Vývoj Capacitor Pluginu pro detekci textu na nativním zařízení

Petr Zámečník

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr Zámečník**
Osobní číslo:	**A20450**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Vývoj Capacitor Pluginu pro detekci textu na nativním zařízení**
Téma práce anglicky:	**Development of a Capacitor Plugin for Text Detection on Native Device**

Zásady pro vypracování

1. Popište proces vývoje mobilní aplikace pomocí frameworku Ionic/Capacitor. Také se zaměřte na výhody a nevýhody využití daného frameworku vůči jiným řešením.
2. Stručně popište technologie vhodné pro vývoj Capacitor pluginu a jeho možné využití.
3. Vypracujte funkční a nefunkční požadavky pro hlavní funkce Capacitor Pluginu a vyberte vhodné technologie pro implementaci.
4. Popište možnosti tvorby prototypu aplikace s použitím vytvořeného Capacitor Pluginu.
5. Implementujte mobilní aplikaci pomocí technologií TypeScript/Angular a frameworku Ionic. Aplikaci otestujte na simulovaném i reálném zařízení.
6. Zhodnoťte implementované řešení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. HAJIAN, Majid. Progressive web apps with Angular: create responsive, fast and reliable PWAs using Angular. [New York, NY]: Apress, [2019], 1 online zdroj. Dostupné z: doi:9781484244487
2. Ionic 6: Create awesome apps for iOS, Android, Desktop and Web. D&D Verlag Bonn, 2022. ISBN 394510257X.
3. DUNCAN, Andy. Objective-C: pocket reference. Beijing: O'Reilly, 2002, v, 122 s. ISBN 0596004230. Dostupné také z: https://digilib.k.utb.cz/handle/10563/52264
4. DORMANN, Andreas. Ionic 4+: creating awesome apps for iOS, Android, Desktop and Web. Bonn: D&D Verlag, [2019], vi, 626 s. ISBN 978-3-945102-52-7.
5. CHAUDHURI, Arindam, Krupa MANDAVIYA, Pratixa BADELIA a Soumya K. GHOSH. Optical character recognition systems for different languages with soft computing. Cham, Switzerland: Springer, 2017, 1 online resource. Studies in fuzziness and soft computing. Dostupné z: doi:9783319502526

Vedoucí bakalářské práce: **Ing. Jakub Josef Forman**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Petr Zámečník v. r.
podpis studenta

ABSTRAKT

Cílem bakalářské práce je návrh, popis a implementace Capacitor pluginu, pomocí kterého bude možné detekovat text kamerou mobilního zařízení, nebo případně z dříve zaznamenané fotografie. Obsahem práce je také zhodnocení již existujícího řešení, jenž je implementováno pomocí Cordovy a porovnání jejich kladů a záporů. Primární platformou bude iOS a Android. Implementace proběhne především pomocí jazyka TypeScript a frameworku Ionic / Capacitor, ale dostane se také na nativní jazyky pro obě platformy – Swift a Java, Kotlin. Vlastnosti pluginu budou poté otestovány na jednoduchém prototypu mobilní aplikace, která ovšem není obsahem práce.

Klíčová slova: iOS, Apple, iPhone, Android, framework, Ionic, Capacitor, Cordova, TypeScript, JavaScript, Swift, Java, Kotlin, plugin, npm, Node JS, kamera, fotografie

ABSTRACT

The focus of the bachelor thesis is the design, description, and implementation of the Capacitor plugin, which will be able to detect text from the camera of a mobile device or possibly from a previously taken image. The content of the thesis is also an evaluation of an existing solution that is implemented using Cordova and a comparison of their pros and cons. The primary platform will be iOS and Android. The implementation will primarily be done using TypeScript and the Ionic / Capacitor framework but will also use the native languages for both platforms - Swift and Java, Kotlin. The plugin features will then be tested on a simple mobile app prototype, which is not the scope of the thesis, however.

Keywords: iOS, Apple, iPhone, Android, framework, Ionic, Capacitor, Cordova, TypeScript, JavaScript, Swift, Java, Kotlin, plugin, npm, Node JS, camera, image

Rád bych poděkoval svému vedoucímu bakalářské práce, panu Ing. Jakubu Josefu Formanovi, za odborné vedení práce a významné rady, které přispěly k dokončení této bakalářské práce. Dále bych rád poděkoval svému team leaderovi, Janu Sedlářovi, a společnosti Papirfly za vstřícnost při plnění mých studijních povinností a podporu ve vzdělání.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V dnešní digitální éře mobilních zařízení se stává rozpoznávání textu pomocí kamery stále důležitějším nástrojem pro mnoho uživatelů. Tato bakalářská práce se zabývá návrhem, popisem a implementací Capacitor pluginu, který umožní detekovat text na fotografii pomocí kamery mobilního zařízení. Plugin také poskytne možnost extrakce textu z dříve zaznamenaných fotografií. Součástí práce je také zhodnocení existujícího řešení, které je implementováno pomocí Cordovy, a srovnání jejich výhod a nevýhod.

Vzhledem k tomu, že primárním zaměřením pluginu jsou mobilní platformy iOS a Android, bude implementace probíhat především pomocí jazyka TypeScript a frameworku Ionic / Capacitor. Každopádně, pro dosažení maximální kompatibility a výkonu, budou nástroje a technologie využívat nativních jazyků obou platforem - Swift pro iOS a Java/Kotlin pro Android. Tímto přístupem se zajistí, že plugin bude efektivně pracovat a poskytovat kvalitní výsledky na obou operačních systémech.

V rámci práce budou navrženy a implementovány klíčové vlastnosti pluginu, které budou následně otestovány na jednoduchém prototypu mobilní aplikace. Samotná aplikace není předmětem této práce, avšak její vytvoření umožní validaci funkcionality a úspěšnosti pluginu.

I. TEORETICKÁ ČÁST

1 FRAMEWORK IONIC

Ionic je populární open-source framework pro vývoj multi-platformních mobilních aplikací s využitím webových technologií jako je HTML, CSS a JavaScript a převážně také TypeScript.

Poskytuje knihovnu předdefinovaných komponent, které umožňují vývojářům vytvářet mobilní aplikace pomocí webových technologií a zároveň poskytuje vzhled a poct nativních aplikací.

Framework Ionic byl vytvořen v roce 2013 firmou Drifty Co. A mezi jeho tvůrce patří Max Lynch, Ben Sperry a Adam Bradley. První verze Ionicu byla postavena na AngularJS a Apache Cordova. Nejnovější verze však postupně přešli na moderní Angular a Capacitor. Avšak umožňují také využití technologií jako React, Vue a Svelte.

Ionic obsahuje velmi pestrou sadu předpřipravených komponent pro tvorbu uživatelského rozhraní aplikace. Komponenty si lze představit jako stavební bloky aplikace, které je možné využít opětovně.

Pro zobrazení obsahu se využívá WebView, který umožňuje vykreslování webového obsahu v nativní aplikaci. To znamená, že je možné napsat kód pouze jednou a spustit ho na jakékoliv platformě která WebView podporuje.

Díky tomu je dosaženo podpory pro různé platformy, jako jsou například web, iOS a Android. [1]

1.1 Ionic CLI

Ionic také disponuje velmi výkonným CLI pro framework Angular, který poskytuje velké množství užitečných funkcí a příkazů pro vývoj aplikace. Je možné jej využit pro tvorbu nového projektu, sestavení aplikací pro produkci nebo spuštění aplikace na emulátoru, nebo reálném zařízení. [1]

2 FRAMEWORK CAPACITOR

Capacitor byl poprvé uveden na trh v roce 2019. Jedná se o nový, převratný a inovativní způsob poskytování nativních mobilních aplikací a progresivních webových aplikací ze stejné kódové základny, a to všechno s využitím webových technologií namísto nativních programovacích jazyků, jako je například Java, Kotlin, Swift a Objective-C.

Tým, který stojí za vznikem Ionicu, vytvořil Capacitor jako duchovního nástupce Apache Cordova a Adobe PhoneGap, přičemž se inspiroval z dalších velmi populárních multiplatformních nástrojů, jako je například React Native a Turbolinks, ale zaměřil se výhradně na to, aby umožnil vývoj mobilních aplikací za pomoci moderních webových technologií.

Cordova po dlouhou dobu běžela pod kapotou téměř každé aplikace, která byla vydána pomocí technologie Ionic. Zatímco se Ionic zaměřoval primárně na uživatelské rozhraní, nativní část byla ponechána Cordově.

Ale s postupem času aspirace a vize týmu Ionic pro nativní runtime předčily to, co Cordova byla schopna nabídnout. To je z velké části způsobeno tím, že mnoho nových, moderních API nebylo dříve dostupných. Když se tým Ionicu dostal na rozcestí a rozhodli se, že by bylo lepší vydat se vlastní cestou než pokračovat v budování na základech, které Cordova již vytvořila. Bylo to těžké rozhodnutí, ale ohlížejíc se zpět, neexistují žádné pochyby o tom, že to bylo správné rozhodnutí. [2]

2.1 Vzestup multiplatformního vývoje

V současnosti se odhaduje, že přibližně 60 % všech aplikací dostupných v obchodě s aplikacemi je hybridních, což znamená, že mohou běžet na zařízeních s operačními systémy iOS i Android pomocí sdíleného kódu. To je umožněno pomocí řešení, jako je React Native, Ionic, Cordova a stále více Capacitor. [2]

Vzhledem k ohromnému množství aplikací, které jsou každý rok vydávány v obchodech s aplikacemi, a neustále rostoucí poptávce po mobilních aplikacích v rámci podniků se hybridní přístupy staly praktickým řešením pro urychlení a zjednodušení vývojového procesu. Tyto přístupy nabízejí několik výhod:

Namísto vývoje specifického kódu pro každou platformu umožňují hybridní řešení vývojářům sdílet významnou část svého kódu mezi různými platformami. Kód se napíše pouze jednou a může bezproblémově fungovat na zařízeních s operačními systémy iOS i Android. Tímto způsobem lze dosáhnout úspory času o 50 % nebo více. [2]

Hybridní vývoj zjednodušuje údržbu a aktualizace aplikací. Jelikož sdílený kód slouží jako základ pro obě platformy, jakékoli úpravy nebo vylepšení provedené na kódovém základu automaticky prospívají jak verzi aplikace pro iOS, tak pro Android. To zjednodušuje vývojový cyklus a zajišťuje konzistenci napříč platformami. [2]

Kromě toho hybridní řešení podporují opětovné použití kódu, jelikož vývojáři mohou využít jediný kódový základ pro vytváření aplikací pro různé platformy. To eliminuje potřebu psát platformě specifický kód od začátku, což snižuje vývojové úsilí a minimalizuje riziko zavedení rozdílů nebo chyb do samostatných kódových základů. [2]

Dodatečně hybridní frameworky často poskytují robustní sady nástrojů a knihoven, které usnadňují rychlý vývoj. Tyto nástroje nabízejí funkce jako předpřipravené uživatelské rozhraní, zjednodušené testování a schopnost ladění kódu. To dále urychluje vývojový proces a zvyšuje produktivitu vývojářů. [2]

Celkově řečeno, hybridní přístupy k vývoji adresují poptávku po mobilních aplikacích tím, že optimalizují efektivitu vývoje a minimalizují zdvojování práce. Umožňují vývojářům napsat kód jednou a nasadit ho na různé platformy, což vede k významným úsporám času a nákladů, zatímco udržuje konzistentní uživatelskou zkušenost na různých zařízeních. [2]

2.2 Co dělá Capacitor jedinečným?

- Moderní přístup zaměřený na web: Capacitor využívá sílu a flexibilitu webových technologií jako základ pro hybridní vývoj. Využívá standardy a technologie webu, jako jsou HTML, CSS a JavaScript, což umožňuje vývojářům využívat své stávající dovednosti a nástroje pro vývoj mobilních aplikací.

