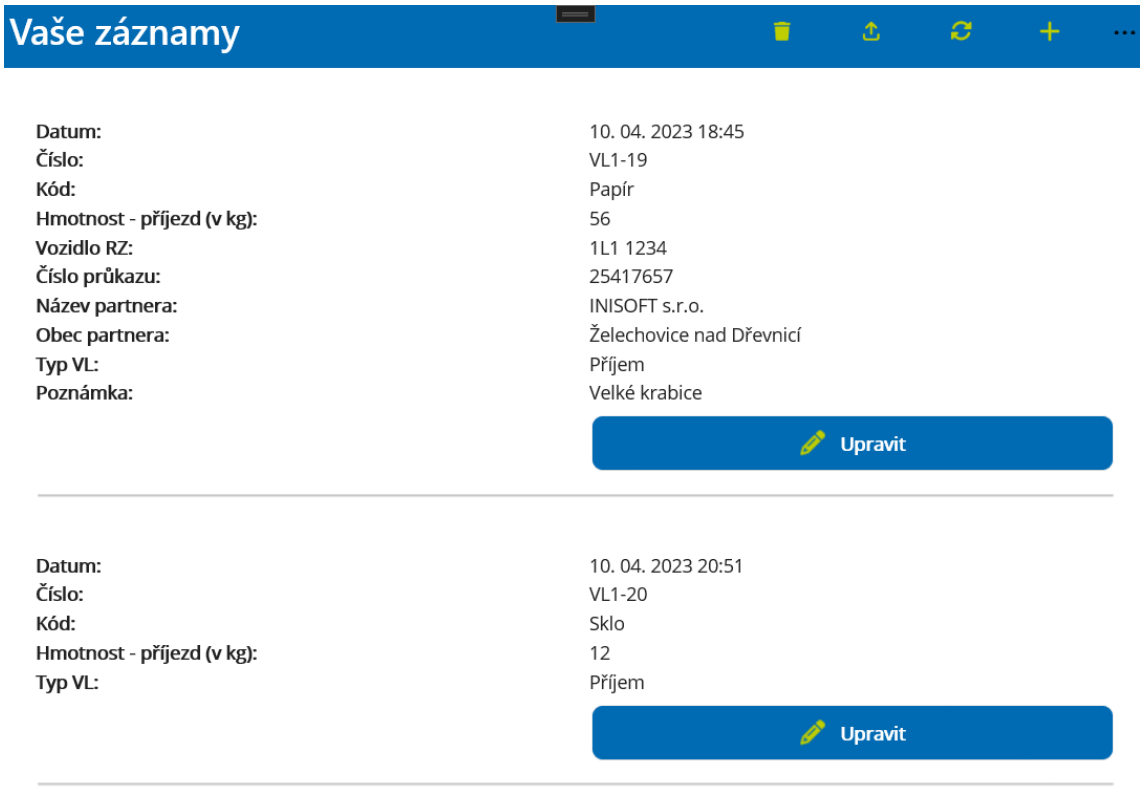
Jelikož je .NET MAUI framework pro vytváření multiplatformních aplikací, tak tato kapitola bude pojednávat o tom, jak tomu je v případě iniAPKY.

Některé stránky se v iniAPCE na iOSu zobrazuji jinak než na Androidu, a dokonce i některé funkce nefungují tak, jak by měly. Možná je to z toho důvodu, že iniAPKA byla primárně vyvíjena pro Android, laděna na něm, a tím pádem tedy chování aplikace není jako ušité na míru pro iOS. Bylo by zajímavé zjistit, jak by se aplikace chovala na Androidu, pokud by primárně byla vyvíjena pro iOS.



Obrázek 42: Snímek z iniAPKY na iOSu
Zdroj: iniAPKA

Co se týče zobrazení a funkčnosti iniAPKY na operačním systému Windows, tak v tomto případě se iniAPKA chová téměř podobně jako na Androidu. Tím pádem by se iniAPKA dala rozšířit v případě potřeby i na OS Windows.



Obrázek 43: Snímek z iniAPKY na OS Windows
Zdroj: iniAPKA

6.19 Náměty na budoucí vylepšení

Jelikož v průběhu vývoje postupně vznikají i náměty na úpravy iniAPKY v budoucích verzích, bude tato kapitola věnována právě výčtu těchto možných změn.

6.19.1 Připojit se

Na stránce Připojit se by se měla dořešit otázka přihlašování uživatelů – zadávání přihlašovacího jména a hesla. Jestli je to v nynějších verzích iniAPKY nutné, nebo by bylo lepší tato pole odebrat do doby, než bude rozhodnuto i o podpoře autentizace uživatelů.

6.19.2 Vaše záznamy

V případě stránky Vaše záznamy by vylepšením mohla být úprava zobrazení položek v seznamu. A to tak, že by bylo předem pevně definováno to, jaká data se mají

zobrazovat. Nyní se zobrazí všechna data, což způsobí to, že nějaký záznam je kratší a některý delší, někdy se zobrazí dvě tlačítka pro vymazání záznamu nebo úpravu, a někdy pouze jedno tlačítko pro úpravu záznamu atd. Pevný předpis zobrazení dat by vyřešil tento problém. Zobrazovat by se mohlo pouze např. datum, číslo, kód odpadu a tlačítko Detail. Po kliknutí na toto tlačítko by se otevřela stránka s detailem daného vážního lístku, ve kterém by bylo možné vidět všechna uložená data. Na této stránce by se uživatel mohl také rozhodnout, jestli chce daný vážní lístek upravit, odeslat, popřípadě odstranit. Z toho vyplývá i přidání podpory pro odstranění vážních lístků i z INI databáze.

Dále by do seznamu záznamů mohly být přidány záložky pro oddělení typů vážních lístků (příjem, výdej, externí vážení) nebo přidání filtru pro výběr pouze některých vážních lístků (dle data, typu, názvu odpadu atd.).

Pokud chce uživatel smazat všechny záznamy, tak se mu ukáže dialogové okno s otázkou, jestli si je vymazáním všech dat jistý. Totéž by se mělo přidat i pro odeslání a získání všech vážních lístků kvůli např. nechtěnému výmazu, odeslání či přepsání všech existujících dat.

6.19.3 Nový záznam a úprava záznamu

U této stránky by se při odesílání nebo ukládání vyplněného formuláře mohl objevit ukazatel průběhu vykonávání dané akce ve formě např. loadovacího kolečka.

Jelikož si nyní uživatel při zakládání nového záznamu nebo editaci již existujícího vážního lístku může vybrat pouze datum, bylo by pěkné, přidat i výběr času.

Dále by iniAPKA mohla získávat od KS vytvořené číselníky z některého INI SW nebo nechat uživatele si tyto číselníky vytvořit přímo v iniAPCE. To by uživatelům mohlo ušetřit mnoho práce při používání iniAPKY (např. nemuseli by několikrát ručně psát stejného partnera, RZ vozidla nebo jiné údaje).

Přidání podpory pro vytváření vážních lístků typu Výdej a Externí vážení (v současné verzi lze vytvářet pouze vážní lístky typu Příjem).

Do formuláře by se mohla přidat pole pro zadání Hmotnost – příjezd a Hmotnost – odjezd. IniAPKA by si tím pádem musela hlídat, aby znemožnila zadání Hmotnost – netto v případě zadávání některé z výše uvedených hmotností, a aby automaticky Hmotnost – netto z těchto dvou hmotností dopočítala. A obráceně, aby když bude uživatel zadávat Hmotnost – netto, tak aby nemohl zadat Hmotnost – příjezd a Hmotnost – Odjezd. Se zadáváním hmotnosti souvisí také myšlenka toho, aby si uživatel mohl vybrat ze tří možností a to, jestli chce zadávat hmotnost v gramech, kilogramech nebo v tunách.

V současné verzi iniAPKY uživatel může vložit pouze jednu fotografii. Je ale v plánu přidat podporu pro vkládání více obrázků.

6.19.4 Odeslání chybového hlášení

Jelikož se objevily problémy s otevíráním e-mailového klienta (systémové aplikace i aplikace třetí strany) při odesílání chybového hlášení, bylo by dobré promyslet zabudování přímého odeslání chybového hlášení do samotné aplikace tak, aby otevírání e-mailového klienta nebylo potřeba.

6.19.5 Změny pro celou aplikaci

IniAPKA sice nějakou podporu (nastavení barev) pro tmavý režim systému má, ale chtělo by to se trochu zamyslet a nastavit lepší barvy, projít možnosti zobrazení obrázků (podpora alfa kanálu formátu PNG) atd.