- Přístup k nativním funkcím a API: Capacitor poskytuje jednoduché a intuitivní rozhraní pro přístup k nativním SDK a nativním API na každé platformě. To znamená, že vývojáři mohou využívat zařízení funkcí a možností, jako je kamera, geolokace, kontakty, notifikace a další, pomocí známých webových vývojových technik.

- Kompatibilita s různými platformami: Capacitor umožňuje vývojářům vytvářet aplikace, které běží nativně na různých platformách, včetně iOS, Android, Electron (pro desktop) a webu (PWA). S Capacitorem mohou vývojáři psát kód jednou a nasazovat ho na různé platformy, což šetří čas a úsilí.

- Zpětná kompatibilita s Cordovou: Capacitor byl navržen tak, aby byl kompatibilní s Cordovou, což znamená, že existující Cordova pluginy lze snadno používat v projektech s Capacitorem. Tím se usnadňuje přechod z Cordovy na Capacitor a umožňuje vývojářům využít své investice do Cordova pluginů.

- Rozšiřitelnost a ekosystém pluginů: Capacitor má rozrůstající se ekosystém pluginů vyvíjených komunitou, které rozšiřují jeho funkce. Tyto pluginy poskytují další funkcionality, integraci s populárními frameworky a přístup k specifickým funkcím a službám nativních operačních systémů [2]

2.3 Capacitor Command Line Interface

V rámci Capacitoru je k dispozici nástroj Command Line Interface (CLI), který lze nainstalovat přímo na lokální úrovni pro každou aplikaci. Tato vlastnost má několik výhod. Především není zapotřebí žádná globální závislost, kterou by bylo nutné spravovat, což usnadňuje používání různých verzí Capacitoru pro každou jednotlivou aplikaci. Tato flexibilita představuje významné plus pro týmy, které vyvíjejí více aplikací s potenciálně rozdílnými verzemi nebo závislostmi. [2]

2.4 Kdy použít Capacitor

Vývoj nativní mobilní aplikace na iOS a Androidu

Capacitor zjednodušuje vývoj a nasazení nativních mobilních aplikací pro iOS a Android pomocí známých webových jazyků, knihoven a frameworků, bez složitosti podkladových nativních SDK a specifického kódu pro iOS a Android. Mobilní aplikace postavené s pomocí Capacitoru jsou optimalizovány pro nativní výkon a mají plný přístup ke všem nativním funkcím zařízení, včetně plného přístupu k nativním SDK, když je to potřeba. UI vrstva každé mobilní aplikace postavené s Capacitorem běží převážně v prohlížeči, takže nové i stávající webové aplikace lze nasadit jako nativní mobilní aplikace. [2]

Vývoj Progresivní webové aplikace (PWA)

Capacitor má plnou podporu pro progresivní webové aplikace a nativní aplikace. To znamená, že pluginový most Capacitoru podporuje spuštění buď v nativním kontextu, nebo na webu, přičemž mnoho základních pluginů je k dispozici v obou kontextech se stejným rozhraním API a volacími konvencemi. To znamená, že je možné používat @capacitor/core jako závislost jak pro nativní aplikaci, tak pro progresivní webovou aplikaci a Capacitor automaticky volá webový kód, když je to potřeba, a nativní kód, pokud je k dispozici.

Capacitor také nabízí řadu nástrojů pro získávání informací o aktuální platformě, aby bylo možné aplikaci přizpůsobit běhu v nativním režimu nebo na webu. [2]

Přidání webové části do stávající nativní mobilní aplikace

Capacitor přidává nadstandardní funkce do tradičního WebView ovládacího prvku dostupného na každé platformě a byl navržen tak, aby mohl být použit tam, kde by byl použit WebView. To znamená, že jej lze snadno integrovat do existujícího kódu nativní aplikace, což umožňuje vytvářet určité obrazovky aplikace pomocí webových technologií, aniž by bylo nutné provádět rozsáhlé změny v zbytku kódu aplikace.

Toto je také skvělý způsob, jak zapojit týmy, které mají spíše tradiční dovednosti ve webovém vývoji, do vývoje aplikace, aniž by bránily tradičnímu nativnímu vývojovému procesu. [2]

Nasazení stávající webové aplikace na mobilní zařízení

Tradičně se týmy, které chtějí rozšířit své stávající webové aplikace na jiné platformy, ocitají v dilematu: buď přepisovat aplikaci pomocí nativních technologií, nebo riskovat vyřazení nových zákazníků (většina získaných zákazníků dnes přichází prostřednictvím mobilních zařízení!). Obě varianty zahrnují ztrátu mnoha let investovaného času, peněz a úsilí, nebo riziko ohrožení budoucího růstu společnosti. [2]

S Capacitorem není potřeba začínat od začátku. Ve skutečnosti byl navržen tak, aby mohl být integrován do stávající moderní webové aplikace napsané v JavaScriptu - včetně aplikací využívajících UI frameworky jako Bootstrap, nebo Material UI. Stačí nainstalovat Capacitor, přidat požadované nativní platformy a začít využívat libovolné cross-platformové API, které Capacitor nabízí.

3 FRAMEWORK CORDOVA

Apache Cordova je open-source framework, který umožňuje vývoj mobilních aplikací pomocí standardních webových technologií, jako jsou HTML, CSS a JavaScript. Tento framework je určený pro multi-platformní vývoj, což znamená, že lze vytvářet aplikace které běží na více platformách, aniž by bylo potřeba implementovat každou část aplikace ve specifických jazycích, či nástrojích pro jednotlivé platformy.

Cordova, podobně jako Capacitor využívá API pro přístup k nativním funkcím zařízení. Což umožňuje využívat funkce nativního zařízení jako například kameru, nebo geolokaci. [3]

3.1 Pluginy

Podobně jako Capacitor je u Cordovy využito pluginů. Ty jsou nedílnou součástí ekosystému Cordova. Poskytují rozhraní mezi Cordovou a nativními komponenty a API vazby na funkce nativních zařízení. To umožňuje volání nativního kódu pomocí JavaScriptu.

Projekt Apache Cordova udržuje sadu pluginů, tzv. „Core Plugins“. Kromě těchto pluginů spravovaných samotným týmem Cordovy, existuje také řada dalších pluginů spravovaných jinými vývojáři.

Je vhodné si uvědomit, že při vytvoření projektu pomocí Cordovy nejsou přítomny žádné pluginy. Veškeré pluginy je tudíž nutno přidat dle požadovaných funkcí. [3]

4 PROCES VÝVOJE MOBILNÍ APLIKACE POMOCÍ IONIC / CAPACITOR

Proces vývoje mobilní aplikace pomocí frameworku Ionic a Capacitoru zahrnuje několik základních částí. Jsou jimi instalace a konfigurace prostředí, vytvoření projektu, vývoj a testování aplikace, případná integrace nativních funkcí pomocí Capacitoru, sestavení a následné nasazení aplikace například v App Store nebo Google Play.

4.1 Instalace a konfigurace prostředí

Instalace a konfigurace zahrnuje instalaci základních nástrojů a jejich závislostí. Ionic a Capacitor vyžadují zejména Node.js a NPM. Dále je potřeba také nainstalovat Ionic CLI a Capacitor CLI. Tyto nástroje umožňují vytvoření nových projektů, správu závislostí a sestavování aplikací.

Také bych rád zmínil nástroj NVM - Node Version Manager, ten není nijak nutný, ale dokáže mě osobně poměrně hodně pomohl. Jelikož jsem byl nucen často měnit verze jak Node.js, tak NPM.

4.2 Instalace nástrojů a závislostí

Budeme zde potřebovat čtyři hlavní nástroje, těmi jsou Node.js, NPM, Ionic CLI a Capacitor CLI.

Nejdříve můžeme nainstalovat Node.js a NPM, tyto nástroje je možné naistalovat pomocí spustitelného souboru přímo ze stránek Node.js, nebo v mém případě pomocí terminálu a nástroje Homebrew, což je správce balíčků pro operační systém macOS.

Poté je možné využít nástroje NPM pro instalaci Ionic CLI a Capacitor CLI.

npm install -g ionic@latest; npm install -g @capacitor/cli

4.3 Vytvoření nového projektu

Po úspěšné instalaci a konfiguraci vývojového prostředí může být vytvořen nový projekt. Zde je možné využít příslušného IDE, například WebStorm od JetBrains, nebo dříve nainstalovaného Ionic CLI, jenž umožňuje snadné vytváření nových projektů pomocí příkazu „ionic start“. V tomto kroku je možné také specifikovat šablonu aplikace, framework, který

chceme využít pro vývoj, například Angular, Svelte, Vue nebo React nebo také libovolné jazyky pro stylování.

4.4 Struktura projektu

Struktura projektu se skládá z několika složek a souborů, které jsou vytvořeny při tvorbě nového projektu, ty mohou být různé v závislosti na zvoleném frameworku.

Mezi tyto složky patří zejména „src“ (obsahuje zdrojové soubory aplikace), „www“ (obsahuje statické soubory jako jsou styly, fonty, nebo obrázky) a „node_modules“ (obsahuje různé knihovny a závislosti projektu).

4.5 Komponenty a moduly

Ionic framework poskytuje různé komponenty a moduly, které jsou předem vytvořené a mohou být použity v aplikaci. Nic nám však ale nebrání ve tvorbě vlastních komponent na míru.

Komponenty jsou základní stavební bloky aplikace, jako jsou například tlačítko, formulářové prvky, menu, dropdowny a mnoho dalších.

Moduly jsou skupiny komponent, které mají společné vlastnosti nebo funkce a mohou být opakovaně použity v aplikaci. V ionic frameworku existuje mnoho předem vytvořených komponent a modulů, což zrychluje vývoj a umožňuje snadnou tvorbu aplikace.

4.6 Vývoj a testování aplikace

Po vytvoření projektu je možné začít s vývojem aplikace. Ionic umožňuje snadný a rychlý vývoj uživatelského rozhraní pomocí HTML a CSS, popřípadě jiného stylovacího jazyka jako je například SCSS.

Je také vhodné, aby byla aplikace otestována. K tomu je možné použít nástroje jako jsou například Cypress, nebo Selenium

4.7 Vývoj uživatelského rozhraní

Vývoj uživatelského rozhraní je v Ionic frameworku založen na HTML a CSS. Jelikož Ionic poskytuje mnoho předem vytvořených komponent, je vhodné je využít ve všech možných případech a držet se jich, jelikož tím dosáhneme čistého vzhledu a ušetříme spoustu času. Máme také zaručeno, že aplikace bude vypadat dobře na všech platformách.

V případě nutnosti je však možné vytvořit vlastní komponenty, nebo upravit již existující.

4.8 Práce s daty a API

Pro práci s daty a API mohou být použity různé knihovny, jako například Angular HttpClient nebo Fetch API. Ionic framework také poskytuje možnost využití různých databázových řešení, jako je například SQLite.

4.9 Integrace nativních funkcí pomocí Capacitoru

Většinu nativních funkcí můžeme integrovat pomocí frameworku Capacitor, který umožňuje přístup k hardwarovým a softwarovým funkcím zařízení, jako jsou například kamera, senzory, notifikace a spousta dalších.

4.10 Nativní pluginy

Capacitor poskytuje mnoho předem vytvořených nativních pluginů, které umožňují přístup k jednotlivým funkcím zařízení. Tyto pluginy jsou napsány v nativním kódu pro jednotlivé platformy, jako je například Swift pro iOS nebo Java a Kotlin pro Android, pluginy jsou k dispozici jako balíčky NPM, což umožňuje jejich snadnou instalaci a následné použití v aplikaci.