Jelikož je iniAPKA předpřipravena pro podporu i jiného jazyka, přidat nastavení i jiného jazyka, než češtiny by nebylo od věci. Společnost INISOFT s.r.o. působí i na Slovensku, takže pro potencionální slovenské uživatele by to určitě bylo velmi příjemné. A kromě slovenštiny by se hodil i univerzální anglický jazyk.

Jelikož iniAPKA je multiplatformní aplikace, byla by škoda, kdyby se nerozšířila i pro iOS. V plánu tedy je odladit aplikaci tak, aby se chovala bezproblémově i na tomto operačním systému.

Do aplikace by mohla být přidána nápověda, která by uživatele seznámila s používáním iniAPKY.

Při testování aplikace, která měla nastavený Target Android Framework na 33 (aktuálně nejnovější verzi), se u některých novějších zařízení s Androidem 13 vyskytoval problém s otevíráním kamery pro pořízení snímku u zakládání nového vážního lístku. Dočasným řešením je snížení této verze Target Android Frameworku na verzi 31. Důležité je sledovat vývoj této chyby a kód opravit tak, aby fungoval na co nejnovější verzi Target Android Frameworku.

6.19.6 Zdrojový kód

Z hlediska kódu by se některé jeho části mohly zjednodušit nebo optimalizovat (např. zredukovat velké množství používání bloků try – catch, což je náročná operace při běhu aplikace a zbytečně ji může zpomalovat) a mohlo by být přidáno logování na více míst. Určitě by také nebylo špatné, kdyby si zdrojový kód iniAPKY prohlédl i jiný vývojář, protože autorská slepota a přesvědčení o správnosti implementace kódu z mých dosavadních zkušeností existuje. Tím chci říct, že na samotném zdrojovém kódu iniAPKY je určitě na čem pracovat a neustále ho vylepšovat.

7 Zhodnocení vývoje

Framework .NET MAUI je dle mého názoru dobrou volbou pro tvorbu multiplatformních aplikací. A to jak z hlediska velkého množství dostupných knihoven, tak i z hlediska obrovské existující komunity. Každopádně některé věci je nutné pro danou platformu trochu poupravit.

Definice layoutu pomocí jazyka XAML je velmi jednoduchou formou pro dosažení potřebného rozložení jednotlivých grafických elementů.

Komunikace pomocí protokolu gRPC nebyla ze začátku pro mě úplně nejsnadnější oproti třeba komunikaci pomocí REST API. Šifrované připojení s komunikačním serverem nebylo úplně bezproblémové a jeho zprovoznění zabralo poměrně dlouhou dobu. Muselo se provést pár úprav na straně serveru i iniAPKY.

IniAPKA, i když se jedná o nově vznikající aplikaci, tak si již stačila projít poměrně velkou řadou vývojových iterací, ve kterých byla obohacena o nové funkce nebo naopak byly původně zamýšlené funkce odebrány, a to samé platí i o grafickém rozhraní.

Práce v týmu složeném speciálně pro vývoj iniAPKY byla pro mě přínosná a příjemná. Zjistila jsem, jak důležité je projektové řízení, díky kterému bylo vše včas zařízeno, na schůzkách byly přítomny osoby, jejichž znalosti nebo práce byla pro danou fázi vývoje potřeba, a to hlavní, že se iniAPKU povedlo vytvořit a dovést ji do požadovaného rozsahu v daném termínu.

Ze začátku jsem si zapisovala dobu strávenou nad vývojem, ale po čase jsem přestala. Tuhle věc bych teď už udělala jinak, a určitě bych si hodiny strávené nad vývojem iniAPKY zapisovala, neboť by to byla určitě zajímavá informace. Jak pro mě, abych věděla, co od sebe můžu očekávat, tak pro nějakou podkapitolu této diplomové práce o tom, jaká část vývoje zabrala nejvíce času a jaká nejméně.

V průběhu vývoje jsem narazila na poměrně dost překážek, které jsem buď sama po delším zkoumání a bádání, anebo optání se přátel či pomoci ze strany společnosti INISOFT s.r.o. dokázala překonat.

Je zde hodně možných cest, kterými se vydat dál, a možností, kterými lze iniAPKU dále vylepšovat.

Závěr

Mezi nejznámější a nejpoužívanější mobilní operační systémy patří iOS a Android, přičemž více používaný z těchto dvou je v současné době Android.