4.11 Přístup k hardwarovým a softwarovým funkcím zařízení

Capacitor umožňuje přístup k různým hardwarovým a softwarovým funkcím zařízení, tedy kamera, senzory, lokalizace a tak podobně. Pro využití například kamery v aplikaci postavené na Ionicu, je nutné plugin nejprve nainstalovat.

Instalace

```
$ npm install @capacitor/camera
```

Implementace

```
import { Camera, CameraResultType } from '@capacitor/camera';

const takePicture = async () => {
  const image = await Camera.getPhoto({
    quality: 90,
    allowEditing: true,
    resultType: CameraResultType.Uri
  });
  const imageUrl = image.webPath;
  imageElement.src = imageUrl;
};
```

[1]

4.12 Sestavení a nasazení aplikace

Po dokončení vývoje aplikace je možné aplikaci sestavit a následně je možné ji nasadit do obchodů s aplikacemi. Sestavení aplikace zahrnuje vytvoření souboru, který může být nahrán do obchodů s aplikacemi. Následující body popisují proces sestavení aplikace a její následné nasazení.

4.12.1 Proces sestavení pro různé platformy

Sestavení aplikace pro různé platformy může být odlišné. Proces sestavení aplikace pro platformu iOS se různí od procesu sestavení pro platformu Android. Proces sestavení aplikace pro platformu iOS vyžaduje použití Xcode, zatímco pro sestavení aplikace pro platformu Android je nutné využít Android Studio. Ionic CLI umožňuje snadné sestavení aplikace pro různé platformy pomocí příkazu „ionic capacitor build“.

4.12.2 Nasazení aplikace do obchodů s aplikacemi

Po dokončení sestavení aplikace je možné ji nahrát do obchodů App Store či Google Play. Pro nahrání aplikace do obchodu s aplikacemi je nutné mít vytvořený účet pro daný obchod a dodržet požadavky na nahrání aplikace. Tyto požadavky mohou zahrnovat nahrání ikon, popisu aplikace a certifikátů. Zde se mohou požadavky lišit, například pro Apple jsou poměrně náročné, a je nutné dosáhnout jak stability aplikace, tak je nutné držet se Applem předepsaných pravidel pro vzhled, aby bylo dosaženo požadavků pro UI a UX.

Další rozdíl může být ten, že vydání aplikace pro Google Play je většinou rychlejší, jelikož danou aplikaci kontroluje stroj. U App Storu je tomu tak, že aplikace je kontrolována člověkem, což způsobuje delší dobu pro vydání aplikace, díky tomu je však dosaženo kontroly jak funkcionality, tak UI a UX.

Po úspěšném nahrání aplikace do obchodu s aplikacemi bude aplikace dostupná pro stažení a instalaci uživateli.

5 VÝHODY A NEVÝHODY POUŽITÍ IONIC/CAPACITOR OPROTI JINÝM ŘEŠENÍM

5.1 Výhody

- **Podpora více platforem**: Ionic framework nabízí velkou výhodu v podpoře více platforem. Vývojáři nemusí psát oddělený kód pro každou platformu zvlášť, ale většina kódu dokáže být multiplatformní. Ovšem i zde se může stát, že bude potřeba sáhnout na nativní úroveň jak pro iOS, tak pro Android. Ale i přes to daný postup dokáže velmi ušetřit jak čas na vývoj aplikace, tak náklady na údržbu i testování.

- **Snadná integrace s webovými technologiemi**: Jelikož je Ionic postaven na webových technologiích, jako jsou HTML, CSS a JavaScript, umožňuje snadnou integraci s existujícími webovými aplikacemi. Díky tomu lze rychle převést webovou aplikaci na mobilní aplikaci, což šetří čas a náklady na vývoj.

- **Široká komunita a podpora**: Ionic má poměrně velkou komunitu uživatelů a vývojářů, což zajišťuje vysokou úroveň podpory a neustálý vývoj nových funkcí. Komunita také poskytuje řešení problémů a sdílí zkušenosti s vývojem aplikací. To nesmírně usnadňuje přístup k užitečným informacím.

- **Přizpůsobivost a škálovatelnost**: Ionic framework umožňuje přizpůsobit aplikaci dle potřeb uživatele a klienta. Mnoho předem vytvořených komponent umožňuje rychlé vytvoření uživatelského rozhraní aplikace, které lze dále upravovat dle daných specifikací. Je také poměrně jednoduché škálovat aplikace pro různé platformy.

5.2 Nevýhody

- **Možné problémy s nativními funkcemi**: Framework Ionic používá webové technologie a může mít omezené možnosti pro přístup ke specifickým funkcím hardwaru zařízení. Tento problém se obvykle řeší pomocí pluginů, které umožňují přístup k nativním funkcím. Nicméně tyto pluginy mohou mít určité problémy a mohou být nekompatibilní s některými verzemi Ionicu, nebo s určitými platformami. Kromě toho se můžeme setkat se špatnou kompatibilitou pro specifické verze Node.js a tak podobně. Kromě toho, pokud se vyskytnou problémy s pluginy, může být obtížnější je opravit, než kdyby se jednalo o nativní funkce. To může vést k prodloužení vývoje aplikace, a růstu nákladů.

- **Menší kontrola oproti nativním aplikacím**: Při použití Ionicu vývojáři nemají plnou kontrolu nad aplikací, jako kdyby ji vyvíjeli čistě nativně. Každopádně tuhle nevýhodu silně převyšují výhody, jako jsou především nižší čas pro vývoj aplikace a nižší náklady, jelikož tohle jsou jedny z nejzajímavějších bodů pro klienty.

5.3 Porovnání s dalšími frameworky

V době, kdy se mobilní aplikace stávají stále důležitějšími pro podnikání, komunikaci a zábavu, může být výběr správného nástroje pro jejich vývoj klíčový. Existuje poměrně velké množství různých technologií a frameworků, které vývojářům umožňují psát mobilní aplikace, přičemž se od sebe mohou frameworky znatelně lišit. Každý z nich má také své výhody, či nevýhody.

Níže v textu je krátké porovnání frameworků, které patří ve světě hybridních aplikací mezi vůbec nejpopulárnější. [5]

5.3.1 React Native

Ionic a React Native patří mezi nejvíce používané hybridní mobilní frameworky na trhu. Oba umožňují vývoj aplikací pro více platforem a snadnou integraci s webovými technologiemi. Nicméně existují rozdíly v architektuře a funkčnosti těchto frameworků.

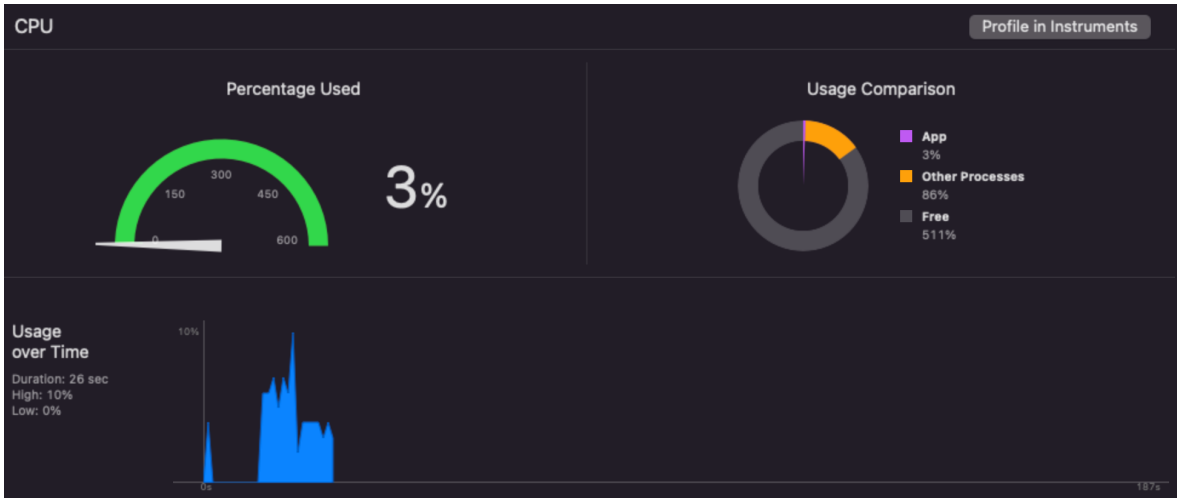
React Native je open source projekt, který byl vydán společností Facebook – dnes Meta, v roce 2015 s cílem přidat ovládání uživatelského rozhraní pro iOS a Android prostřednictvím frameworku React a Javascriptu a urychlit tak vývoj aplikací. React Native je možné integrovat do stávajících nativních aplikací nebo použít k vytvoření aplikace od nuly. [7]

Jelikož React Native nepoužívá k vykreslování prvky DOM, vyžaduje použít jiné vykreslovací knihovny než tradiční React, používaný pro vývoj webových aplikací. [7]

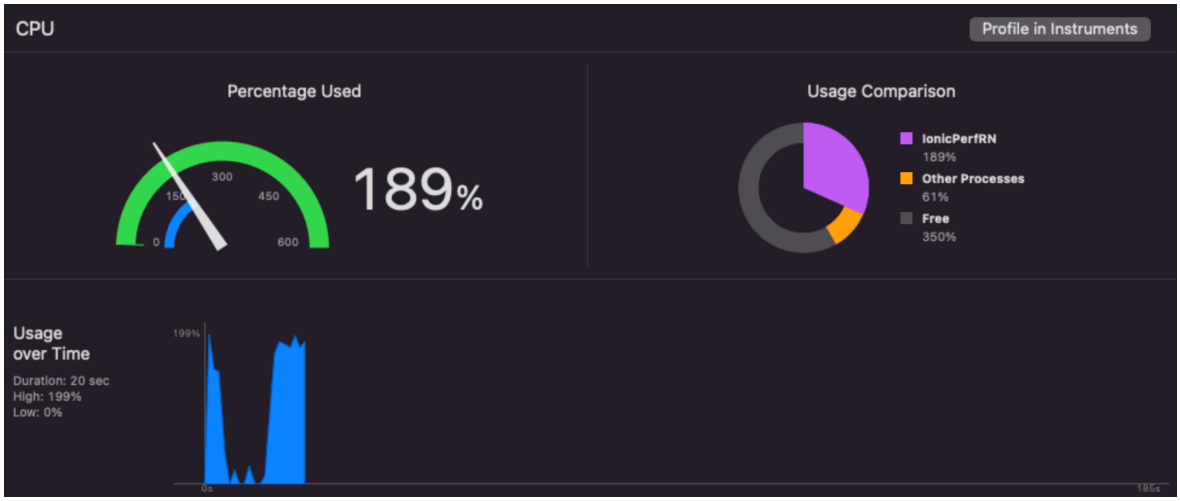
React Native používají společnosti jako jsou Shopify, Meta a Wix.

Využití hardwarových prostředků pro běh aplikace

Následující test ukazuje, jak se aplikace chová při spuštění pomocí Xcode a scrollování přes list zaměstnanců



Obrázek 1 - využití CPU aplikací vytvořenou pomocí Ionicu [7]



Obrázek 2 - využití CPU aplikací vytvořenou pomocí React Native [7]

Z těchto obrázků je patrné, že aplikace Ionic je mnohem méně náročná na procesor než aplikace React Native. Aplikace React Native využívala v nejvyšších hodnotách téměř 200 % procesoru, zatímco aplikace Ionic pouze 10 %. Důvodem je to, že hybridní aplikace Ionic mají přístup k rychlejšímu JS enginu v rámci WKWebView než hybridní aplikace využívající JavaScriptCore, například ty vytvořené pomocí React Native. Zvenčí se zdá, že obě tyto

aplikace běží hladce, ale pod kapotou zařízení pracuje mnohem více, aby zajistilo plynulý běh aplikace napsané pomocí React Native. [7]

Vzhledem k metrikám zatížení procesoru je možno vidět, že aplikace React Native zatěžovala zařízení o dost více než aplikace psaná v Ionic.