Mobilní aplikace lze dle typu jejich vývoje rozdělit na nativní, multiplatformní, webové a hybridní. Nativní jsou vyvíjeny pro konkrétní mobilní operační systém v daném programovacím jazyce (např. Kotlin pro Android). Multiplatformní, webové a hybridní umožňují vývoj aplikací pro více operačních systémů najednou a v různých programovacích jazycích. Funkčnost webových a hybridních aplikací je závislá na připojení k internetu, ale zase je lze nejrychleji aktualizovat.

Technologií pro vývoj mobilních aplikací je celá řada. Mezi nejznámější a nejpoužívanější patří .NET (.NET MAUI), React Native, Flutter, Apache Cordova, Ionic atd. Každý z nich má své plusy a mínusy, a je nutné zapřemýšlet nad tím, který z nich je pro vývoj dané mobilní aplikace nejvhodnější.

Po analýze požadavků a dohodě se společností INISOFT s.r.o. byl z těchto technologií vybrán framework .NET MAUI, ve kterém byla vytvořena multiplatformní aplikace s názvem iniAPKA, která umožňuje pracovat s vážními lístky a vyměňovat si je s ostatními programy od společnost INISOFT s.r.o. prostřednictvím gRPC a komunikačního serveru.

IniAPKA je určena hlavně pro OS Android, funguje ale i na operačním systému Windows a zčásti na operačním systému iOS. Pro plnou podporu funkčnosti na iOS by bylo potřeba některé části iniAPKY poupravit.

Pro zprovoznění iniAPKY je potřeba si aplikaci stáhnout z Google Play na zařízení s Androidem, mít k dispozici komunikační server, službu pro komunikaci komunikačního serveru s databází, tuto databázi a případně některý z programů od společnosti INISOFT s.r.o.

Ovládání iniAPKY je intuitivní a jednoduché. V případě provedení chybné operace, je uživatel upozorněn dialogovým oknem s hláškou o tom, co je špatně a co je potřeba udělat jinak.

IniAPKA umožňuje práci ve dvou režimech: připojeno a nepřipojeno ke komunikačnímu serveru. Pokud uživatel není připojen ke komunikačnímu serveru, může si pouze vytvářet vážní lístky, upravovat si je, mazat a prohlížet. Pokud je ale iniAPKA připojena ke komunikačnímu serveru, můžou být vytvořené vážní lístky odeslány a uloženy pomocí komunikačního serveru a IS ENVITA služby do databáze, a následně prohlíženy v některém z programů od společnosti INISOFT s.r.o. Komunikace je možná i obráceně, kdy iniAPKA může přijmout již existující vážní lístky od komunikačního serveru.

Vážní lístek založený v iniAPCE může zatím obsahovat pouze některé údaje oproti vážnímu lístku vytvořeném z některého programu od společnosti INISOFT s.r.o. Do vážního lístku lze zadat v iniAPCE následující údaje: datum, kód odpadu, čistou hmotnost odpadu v kilogramech, RZ vozidla, číslo průkazu, poznámku a fotografii. V budoucích verzích iniAPKY je ale v plánu přidat podporu pro zadání téměř všech dat jako je tomu u již zmíněných programů od společnost INISOFT s.r.o., ale to ukáže až čas a připomínky nebo poznámky od budoucích uživatelů iniAPKY.

V iniAPCE je možné odeslat chybové hlášení pomocí e-mailového klienta (některé z aplikací pro odesílání e-mailů). Soubor, který je k e-mailu vytvořeném pro odeslání chybového hlášení přibalen, je postupně vytvářen logováním chyb pomocí knihovny MetroLog. Veškerá potřebná nastavení pro odeslání chybového hlášení, např. komu se má hlášení odeslat, jsou definována v souboru appsettings.json.

Kód je strukturován dle architektonického vzoru MVVM a je po celou dobu vývoje ukládán na GitHub. Ve větvi devel je aktuálně rozpracovaný kód, který ještě neprošel testováním společností INISOFT s.r.o. dle předem připravených scénářů. Ve větvi master je kód aplikace, která je nahrána na Google Play.

IniAPKA najde své uplatnění na místech, na kterých se produkuje nebo se nějakým způsobem nakládá s odpady (např. na sběrných dvorech atd.) s podmínkou toho, že mají na těchto místech zakoupen některý z programů od společnosti INISOFT s.r.o. (např. IS ENVITA, SKLAD Odpadů 8 atd.).