Obrázek 3 - Spotřeba energie aplikace Ionic [7]

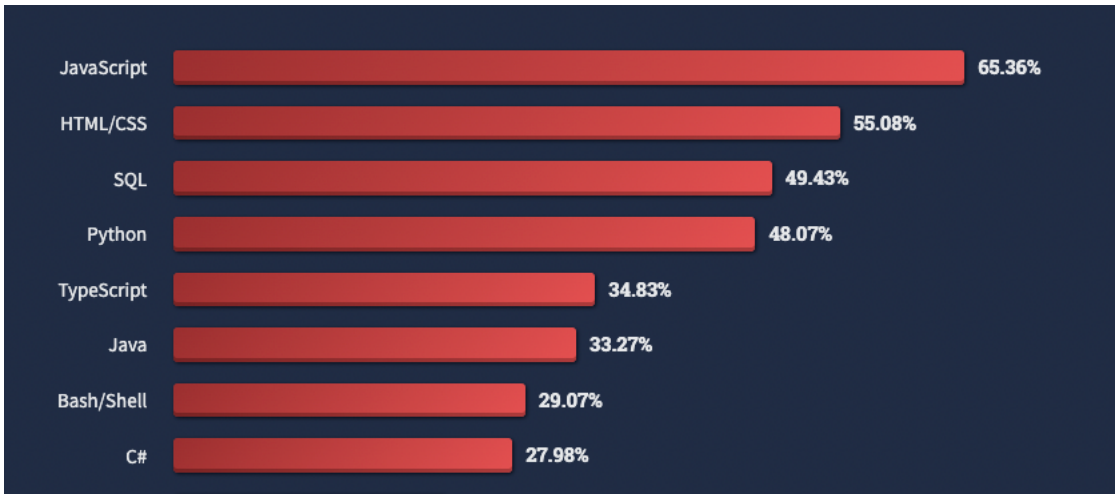
Obrázek 4 - Spotřeba energie aplikace React Native [7]

Při opětovném procházení seznamů se Ionic dostal do kategorie „High Energy Impact“, zatímco React Native se dostal o něco dále, a to až do kategorie „Very High Energy Impact“. Bylo důležité podívat se na tyto rozsahy v době, kdy aplikace nebyly nečinné, protože bude mnohem méně případů, kdy uživatelé interagují s aplikací, aniž by provedli nějakou akci. Pokud spotřeba energie zůstane po celou dobu jejího využívání velmi vysoká, bude to mít poměrně špatný vliv na baterii zařízení. Z výše uvedených příkladů vyplývá, že aplikace Ionic bude mít menší dopad na celkovou výdrž baterie zařízení. [7]

5.3.2 Xamarin

Jeden z největších rozdílů pro technologie Ionic a Xamarin není ani tak technický, jako jejich popularita technologii, které jsou pro ně potřeba. Z obrázku uvedeného níže je patrné, že zde jednoznačně vede JavaScript, společně s HTML a CSS nad C#, který je používán pro Xamarin. Přestože se nejedná o přímo technický aspekt, patří mezi poměrně důležité, jelikož z něj vyplývá že sehnat vývojáře, kteří mají nějaké zkušenosti s trojicí Javascript, HTML a CSS bude mnohem jednodušší než shánět vývojáře pro C#. [9]

StackOverflow developers's survey - 2022



Obrázek 5 - StackOverflow developers's survey - 2022 [8]

Pokud se zaměříme na více technické aspekty, přestože je Xamarin skvělý pro vývoj mobilních aplikací a .NET je skvělý pro web, stále je zde nutné vytvářet a spravovat několik oddělených kódů, a to hlavně pokud je žádoucí stejnou aplikací cílit na mobilní zařízení a zároveň web. Zejména pak ve chvíli, pokud se bude jednat o tradiční nativní mobilní aplikaci, která bude existovat také jako PWA neboli progresivní webová aplikace. [9]

Na rozdíl od Xamarinu, aplikace tvořena pomocí Ionicu založeného na webových technologiích, běží přímo v jakémkoli desktopovém nebo mobilním prohlížeči – a to vše za pomoci jednoho kódu. [9]

Pro srovnání, framework Xamarin je možné použít k vývoji nativních aplikací pro Android a iOS pro mobilní zařízení a k vytváření nativních aplikací pro Windows a Mac pro stolní počítače. Xamarin však nenabízí možnosti nasazení na webu. Webové aplikace je tedy nutné psát pomocí ASP.NET, problém nastává v tom, že mezi aplikacemi napsané v Xamarinu a ASP.NET nelze sdílet příliš mnoho kódu, takže je nutné spravovat a udržovat dva projekty odděleně. [9]

5.3.3 Flutter

Ionic a Flutter sdílí společnou vizi vytváření krásných, vysoce výkonných aplikací, které fungují všude. Avšak filozofie obou projektů už nemůžou být odlišnější. [10]

Hlavní zásadou Ionicu je používat webovou platformu a přijímat otevřené standardy všude, kde je to možné.

Při tvorbě aplikace s Ionicem je kladen důraz na používaní nástrojů a jazyky webu, a využití frameworku – tedy Ionicu, který je navržen tak, aby poskytoval skvělý výkon na mobilních zařízeních, počítačích a zejména na webu.

Flutter se naproti tomu rozhodl jít jinou cestou a vytvořil samostatný ekosystém, který je v rozporu s běžnými jazyky, nástroji a standardy, které se vyskytují v širším světě vývoje hybridních aplikací.

Flutter sice poskytuje velmi dobrý výkon na mobilních zařízeních, ale zásadní omezení jeho architektury z něj činí poměrně špatnou volbu pro nasazení na webu. [10]

Jak funguje Flutter

Jádrem frameworku Flutter je programovací jazyk Dart, který byl vydán během roku 2011. Přestože je již nějakou dobu na světě, dnes ho využívá mnohem méně vývojářů než výše uvedené alternativy. [10]

Při vytváření aplikací pro mobilní zařízení používá Flutter kompilátor na převod kódu napsaného v jazyce Dart na nativní strojový kód, který poběží na platformě zařízení společně s vlastním vykreslovacím jádrem pro zobrazení uživatelského rozhraní uvnitř mobilní aplikace. Flutter nepoužívá nativní prvky uživatelského rozhraní, se kterými se můžeme setkat například u React Native, ani nepoužívá webové komponenty jako Ionic. Místo toho Flutter nabízí vlastní knihovnu widgetů pro tvorbu uživatelského rozhraní. [10]

Mobilní aplikace tvořené pomocí frameworku Flutter přistupují k nativním prvkům zařízení pomocí knihovny zásuvných modulů, která je podobná knihovnám, které používají jak Ionic, tak třeba React Native. Pro přístup k běžným funkcím zařízení jsou k dispozici hotové balíčky. Pokud však takový balíček, nebo zásuvný modul neexistuje, je zde možnost si napsat vlastní kód specifický pro danou platformu, a to pomocí asynchronní komunikační platformy specifické pro Flutter, která zajišťuje komunikaci mezi klientem – uživatelským rozhraním a hostitelem – operační systém platformy. [10]

Přenositelnost aplikací tvořených pomocí frameworku Flutter

Zatímco Flutter se může chlubit úplnou přenositelností aplikace, nasazení Flutter aplikace na webu vyžaduje určitou „kreativní akrobacii“. Flutter používá ke generování JavaScriptu, který se vykreslí v prohlížeči stejný kompilátor Dart. Poté nahradí grafické jádro a vykreslovač textu založená na Skia, vlastním vykreslovacím jádrem a následně od základu přebuduje základní prvky webového prohlížeče, jako je výběr textu, kopírování a vkládání. Každá vytvořená aplikace pak obsahuje celé tyto přepracované prvky. Zde jde možné si všimnout určité podobnosti s Adobe Flash. [10]

6 TECHNOLOGIE VHODNÉ PRO VÝVOJ CAPACITOR PLUGINU

Níže jsou popsány technologie, které jsou vhodné, popřípadě také nutné pro vývoj Capacitor pluginu. Může se jednat o různé programovací jazyky, popřípadě verzovací systémy a další.

6.1 Javascript

Javascript je skriptovací, nebo také programovací jazyk, který umožnuje u webových stránek, popřípadě dalších aplikací implementovat složitější funkce. Ve většině případů, když webová stránka dělá něco víc, než jen zobrazuje statické informace – například aktualizuje obsah, zobrazuje interaktivní mapy, data, 2D nebo 3D grafiku, popřípadě video a tak podobně, je zde s nejvyšší pravděpodobností použit právě JavaScript.

6.2 Typescript

TypeScript je velmi podobný JavaScriptu, je možné říct, že se jedná o jeho rozšíření. Velmi důležitou vlastností je zde statické typování, které umožní předejít problémům s datovými typy, a ve většině případech je tak mnohem jednodušší editovat či rozšiřovat větší aplikace.

6.3 HTML

HTML je značkovací jazyk, který se používá pro tvorbu struktury webu, například k definování odstavců, nadpisů, textů, tabulek či vkládání obrázku.

6.4 CSS

CSS je jazyk který umožňuje aplikovat styly na výše zmíněný obsah jazyka HTML. Například nastavení barvy pozadí, písma, úprava velikosti jednotlivých elementů a tak podobně.

6.5 GIT

Git je systém pro správu verzí, který je široce rozšířen a běžně používán při vývoji softwaru. Umožňuje vývojářům sledovat změny v kódu, spolupracovat s ostatními a sdílet kód. Také je možné kód rozdělit do různých větví, které lze později opětovně spojit do hlavní. Znalost Gitu je tedy velmi žádaná pro vývoj softwaru.

6.6 Swift / Objective-C

Jelikož může být při vývoji Capacitor pluginu potřeba využít nativních funkcí iOS zařízení, je vhodné znát také jazyky používané pro nativní vývoj. Těmi jsou Objective-C a nově Swift,

což je moderní, bezpečný a výkonný jazyk, zatímco Objective-C je starší, ale přesto stále používaný jazyk pro vývoj iOS aplikací.

6.7 Java / Kotlin

Jelikož může být při vývoji Capacitor pluginu potřeba využít nativních funkcí Android zařízení, je vhodné znát také jazyky používané pro nativní vývoj. Těmi jsou Java a nově Kotlin, který je vyvíjen českou společností JetBrains, jenž byla založena roku 2000 v Praze. Jedná se o staticky typovaný programovací jazyk, který běží na JVM – Java Virtual Machine a může být také kompilován do JavaScriptu a nativního kódu. Kotlin je také od roku 2017 oficiálně podporován společností Google jako jazyk pro vývoj Android aplikací. [11]

Oproti Jave má několik vylepšení, které zvyšují produktivitu a zjednodušují vývoj, ty jsou například:

- Vestavěná podpora pro manipulaci s nullovými hodnotami, což značně pomáhá předejít chybám [11]
- Podpora coroutines, což je funkce, která zjednodušuje asynchronní programování a umožňuje psát čitelnější a efektivnější kód. [11]
- Skriptování – Kotlin může být také použit jako skriptovací jazyk [11]
- Immutabilita, neboli podpora neměnných proměnných a kolekcí, což je užitečné pro psaní srozumitelného kódu, který je zároveň méně náchylný na chyby [11]

6.8 Další vhodné technologie

Jako další technologie vhodné pro vývoj Capacitor pluginu, je možné zařadit Capacitor jako takový, poté je vhodné zmínit Node.js a NPM, které jsou v bakalářské práci již více detailně popsány.