Veškerá komunikace v průběhu vývoje byla zadávána do aplikace Confluence, kde je zaznamenán tedy i postup vývoje (projektová timeline) – plán, termíny schůzek, úkoly, nápady nebo problémy.

V průběhu vytváření iniAPKY se střídaly těžké chvíle plné demotivace (např. při hledání cesty pro podporu šifrované komunikace mezi iniAPKOU a komunikačním serverem prostřednictvím gRPC či v průběhu vytváření spustitelné aplikace) s chvílemi radosti a naplnění (např. když tyto překážky byly překonány nebo ve chvíli, kdy iniAPKA splnila všechny požadavky definované při zahájení vývoje). S dokončením diplomové práce ale vývoj iniAPKY nekončí. Se společností INISOFT s.r.o. je totiž dohodnuto pokračování vývoje a budoucí spolupráce na iniAPCE.

Seznam použité literatury

ANDROID DEVELOPERS, 2023. Platform Architecture. *Android Developers* [online] [vid. 2023-04-15]. Dostupné z: https://developer.android.com/guide/platform

APACHE CORDOVA, 2023. Architectural overview of Cordova platform - Apache Cordova. *Apache Cordova* [online] [vid. 2023-02-07]. Dostupné z: https://cordova.apache.org/docs/en/11.x/guide/overview/index.html

AZURE MICROSOFT, 2022. Co je vývoj mobilních aplikací? Naučte se vytvářet aplikace | Microsoft Azure. *Azure* [online] [vid. 2022-12-31]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-mobile-app-development/

BENNETT, Jim, 2018. *Xamarin in action: creating native cross-platform mobile apps*. Shelter Island: Manning. ISBN 978-1-61729-438-9.

BRITCH, David, 2023. Deployment - .NET MAUI. *Microsoft* [online] [vid. 2023-02-01]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/maui/deployment/

CARTHERN, Chris, William WILSON, Noel RIVERA a Richard BEDWELL, 2015. *Cisco networks: engineers handbook of routing, switching, and security with IOS, NX-OS and ASA*. First edition. New York: Apress. The expert's voice in networking. ISBN 978-1-4842-0860-1.

DOUGLAS, Jon, 2022. Co je NuGet a co to dělá? *Microsoft* [online] [vid. 2023-01-08]. Dostupné z: https://learn.microsoft.com/cs-cz/nuget/what-is-nuget

EISENMAN, Bonnie, 2015. *Learning React Native: building mobile applications with JavaScript*. First edition. Beijing: O'Reilly. ISBN 978-1-4919-2900-1.

FEILER, Jesse, 2015. *Introducing SQLite for mobile developers*. New York, NY? Apress. The expert's voice in mobile programming. ISBN 978-1-4842-1765-8.

FLUTTER, 2023. FAQ. *Flutter* [online] [vid. 2023-02-06]. Dostupné z: https://docs.flutter.dev/resources/faq

GEORGE, Andy a David BRITCH, 2023. Tutorial: Create a .NET MAUI app - .NET MAUI. *Microsoft* [online] [vid. 2023-01-30]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/maui/tutorials/notes-app/

HALF, Robert, 2023. 6 Basic SDLC Methodologies: Which One Is Best? *RobertHalf* [online] [vid. 2023-01-29]. Dostupné z: https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best

HOLEC, Miroslav, 2022a. gRPC | Historie gRPC a .NET | Školení. *Miroslav Holec* [online] [vid. 2022-10-19]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-historie

HOLEC, Miroslav, 2022b. gRPC | Komunikace v gRPC | Školení. *Miroslav Holec* [online] [vid. 2022-10-20]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-komunikace

HOLEC, Miroslav, 2022c. gRPC | Návrh gRPC služeb | Školení. *Miroslav Holec* [online] [vid. 2022-10-20]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-navrh-sluzeb

HOLEC, Miroslav, 2022d. gRPC | Protobuf | Školení. *Miroslav Holec* [online] [vid. 2022-10-20]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-protobuf

HOLEC, Miroslav, 2022e. gRPC | Reflection | Školení. *Miroslav Holec* [online] [vid. 2022-10-20]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-reflection

HOLEC, Miroslav, 2022f. gRPC | Úvod do gRPC | Školení. *Miroslav Holec* [online] [vid. 2022-10-16]. Dostupné z: https://www.miroslavholec.cz/blog/grpc-uvod

HUGHES, Lawrence E., 2022. *Pro active directory certificate services: creating and managing digital certificates for use in Microsoft networks*. New York, NY: Apress. ISBN 978-1-4842-7486-6.

CHACON, Scott, 2014. *Pro Git*. Second edition. New York, NY: Apress. The expert's voice in software development. ISBN 978-1-4842-0077-3.

CHENG, Fu, 2018. *Build Mobile Apps with Ionic 4 and Firebase: Hybrid Mobile App Development* [online]. 2nd ed. 2018. Berkeley, CA: Apress : Imprint: Apress. ISBN 978-1-4842-3775-5. Dostupné z: doi:10.1007/978-1-4842-3775-5

IBM, 2022. What is Software Testing and How Does it Work? | IBM. *IBM* [online] [vid. 2023-01-26]. Dostupné z: https://www.ibm.com/topics/software-testing

IONIC, 2023. Open-Source UI Toolkit to Create Your Own Mobile or Desktop Apps | Ionic Documentation. *Ionic Framework Docs* [online] [vid. 2023-02-07]. Dostupné z: https://ionicframework.com/docs/v5

KITNER, Radek, 2017. Typy testování software. *Radek Kitner* [online] [vid. 2023-01-29]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/

KOTLIN, 2022. Get started with Kotlin Multiplatform Mobile | Kotlin. *Kotlin Help* [online] [vid. 2023-01-01]. Dostupné z: https://kotlinlang.org/docsmultiplatform-mobile-getting-started.html

LACKO, Ľuboslav, 2018. *Vývoj aplikací pro iOS*. 1. vydání. Přel. Martin HERODEK. Brno: Computer Press. ISBN 978-80-251-4942-3.

LAURENCE, Laurence, Pierre-Olivier, 2021. *PROGRAMMING ANDROID WITH KOTLIN achieving structured concurrency with coroutines*. S.l.: O'REILLY MEDIA. ISBN 978-1-4920-6295-0.

LUTKEVICH, Ben, 2022. What is Software Documentation? Definition, Types and Examples. *Software Quality* [online] [vid. 2023-02-07]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/documentation

META, 2021. *The Facebook Company Is Now Meta | Meta* [online] [vid. 2023-04-20]. Dostupné z: https://about.fb.com/news/2021/10/facebook-company-is-now-meta/

MICROSOFT, 2021. What is Xamarin.Forms? - Xamarin. *Microsoft* [online] [vid. 2022-10-16]. Dostupné z: https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms

MICROSOFT, 2022a. | nástroje MSBuild Microsoft Docs - MSBuild. *Microsoft* [online] [vid. 2022-10-16]. Dostupné z: https://learn.microsoft.com/cs-cz/visualstudio/msbuild/msbuild

MICROSOFT, 2022b. Co je .NET MAUI? - .NET MAUI. *Microsoft* [online] [vid. 2022-10-16]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/maui/what-is-maui

MICROSOFT, 2022c. .NET (a .NET Core) – úvod a přehled. *Microsoft* [online] [vid. 2022-10-16]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/core/introduction

MICROSOFT, 2022d. Xamarin official support policy | .NET. *Microsoft* [online] [vid. 2022-10-10]. Dostupné z: https://dotnet.microsoft.com/en-us/platform/support/policy/xamarin

PATADIYA, Jaydeep, 2022. Web vs. Native vs. Hybrid Apps: Which is Better? *Radix* [online] [vid. 2022-12-30]. Dostupné z: https://radixweb.com/blog/web-vs-native-vs-hybrid-application

PATHAK, Nishith a Anurag BHANDARI, 2018. *IoT, AI, and Blockchain for .NET: Building a Next-Generation Application from the Ground Up* [online]. 1st ed. 2018. Berkeley, CA: Apress : Imprint: Apress. ISBN 978-1-4842-3709-0. Dostupné z: doi:10.1007/978-1-4842-3709-0

PETZOLD, Charles, 2015. *Creating Mobile Apps with Xamarin.Forms*. Washington: Microsoft Press. ISBN 978-1-5093-0297-0.