7 FUNKČNÍ A NEFUNKČNÍ POŽADAVKY

7.1 Funkční požadavky

1. Capacitor plugin musí být schopen přijmout base64String obrázku jako vstup.
2. Capacitor plugin musí mít implementovanou funkcionalitu pro rozpoznání textu v obrázku.
3. Capacitor plugin musí být schopen vrátit nalezený text.
4. V případě, že nastane nějaká chyba, capacitor plugin musí vyhodit chybovou hlášku.

7.2 Nefunkční požadavky

1. Capacitor plugin musí být snadno použitelný pro vývojáře.
2. Capacitor plugin musí být dostatečně zdokumentovaný.
3. Capacitor plugin by měl být v souladu se standardy pro vývoj softwaru a měl by být otestován pro různé platformy a za různých podmínek tak, aby se zajistila jeho spolehlivost.
4. Capacitor plugin by měl být dobře optimalizován, aby minimalizoval využití hardwarových zdrojů.
5. Capacitor plugin by měl být použitelný také na zařízeních s omezenými výpočetními kapacitami a pamětí.
6. Capacitor plugin by měl být kompatibilní s nejnovějšími verzemi operačních systémů a webových prohlížečů, aby byla zajištěna dostupnost pro co nejvíce uživatelů.

II. PRAKTICKÁ ČÁST

8 TVORBA CAPACITOR PLUGINU

Pro tvorbu Capacitor pluginu bude využito několika technologií, které byly již dříve popsány v bakalářské práci. Jedná se zejména o samotný Capacitor, Node.js a Node Package Manager.

8.1 Nastavení vývojového prostředí

Před začátkem samotného vývoje, je potřeba nastavit prostředí ve kterém bude vývoj probíhat. Je nutné mít nainstalované Node.js a Node Package Manager.

Pro ověření, zda jsou výše zmíněné nástroje správně nainstalované, můžeme využít následující příkazy, pokud každý z nich vypíše nějakou verzi, nástroj je nainstalován.

- *node -v*
- *npm -v*

8.2 Vytvoření projektu

Pro vytvoření projektu je vhodné použít NPM a Capacitor. To umožní vytvoření celého projektu pomocí krátkého příkazu v terminálu.

- *npm init @capacitor/plugin@latest*

Po zadání příkazu je možné zadat jméno pluginu, jeho ID a cestu k projektu.

8.3 Struktura projektu

Po vygenerování projektu bude jeho struktura vypadat přibližné takto:

- android/ - obsahuje kód pro platformu Android
- ios/ - obsahuje kód pro platformu iOS
- src/definitions.ts - obsahuje TypeScriptové interface pro capacitor plugin
- src/web.ts - obsahuje webovou implementaci pluginu
- src/index.ts - tento soubor exportuje vše ze dvou výše uvedených souborů, kterými jsou „definitions.ts“ a „web.ts“
- package.json - tento soubor definuje použité npm balíčky

Většina kódu, který bude potřeba pro tvorbu pluginu, který umožní rozpoznání textu z obrázku, bude implementován právě ve složkách „ios“, „android“, a „src“.

8.4 Sestavení projektu

Po dokončení implementace, je potřeba plugin sestavit. Tento proces zahrnuje kompilaci TypeScript kódu do JavaScriptu. Pro sestavení projektu stačí otevřít projekt v terminálu a použít následující příkaz:

- *npm run build*

8.5 Publikování projektu

Po dokončení implementace a následném sestavení, je možné projekt, potažmo Capacitor plugin publikovat pro Node Package Manager. Při tomto kroku je potřebné vytvořit účet u NPM.

Následně je možné se v terminálu přihlásit pomocí příkazu:

- *npm login*

Poté už postačí pouze otevřít projekt v terminálu, a zadat příkaz:

- *npm publish*

8.6 Aktualizace projektu

Pokud jsou v projektu provedeny nějaké změny, jeho aktualizace je poměrné jednoduchá záležitost. Avšak je potřeba provést změnu verze v souboru „package.json“. Poté je potřeba projekt znovu sestavit a publikovat. K tomu lze použít již dříve zmíněné příkazy, které jsou následující:

- *npm build*
- *npm publish*

8.7 Filozofie vývoje pluginů

Pokud je plugin určen pro veřejnost, existuje několik filozofií ohledně vývoje capacitor pluginů, které jsou doporučeny dodržovat. [6]

8.7.1 Spolupráce

Tým Capacitoru je pevným zastáncem toho, že spolupráce přinese vyšší kvalitu pluginů než soutěžení mezi vývojáři. To je jeden z důvodů, proč vytvořili organizaci *Capacitor Community* na platformě GitHub, která usnadňuje spolupráci v komunitě při hostování jednotlivých pluginů v osobních repositářích.

Pokud již nějaký plugin pro dané téma v Capacitor komunitě existuje, je vhodné zvážit, za je možné k němu nějak přispět, namísto vývoje nového pluginu.[6]

8.7.2 Malý rozsah

Capacitor tým věří, že jednotlivé pluginy by měly být menšího, popřípadě rozumného rozsahu. Capacitor pluginy přidávají nativní kód do aplikací, který se může, nebo naopak nemusí používat. Udržováním malého rozsahu pluginů je možné zajistit, že aplikace budou mít minimální množství nativního kódu, který potřebují. To zabrání zbytečnému zvětšování aplikace a varováním či odmítnutím ze strany App Store, kvůli API bez popisů použití či podobných důvodů. [6]

Malý rozsah pluginů však přináší i další výhody, jako je rychlejší nasazení, jednodušší spolupráce, udržovatelnost a podobně. [6]

8.7.3 Jednota a idiomatičnost

Capacitor pluginy by měly usilovat o poskytování jednotného zážitku napříč platformami, který je znám především JavaScriptovým vývojářům. [6]

Zde je několik bodů, jak je možné zajistit jednotný a idiomatický zážitek:

- **použití *undefined* namísto *null hodnot*** - například pokud API pro Android vrací *0.0* která označuje „žádnou hodnotu“, pak by měla být tato hodnota pro vrstvu JavaScriptu převedena na hodnotu *undefined*. [6]
- **použití stejných jednotek** - například pokud iOS API používá stupně celsia a Android API používá stupně Fahrenheita, potom by hodnota měla být sjednocena ještě předtím, než se dostane k JavaScript kódu. [6]
- **preferování ISO 8601 normy** - Jelikož je jednoduché získat přesný JavaScriptový objekt *Date* z řetězce jako „2020-12-13T20:21:58.415Z“, ale může nastat problém, pokud je poskytnut Unixový časový údaj, jelikož JavaScriptové časové údaje jsou v milisekundách. Vždy je nutné zahrnout časovou zónu, jinak mohou být datumy a časy interpretovány nesprávně v různých lokalitách. [6]

9 CAPACITOR PLUGIN

Capacitor plugin byl vytvořen pomocí technologií Capacitor, Node.js a NPM, pro samotné rozpoznání textu byly využity následující technologie:

- webová část - Tesseract.js
- iOS část - Apple Vision Framework
- Android část - Google ML Kit

Plugin je hostovaný na GitHubu a publikovaný na npm pod názvem *cap-ocr*.

Jeho klíčovou funkcionalitou je rozpoznávání textu z obrázku, nebo také na fotkách, které byly zaznamenány pomocí kamery zařízením, na kterém běží aplikace, jenž plugin využívá. Jedná se o poměrně žádanou vlastnost v moderních technologiích.

Nyní vyvíjený plugin poskytuje dvě hlavní API funkce, těmi jsou:

- *detectText()*
- *detectData()*

Obě funkce přijímají parametr ve formě objektu, který by měl obsahovat obrázek zakódovaný do formátu base64. Po úspěšném zpracování a extrakci textu vrací promise, ze kterého je poté možno získat objekt, jenž obsahuje výsledná data.

Instalace samotného pluginu je velmi jednoduchá, a lze ji provést za pomoci správce balíčku NPM. Je potřeba využít následujících příkazů:

- *npm install cap-ocr* - nainstaluje požadovaný plugin
- *npx cap sync* - synchronizuje plugin v projektu

Aktuálně je verze pluginu *0.0.3*. Každopádně i v této počáteční fázi vývoje si plugin vysloužil už nějakou malou pozornost od uživatelů. Dva dny od publikace dosáhnul něco málo přes sto stažení, což pouze zdůrazňuje poptávku po funkcích optického rozpoznávání znaků v dnešních aplikacích.

Plugin je distribuován pod licencí MIT. Tato licence uživatelům poskytuje velkou svobodu, umožňuje jim používat, kopírovat, upravovat, publikovat a distribuovat, popřípadě také prodávat kopie tohoto softwaru. Tato flexibilita činí plugin atraktivní volbou pro vývojáře, kteří si jej chtějí přizpůsobit svým konkrétním potřebám.

Pokud jde o velikost, plugin je poměrně malý. Jeho zanedbatelná velikost je pouze necelých 40 kB, což znamená, že nijak významně nezvětší velikost projektu.

9.1 Popis funkcionality pluginu

V této části je detailně popsáno, jak plugin funguje. Je využito webové části, a to z toho důvodu, že implementace je nejjednodušší a je tedy vhodnější pro pochopení.

Hlavní implementace capacitor pluginu je obsažena v následujících souborech:

- *index.ts* - jedná se o vstupní soubor Capacitor pluginu, importuje *registerPlugin* z jádra Capacitoru a také se stará o import interface ze souboru *definitions.ts*
- *definitions.ts* - tento soubor definuje hlavní interface, které plugin využívá
- *web.ts* - zde se nachází implementace samotných funkcí pro detekci textu

Pokud se blíže podíváme na samotnou implementaci, ta je rozdělena na do tří hlavních funkcí, těmi jsou:

- *getBlob(base64: string): Blob*
- *detectText(options: {imageBase64: string}): Promise<{value: string}>*
- *detectData(options: {imageBase64: string}): Promise<{value: string}>*

Funkce getBlob:

Jedná se o funkci, která není přístupná skrze Capacitor API, a tudíž ji uživatel nemůže využívat samostatně. Jelikož je ve webové implementaci využito knihovny Tesseract.js pro extrakci textu, bylo nutné pro ni data předpřipravit, jelikož nedokáže použít textový řetězec zakódovaný do base64 formátu. Tesseract.js dokáže pracovat s různými typy vstupních dat, jako je například *HTMLImageElement*, *File*, nebo právě *Blob*, který byl zvolen.

Ve zkratce jde tedy pouze o pomocnou funkci, která předpřipravý data pro Tesseract.js.

Zde může nastat otázka, proč byl obrázek v první řadě vůbec převeden a kódován do base64 a nebyl ponechán v původním formátu. Zde je vhodné zmínit, že by to samozřejmě možné bylo, ale jelikož má plugin fungovat jak pro web, tak také iOS a Android. To zapříčinilo, že bylo využito různých nástrojů pro rozpoznávání textu, přičemž každý funguje nějak jinak, ať už se jedná o samotné zpracování nebo vstupní data, se kterými jednotlivá technologie dokáže pracovat. Přesně z toho důvodu byl zvolen formát base64, se kterým přímo uživatel pracuje, a veškerá následující úprava je provedena na straně Capacitor pluginu. Tim byla zajištěna určitá všeobecnost pro jednotlivé platformy.

Funkce detectText:

Jedná se o samotnou funkci pro provedení extrakce textu. Funkce přijímá parametr, ve kterém je obsažen textový řetězec s obrázkem, jenž je zakódován do formátu base64, následně využívá pomocné funkce *getBlob*, která vstupní řetězec převede na blob, tedy „Binary large object“, se kterým už dokáže pracovat Tesseract.js. Po zpracování a extrahování textu funkce vrátí textový řetězec, ve kterém je obsáhnutý pouze samotný nalezený text, avšak neobashuje žádné další data, jako jsou například jednotlivé odstavce, řádky textu, nebo pozice, kde se text nachází.

Funkce detectData:

Jde o funkci, která je velmi podobná výše zmíněné funkci *detectText.* Tato funkce také přijímá parametr, kde je obsažen textový řetězec s obrázkem zakódovaným do formátu base64. Poté je využita pomocná funkce *getBlob*, díky které je vstupní řetězec převeden na blob. Ten je následně předán Tesseractu pro zpracování a samotnou extrakci textu. Ve chvíli, kdy jsou úspěšně extrahovány veškeré data, funkce vrátí promise, který obsahuje objekt s nalezenými daty. Na rozdíl od předchozí funkce jsou zde také obsažena data jako je samotný nalezený text, data obsahují navíc například jednotlivé odstavce, řádky textu, pozici, ve které se text nachází a dále hodnota, jak si je Tesseract jistý extrakcí a tak podobně. Jde tedy o velmi podobnou funkci, která ale vrací více dat.

9.2 Budoucnost Capacitor pluginu

Jelikož se ukázalo, že existuje reálný zájem o capacitor plugin, který dokáže ulehčit rozpoznávání textu jak na webu, tak na platformách iOS a Android, jeho vývoj bude nadále pokračovat i po dokončení a obhajobě bakalářské práce.

Mezi další funkcionalitu, která by mohla být žádaná, patří následující:

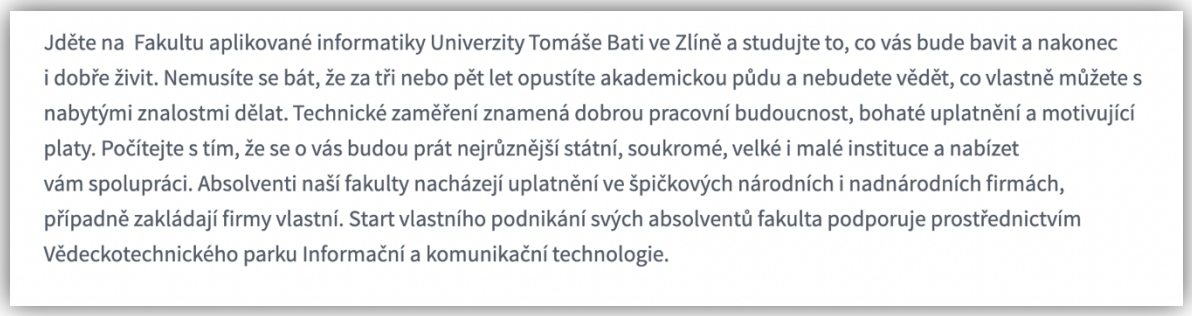
1) **Vylepšené rozpoznávání znaků** - bylo by vhodné přidat možnost pro předpřipravení obrázků takovým způsobem, aby bylo pro jednotlivé technologie jednodušší provést extrakci textu. Tím by bylo možné dosáhnout také rychlejšího a efektivnějšího rozpoznání textu. Zde může patřit například změna jasu, ořezání a tak podobně.
2) **Rozpoznání stylování textu** - tato funkcionalita by přinesla možnost rozpoznat stylistické prvky jakou je barva, tučné písmo, písmo s kurzívou a také přibližnou velikost daného písma
3) **Rozpoznání tabulek** - funkcionalita zaměřena na rozpoznání tabulek ve skenovaných dokumentech a poté generování objektů spolu s převedeným textem může pomoci usnadnit převod dokumentů na text a zároveň ponechání původních tabulek.
4) **Více vláknové zpracování** - schopnost provádět detekci a rozpoznání textu v paralelních vláknech může výrazně zlepšit rychlost a efektivitu procesu. Zejména v případě, pokud by aplikace vkládala více obrázku najednou. Aktuálně je počítáno pouze s jedním obrázkem.
5) **Automatické rozpoznání a úprava rotace stránky** - jelikož se může stát, že obrázek či fotka nebudou otočeny správným směrem, je vhodné, aby plugin dokázal tenhle problém detekovat a následně také správně vyřešit. To může urychlit samotné zpracování a přesnost extrakce textu.

9.3 Zhodnocení implementovaného Capacitor pluginu

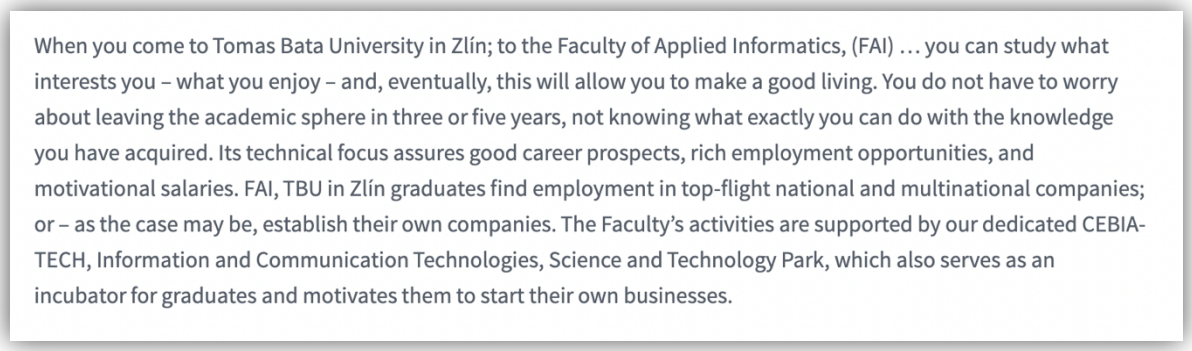
Funkcionalita Capacitor pluginu byla otestována jak na simulovaném, tak reálném zařízení. Jelikož bylo využito tří rozdílných technologií pro jednotlivé platformy, kterými jsou web, iOS a Android, jejich chování a efektivita se mohou lišit. Avšak všechny tři řešení jsou schopny přijmout obrázek zakódovaný do formátu base64, a následně z něj extrahovat text, což splňuje požadavky na vytvořený plugin.

9.4 Porovnání úspěšnosti extrakce textu a limity aktuální implementace

Jelikož se Capacitor plugin spoléhá na tři různé technologie, bude se porovnání zaměřovat na každou z nich, budou zhodnoceny v čem vynikají a taktéž jejich limity. Testování bude probíhat na následujících obrázcích


Jděte na Fakultu aplikované informatiky Univerzity Tomáše Bati ve Zlíně a studujte to, co vás bude bavit a nakonec i dobře živit. Nemusíte se bát, že za tři nebo pět let opustíte akademickou půdu a nebudete vědět, co vlastně můžete s nabytými znalostmi dělat. Technické zaměření znamená dobrou pracovní budoucnost, bohaté uplatnění a motivující platy. Počítejte s tím, že se o vás budou prát nejrůznější státní, soukromé, velké i malé instituce a nabízet vám spolupráci. Absolventi naší fakulty nacházejí uplatnění ve špičkových národních i nadnárodních firmách, případně zakládají firmy vlastní. Start vlastního podnikání svých absolventů fakulta podporuje prostřednictvím Vědeckotechnického parku Informační a komunikační technologie.

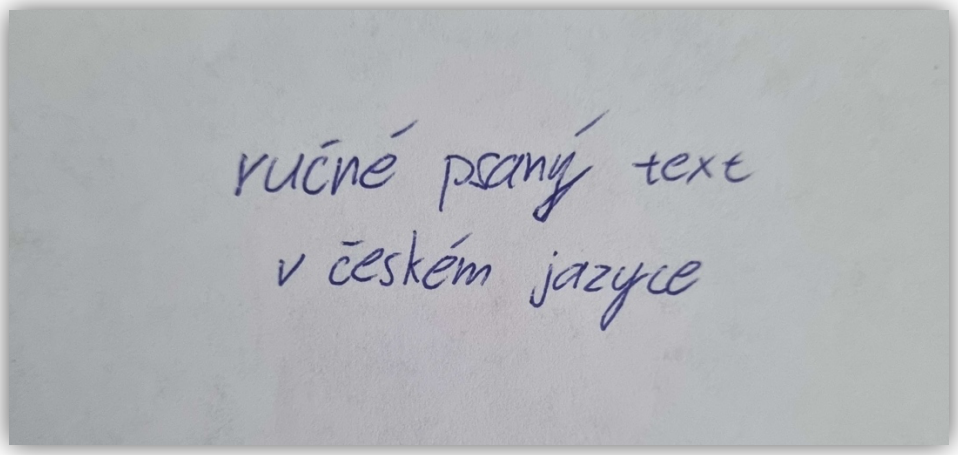
Obrázek 6 - text(cz) "proč studovat na FAI" [22]


When you come to Tomas Bata University in Zlín; to the Faculty of Applied Informatics, (FAI) … you can study what interests you – what you enjoy – and, eventually, this will allow you to make a good living. You do not have to worry about leaving the academic sphere in three or five years, not knowing what exactly you can do with the knowledge you have acquired. Its technical focus assures good career prospects, rich employment opportunities, and motivational salaries. FAI, TBU in Zlín graduates find employment in top-flight national and multinational companies; or – as the case may be, establish their own companies. The Faculty's activities are supported by our dedicated CEBIA-TECH, Information and Communication Technologies, Science and Technology Park, which also serves as an incubator for graduates and motivates them to start their own businesses.

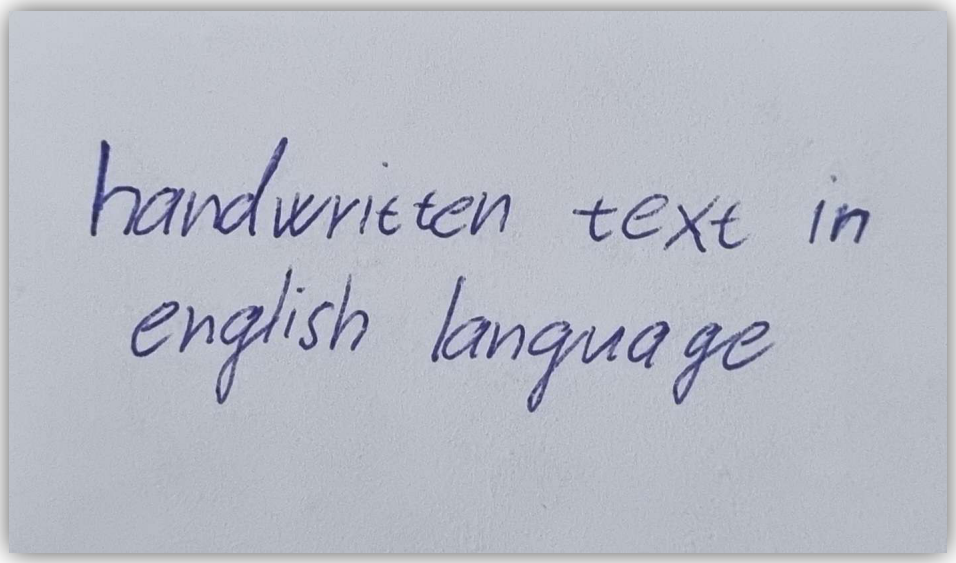
Obrázek 7 - text(en) "why study at FAI" [22]

Obrázek 8 - web FAI - hlavička(cz) [22]

Obrázek 9- web FAI - hlavička(en) [22]



Obrázek 10 - ručně psaný text(cz)



Obrázek 11 - ručně psaný text(en)

9.4.1 Testovací případ pro Obrázek 8

Extrahovaný text na webové platformě:

„Jdéte na Fakultu aplikované informatiky Univerzity Tomase Bati ve Zliné a studujte to, co vas bude bavit a nakonec i dobre Zivit. Nemusite se bat, Ze za tfi nebo pét let opustite akademickou plidu a nebudete védét, co vlastné mizete s nabytymi znalostmi délat. Technické zaméreni znamena dobrou pracovni budoucnost, bohaté uplatnéni a motivujici platy. Pocitejte s tim, Ze se o vas budou prat nejriiznéjsi statni, soukromé, velké i malé instituce a nabizet vam spolupraci. Absolventi nasi fakulty nachazeji uplatnéni ve Spi¢kovych narodnich i nadnarodnich firmach, pripadné zakladaji firmy vlastni. Start vlastniho podnikani svych absolventt fakulta podporuje prostrednictvim Védeckotechnického parku Informacni a komunikacni technologie.“

Extrahovaný text na platformě iOS:

„Jdête na Fakultu aplikované informatiky Univerzity Tomáse Bati ve Zline a studuite to, co vás bude bavit a nakonec i dobre zivit. Nemusite se bat, ze za tri nebo pet let opustite akademickou püdu a nebudete vedêt, co vlastne múzete s nabytymi znalostmi delat. Technické zamèreni znamená dobrou pracovni budoucnost, bohaté uplatnèni a motivujici platy. Pocitejte s tim, ze se o vás budou prát nejrüznejsi státni, soukromé, velké i malé instituce a nabizet vám spolupráci. Absolventi nasi fakulty nacházeji uplatnèni ve spickovich národnich i nadnárodnich firmach, pripadne zakládaji firmy vlastni. Start vlastniho podnikáni svych absolventu fakulta podporuje prostrednictvim Vedeckotechnického park Informacni a komunika&ni technologie.“

Extrahovaný text na platformě Android:

„Jděte na Fakultu aplikované informatiky Univerzity Tomáše Bati ve Zlíně a studujte to, co vás bude bavit a nakonec i dobře živit. Nemusíte se bát, že za tři nebo pět let opustíte akademickou půdu a nebudete vědět, co vlastně můžete s nabytými znalostmi dělat. Technické zaměření znamená dobrou pracovní budoucnost, bohaté uplatnění a motivující platy. Počítejte s tím, že se o vás budou prát nejrůznější státní, soukromé, velké i malé instituce a nabízet vám spolupráci. Absolventi naší fakulty nacházejí uplatnění ve špičkových národních i nadnárodních firmách, případně zakládají firmy vlastní. Start vlastního podnikání svých absolventů fakulta podporuje prostřednictvím Vědeckotechnického parku Informační a komunikační technologie“

Zhodnocení výsledků:

Obě platformy iOS a web text úspěšně nalezli a extrahovali, ale protože ani jedna z platforem nepodporuje český jazyk, extrahovaný text není správný, na druhou stranu se alespoň trochu podobá vstupnímu obrázku. Naopak na platformě Android je vidno že zafungovala podpora češtiny ze strany Google ML Kit, a text byl extrahován perfektně.

9.4.2 Testovací případ pro Obrázek 9

Extrahovaný text na webové platformě:

„When you come to Tomas Bata University in Zlin; to the Faculty of Applied Informatics, (FAI) ... you can study what interests you - what you enjoy - and, eventually, this will allow you to make a good living. You do not have to worry about leaving the academic sphere in three or five years, not knowing what exactly you can do with the knowledge you have acquired. Its technical focus assures good career prospects, rich employment opportunities, and motivational salaries. FAI, TBU in Zlin graduates find employment in top-flight national and multinational companies; or - as the case may be, establish their own companies. The Faculty's activities are supported by our dedicated CEBIA- TECH, Information and Communication Technologies, Science and Technology Park, which also serves as an incubator for graduates and motivates them to start their own businesses"

Extrahovaný text na platformě iOS:

„When you come to Tomas Bata University in Zlin; to the Faculty of Applied Informatics, (FAI) ... you can study what interests you - what you enjoy - and, eventually, this will allow you to make a good living. You do not have to worry about leaving the academic sphere in three or five years, not knowing what exactly you can do with the knowledge you have acquired. Its technical focus assures good career prospects, rich employment opportunities, and motivational salaries. FAI, TBU in Zlin graduates find employment in top-flight national and multinational companies; or - as the case may be, establish their own companies. The Faculty's activities are supported by our dedicated CEBIA- TECH, Information and Communication Technologies, Science and Technology Park, which also serves as an incubator for graduates and motivates them to start their own businesses"

Extrahovaný text na platformě Android:

„When you come to Tomas Bata University in ZIín; to the Faculty of Applied Informatics, (FAl) ..you can study what interests you - what you enjoy - and, eventually, this will allow you to make a good living. You do not have to worry about leaving the academic sphere in three or five years, not knowing what exactly you can do with the knowledge you have acquired. Its technical focus assures good career prospects, rich employment opportunities, and motivational salaries. FAI, TBU in Zlín graduates find employment in top-flight national and multinational companies; or-as the case may be, establish their own companies. The Faculty's activities are supported by our dedicated CEBIA- TECH, Information and Communication Technologies, Science and Technology Park, which also serves as an ncubator for graduates and motivates them to start their own businesses"

Zhodnocení výsledků:

Zde si všechny tři platformy vedly skvěle. Text je téměř shodný a správně extrahovaný. Jediným rozdílem jsou slova napsané v češtině, zde zase boduje implementace pomocí Google ML Kitu

9.4.3 Testovací případ pro Obrázek 10

Extrahovaný text na webové platformě:

„STUDIUM O FAKULTE VEDA A VYZKUM SPOLUPRACE ABSOLVENT“

Extrahovaný text na platformě iOS:

„STUDIUM O FAKULTE VEDA A VÝZKUM SPOLUPRÁCE ABSOLVENT“

Extrahovaný text na platformě Android:

„STUDIUM O FAKULTĚ VĚDA A VÝZKUM SPOLUPRÁCE ABSOLVENT“

Zhodnocení výsledků:

V tomto případě si všechny tři platformy vedly poměrně dobře, a test dopadl dle očekávání. Všechny platformy zaznamenaly znaky správné, v případě webu byla vynechána veškerá diakritika. U platformy iOS je vidno, že přestože nepodporuje český jazyk, správně detekoval také několik čárek. Platforma Android však text extrahovala bez jediného zaváhání.

9.4.4 Testovací případ pro Obrázek 11

Extrahovaný text na webové platformě:

„STUDY FACULTY RESEARCH AND DEVELOPMENT COOPERATION GRADUATE“

Extrahovaný text na platformě iOS:

„STUDY FACULTY RESEARCH AND DEVELOPMENT COOPERATION GRADUATE“

Extrahovaný text na platformě Android:

„STUDY FACULTY RESEARCH AND DEVELOPMENT COOPERATION GRADUATE“

Zhodnocení výsledků:

U obrázku 11 proběhla extrakce textu naprosto perfektně ve všech případech.

9.4.5 Testovací případ pro Obrázek 12

Extrahovaný text na webové platformě:

„L P vucne pamy texe“

Extrahovaný text na platformě iOS:

„rucné psany text v ceskem jazyce“

Extrahovaný text na platformě Android:

„ruCne text“

Zhodnocení výsledků:

Zde jsou výsledky pro web a Android velmi špatné, což je zapříčiněno tím, že v aktuální implementaci není vyřešeno rozpoznávání psaného textu. V případě Tesseractu, jenž je použit pro webovou platformu by bylo jako vhodné řešení doučení modelu na rozpoznání textu. Pokud se jedná o Google ML Kit, samotný modul, jenž se stará o rozpoznání textu psaný text nepodporuje, zde by bylo teoreticky možné využít modul *Digital Ink Recognition*, který dokáže rozpoznat psaný text v digitální podobě. Avšak je nutno zdůraznit že se jedná pouze o úvahu, jelikož nebylo možné najít dostatek informací pro potvrzení této teorie. Naopak Apple Vision Kit zde překvapil, přestože se jedná o češtinu, která zatím není podporována, text byl alespoň částečně rozpoznán. Tudíž si zde bere vítězství právě iOS.

9.4.6 Testovací případ pro Obrázek 13

Extrahovaný text na webové platformě:

„avd e ten tEXE 1M cfhf/fo'/) /o7n7chz 7“

Extrahovaný text na platformě iOS:

„handwritten text in english language“

Extrahovaný text na platformě Android:

„hardwneen text in cngish language“

Zhodnocení výsledků:

V tomto testu si zaslouženě připsala vítězství platforma iOS, jelikož zvládne rozpoznávat psaný text a zároveň je text napsaný v anglickém jazyce, rozpoznání proběhlo naprosto perfektně. V případě Androidu je zde alespoň náznak rozpoznání, avšak přesto je výsledek poměrně špatný. Pokud se jedná o webovou platformu, ta zde naprosto zklamala, avšak výsledek textu plně odpovídá očekávání.

9.5 Zhodnocení výsledků testování

Výše uvedené testy probíhaly na následujících zařízeních:

- webová platforma - MacBook Pro, Google Chrome 109.0.5414.87 (arm64)
- iOS platforma - IPad Pro 2018, iPadOS 16.3
- Android platforma - Samsung S21+ 5G, Android 13

Téměř všechny výše uvedené testy dopadly dle očekávání. Jedinou výjimkou je testovací případ pro Obrázek 12, ve kterém se zařízení iOS povedlo téměř správně rozpoznat ručně napsaný text, který byl psán v českém jazyce.

Z výsledků je patrné, že pro všechny tři platformy není problém rozpoznat a extrahovat text v anglickém jazyce, který je v digitální podobě. Což přesně odpovídá očekávání, jelikož ve všech případech nástroje Tesseract.js, Google ML Kit a také Apple Vision Framework tuto možnost podporují.

Pokud se jedná o český jazyk, výsledky zde nebyly úplně nejhorší, jelikož je oficiálně podporován pouze nástrojem Google ML Kit, tak jak Tesseract.js, tak Apple Vision Framework dokázali text alespoň částečně rozpoznat.

V případě ručně psaného textu nastává největší problém, zde text dokáže správně extrahovat pouze Apple Vision Framework.

Z výsledků je patrné, že vývoj Capacitor pluginu pro rozpoznávání textu a jeho následná implementace do prototypu aplikace, která je schopna běžet na webové, iOS a Android platformě proběhla nad očekávání dobře. Je také vhodné zmínit, že je možné plugin jako takový ještě vylepšit. Toho lze docílit například předpřipravením vstupních dat ještě před samotným procesem extrakce textu, nebo v případě nástroje Tesseract.js přidáním dalších modelů, které umožní jak rozpoznání psaného textu, tak českého jazyka.