PRICE, Mark J., 2022. *C# 11 and .NET 7 - modern cross-platform development: start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7*. Seventh edition. Birmingham: Packt Publishing. ISBN 978-1-80324-895-0.

RAJ, Pethuru, Anupama C. RAMAN a Harihara SUBRAMANIAN, 2017. *Architectural patterns: uncover essential patterns in the most indispensable realm of enterprise architecture*. Birmingham, UK: Packt Publishing. ISBN 978-1-78728-749-5.

ROSE, Robert F., 2022. *Software development activity cycles: collaborative development, continuous testing and user acceptance*. New York, NY: Apress. ISBN 978-1-4842-8239-7.

SASEENDRAN, Sarathlal, 2022. Create An Android App With .NET MAUI And Visual Studio 2022. *C# Corner* [online] [vid. 2023-02-01]. Dostupné z: https://www.c-sharpcorner.com/article/create-an-android-app-with-net-maui-and-visual-studio-2022/

SEIDL, Martina, 2015. *UML @ classroom: an introduction to object-oriented modeling*. New York, NY: Springer Berlin Heidelberg. ISBN 978-3-319-12741-5.

SPENCER, Jamie, 2022. The Complete History of iOS - From Version 1.0 to 16.0 - Practically Networked. *Practically networked* [online]. [vid. 2023-04-15]. Dostupné z: https://www.practicallynetworked.com/ios-history/

STONIS, Michael, 2022. *Enterprise Application Patterns using .NET MAUI* [online]. 2022. B.m.: Microsoft Developer Division, .NET, and Visual Studio product teams. Dostupné z: https://raw.githubusercontent.com/dotnet-architecture/eBooks/main/current/maui/Enterprise-Application-Patterns-Using-.NET-MAUI.pdf

TEHERAN, Miguel, 2022. Creating Your First Application For MacOS Using .NET MAUI. *C# Corner* [online] [vid. 2023-02-01]. Dostupné z: https://www.c-sharpcorner.com/article/crreating-your-first-application-with-macos-using-net-maui/

THINK EASY, 2023. Jak publikovat aplikaci na Google Play a App Store? *Think Easy s.r.o.* [online] [vid. 2023-02-08]. Dostupné z: https://thinkeasy.cz/jak-publikovat-aplikaci-na-google-play-a-app-store/

TORRE, Cesar De la, 2016. .NET Core, .NET Framework, Xamarin – The "WHAT and WHEN to use it". *Cesar de la Torre* [online] [vid. 2022-10-16]. Dostupné z: https://devblogs.microsoft.com/cesardelatorre/net-core-1-0-net-framework-xamarin-the-whatand-when-to-use-it/

VÁVRŮ, Jiří a Miroslav UJBÁNYAI, 2013. *Programujeme pro Android*. 2., rozš. vyd. Praha: Grada. ISBN 978-80-247-7983-6.

VELU, Vijay Kumar, 2016. *Mobile application penetration testing: explore real-world threat scenarios, attacks on mobile applications, and ways to counter them*. Birmingham, UK: Packt Publishing. ISBN 978-1-78588-869-4.

WAGNER, Bill, 2023. Asynchronní programování – C #. *Microsoft* [online] [vid. 2023-04-15]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/asynchronous-programming/async-scenarios

WARGO, John M., 2015. *Apache Cordova 4 programming*. Upper Saddle River, NJ: Addison-Wesley. Addison-Wesley mobile programming series. ISBN 978-0-13-404819-2.

WARGO, John M., 2020. *Learning progressive web apps: building modern web apps using service workers*. First edition. Boston: Addison-Wesley. Learning series. ISBN 978-0-13-648422-6.

WHALEN-DUNN, Kenzie, 2019. Xamarin.Forms vs Xamarin Native. *Kenzie Whalen-Dunn* [online]. [vid. 2022-10-16]. Dostupné z: https://knzwhalen.com/2019/09/13/xamarin-forms-vs-xamarin-native/

WIEGERS, Karl a Candase HOKANSON, 2023. *Software requirements essentials*. First edition. Boston: Addison-Wesley. ISBN 978-0-13-819028-6.

ZACCAGNINO, Carmine, 2020. *PROGRAMMING FLUTTER native, cross-platform apps the easy way;native, cross-platform apps the easy way*. S.l.: PRAGMATIC BOOKSHELF, THE. ISBN 978-1-68050-763-8.