9.6 Alternativní řešení

Jelikož je optické rozpoznávání znaků poměrně populární, zvlášť v dnešní době, je vysoce pravděpodobné že již existují nějaké podobné řešení. Po průzkumu trhu bylo zjištěno, že jisté alternativy opravdu existují. V následujících odstavcích budou popsány a porovnány výhody a nevýhody oproti capacitor pluginu, kterým se zabývá tato bakalářská práce.

cordova-plugin-mobile-ocr

Jedná se o plugin vytvořený pomocí technologie Cordova, která je již poměrně stará, a postupně se používá čím dál tím méně. Problém může také nastat ve zvolených technologiích. Jelikož tento plugin využívá technologii *Mobile Vision*, která je v dnešní době již nepodporovaná. Hlavním rozdílem mezi tímto pluginem a tím, kterým se zaobírá bakalářská práce je především to, že je zde implementace pouze pro platformy iOS a Android, kdežto u nového capacitor pluginu je přidána podpora také pro web. [19][20]

Cap-ML

Tento plugin je už vytvořen pomocí technologie Capacitor, a využívá moderní technologie jak pro Android, tak iOS. Těmi jsou Google ML Kit a Apple Vision Framework. Dalo by se tedy říct, že se jedná o plugin, který je možné zařadit někam mezi předchozí plugin, a plugin kterým se zabývá bakalářská práce. Na jednu stranu používá moderní a velmi kvalitní technologie, avšak na druhou stranu zde stejně jak v dřívějším řešení chybí implementace pro web, což může být v některých případech naprosto zásadní. [21]

10 POROVNÁNÍ TECHNOLOGII PRO ROZPOZNÁNÍ TEXTU

V dnešní době existuje poměrně rozmanité množství technologií, které je možno využít pro rozpoznání textu ať už na fotce, či obrázku. Mohou se lišit jak efektivitou, tak možnostmi, které nabízí. Také je vhodné zmínit, že některé řešení je možné použít pouze pokud má dané zařízení přístup k internetu, a další řešení umožňují provádět rozpoznání přímo na straně zařízení. To může mít za následek například větší velikost aplikace, nebo pomalejší rozpoznání, které může být ovlivněno především rychlostí samotného zařízení. Na druhou stranu je zde jistota že data zůstanou na zařízení a nejsou nikam odesílána.

10.1 Tesseract

Tesseract je technologie zaměřena na OCR neboli optické rozpoznávání znaků, kterou lze použít na různých operačních systémech. Původně byl vyvinut společností Hewlett-Packard jako proprietární software, ale později roku 2005 byl uvolněn jako open-source. Později začal jeho vývoj sponzorovat. [12][13]

Jelikož je v dnešní době Tesseract open-source, poměrně hodně se rozšířil. Je možné se s ním setkat jako s knihovnou pro JavaScript, nebo také Python. Také existuje poměrně hodně projektů, které jsou založeny právě na Tesseractu, avšak upraveny tak, jak to vývojáři zrovna potřebovali.

Jako pozitivum se dá označit právě to, že se jedná o open-source projekt který je naprosto zdarma. Je možné jej různě modifikovat a přizpůsobit si pro konkrétní účely. Dalším pozitivem je možnost využití Tesseractu bez přístupu k internetu. [13]

Jako negativní vlastnost je možné zmínit to, že pokud je použit pro méně kvalitní obrázek či fotku, kvalita rozpoznání rapidně klesá, další nevýhodou je to, že Tesseract si sám obrázek nějak moc neupravuje, proto je vhodné, aby byl optimálněji zpracován ještě před využitím samotného Tesseractu. [13]

10.2 Apple Vision Framework

Vision Framework od firmy Apple je velmi spolehlivý a rychlí nástroj pro rozpoznávání textu z obrázků. Jelikož aplikuje různé algoritmy počítačového vidění pro provádění množství úkolů na vstupních datech, kterými mohou být jak obrázky, tak videa, je možné jej využít na mnohem více úkolů, než je samotné rozpoznání textu. Vision Framework je součástí operačních systémů iOS, iPadOS a také macOS. Z toho vyplývá že je možné jej použít taktéž bez přístupu k internetu, jelikož běží přímo na straně zařízení.[14]

Mezi jeho výhody patří především to, že dokáže provádět širokou škálu úkolů, včetně detekce obličejů, rozpoznávání čárových kódů, QR kódu, a především také textu. Díky použití vlastních modelů Core ML pro řešení problémů jako je například klasifikace nebo detekce objektů, což poskytuje určitou flexibilitu a možnost přizpůsobení.[14]

Nevýhodou je potom zejména právě to, že jelikož se jedná o produkt od Apple, je omezen pouze na platformy, které jsou taktéž od Apple a není dostupný ani pro Android, ani pro Windows. Další nevýhodou může být fakt, že přesnost a rychlost Vision Frameworku může záviset na zařízení, na kterém je použit, jelikož využívá prostředky daného zařízení. Avšak při vlastním testování jsem se s žádným problémem nesetkal, a samotné rozpoznání textu probíhalo velmi rychle. [14][15]

10.3 Google ML Kit

Google ML Kit je mobilní SDK, které přináší odborné znalosti a technologie Googlu v oblasti strojového učení do aplikací pro Android a iOS. Jedná se o velmi výkonné a zároveň snadno použitelné řešení pro vývojáře. [16]

ML Kit poskytuje velké množství funkcí, jako je detekce obličejů, skenování čárových kódů, vyhledávání významných bodů v obrázcích a v neposlední řadě také rozpoznání textu. Podporuje také vlastní modely TensorFlow Lite, což znamená že je možné pomocí ML Kitu přenést vlastní modely strojového učení přímo do mobilních aplikací. [16]

Jako klíčovou výhodu technologie Google ML Kit, je bezpochyby jeho schopnost pracovat jak online, tak offline. Jde především o funkce jako jsou rozpoznání textu, detekce obličejů a skenování čárových kódu. [16]

Mezi nevýhody lze zařadit například potřebu připojení k internetu pro některé funkce.

10.4 Amazon Textract

Amazon Textract je služba zaměřena na strojové učení, která automaticky extrahuje text, ruční psaní a data ze skenovaných dokumentů. Jde o něco dál než běžné rozpoznávání znaků a dokáže identifikovat, porozumět a extrahovat data z formulářů a tabulek. [17]

Tuto službu je možné použít pouze online, jelikož je poskytována jako služka v cloudu od společnosti AWS neboli Amazon Web Services. [17]

Mezi výhody technologie Textract patří jednoduchost použití a velmi solidní rozpoznání textu. Může se jednat o tištěný text, text psaný rukou a také o data z množství různých dokumentů. Lze ho také integrovat s Amazon Augmented AI pro přidání lidských recenzí ke kontrole modelů a ověření citlivých dat. [17]

Hlavni nevýhodou je to, že pro fungování vyžaduje přístup k internetu. To je zapříčiněno především tím, že tato technologie běží v Cloudu a nikoli lokálně na použitém zařízení. [17]

10.5 Microsoft Azure Computer Vision

Microsoft Azure Computer Vision je API, která je součástí Azure Cognitive Services, která poskytuje vývojářům, přístup k pokročilým algoritmům pro zpracování obrázků a získávání informací.

Dokáže z obrázků extrahovat velké množství informací pro kategorizaci a zpracování vizuálních dat. Může analyzovat obsah několika způsoby, včetně označování obrázků, extrakce textu pomocí optického rozpoznávání znaků a také dokáže například rozpoznávat obličeje. Nabízí také funkce pro analýzu nejen obrázků, ale také prostorovou analýzu a vlastní klasifikaci a detekci objektů. [18]

Mezi pozitiva patří široká škála funkcionalit, což ho činí všestranným nástrojem pro různé použití. Nabízí také předpřipravené modely, což zjednodušuje proces pro vývojáře bez rozsáhlých zkušeností v oblasti strojového učení. Další výhodou je například podpora mnoha jazyků a psacích stylů, což je velmi užitečné právě pro rozpoznávání textu. [18]

Mezi negativa lze zařadit nutnost připojení k internetu pro své fungování. Může být také méně efektivní při zpracování obrázků, které mají nízkou kvalitu, nebo obsahují velmi složitý vizuální obsah. [18]

11 TVORBA PROTOTYPU APLIKACE S VYUŽITÍM VYTVOŘENÉHO CAPACITOR PLUGINU

Pro tvorbu prototypu aplikace byly zvoleny frameworky Ionic a Angular. Díky využití těchto frameworků je možný snadný vývoj aplikace, která může běžet jak na webu, tak Androidu a iOS, případně iPad OS.

Framework Ionic umožňuje jednoduchou, a přitom velmi efektivní tvorbu uživatelského rozhraní, jelikož všechny potřebné komponenty jsou již vytvořené. Není tedy nutné se zaobírat stylováním a responzivitou. Většinu práce zde odvede právě Ionic.

11.1 Nástroje potřebné pro vývoj aplikace

Pro vývoj aplikace je nutné mít nainstalováno několik knihoven či frameworků. Tvorba aplikace probíhá na zařízení s operačním systémem macOS, takže tomu budou také odpovídat následující nástroje, popřípadě také použité příkazy pro terminál. Při vývoji na jiných platformách jako je Windows či Linux se postup může lehce lišit.

11.1.1 Homebrew

Homebrew je správce balíčku pro operační systém macOS. Balíčky instaluje do vlastního adresáře a na jejich soubory odkazuje pomocí symlinku do „/opt/Homebrew“, pokud se jedná o mac procesorem od Apple.

Je možné jej nainstalovat pomocí příkazu:

- */bin/bash -c „$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)“*

11.1.2 Node.js

Node.js je open-source, multiplatformní běhové prostředí pro spouštění kódu psaného pomocí JavaScriptu. Většinou se používá pro programování na straně serveru, avšak my jej budeme potřebovat také pro tvorbu hybridní mobilní aplikace.

Je možné jej nainstalovat pomocí příkazu

- *brew install node*

11.1.3 Node Package manager

Node Package manager – neboli NPM je již součástí Node.js, takže není nutné jej instalovat zvlášť.

11.1.4 Ionic command line interface

Ionic command line interface je hlavní nástroj používaný při vývoje Ionic aplikace. Sdružuje řadu různých nástrojů pod jedním rozhraním. Obsahuje řadu příkazů, které jsou klíčové pro vývoj Ionic aplikace, jako například „start“, „serve“, a „run“.

Je možné jej nainstalovat pomocí NPM a příkazu:

- *npm install -g @ionic/cli*

11.1.5 Capacitor

Capacitor je open-source nativní prostředí pro tvorbu multiplatformních aplikací. Umožňuje použít nativní funkce systémů jako je iOS a Android při tvorbě aplikace.

Je možné jej nainstalovat pomocí příkazů:

- npm install @capacitor/core
- npm install -D @capacitor/cli

11.2 Vytvoření prototypu aplikace

Pro vytvoření prototypu aplikace je možné využít přímo framework Ionic, který umožní vygenerovat veškerou potřebnou strukturu aplikace, včetně předpřípravách obrazovek, do kterých je poté možné vkládat další komponenty.

Při generování nového Ionic projektu je požadováno, aby v něm byl použit Angular, Capacitor, SCSS pro stylování.

Dále je možné zvolit různé šablony, například:

- „blank“ – jedná se o prázdnou šablonu
- „tabs“ – jedná se o šablonu se třemi různými taby, mezi kterými je možné libovolně přepínat pomocí tlačítek na dolní straně obrazovky.
- „super“ – tato šablona obsahuje škálu různých komponent

Zvolená šablona bude teda „blank“, jelikož v prototypu aplikace nebude nutné použít ani taby, ani další složitější komponenty.

Pro dosažení výše popsaných vlastností je možné použít následující příkaz v terminálu:

- *ionic start „application“ blank –type=angular –style=scss“*

11.3 Struktura aplikace

Struktura aplikace popisuje organizaci a soubory, popřípadě složky, které se vyskytují v projektu. Základní strukturu aplikace, která je vytvořena pomocí frameworků Ionic a Angular, může vypadat například takto:

- /node_modules – složka, která obsahuje veškeré knihovny a závislosti které jsou nezbytné pro běh aplikace, popřípadě mohou zajišťovat nějakou další funkcionalitu
- /src – jde o hlavní složku, která obsahuje zdrojový kód aplikace. Běžně se zde nachází veškeré komponenty, služby, moduly a styly aplikace
- /www – tato složka může obsahovat statické soubory, jako například logo, různý obrázky a tak podobně
- package.json – tento soubor obsahuje seznam všech závislostí a konfigurací potřebných pro běh a vývoj aplikací, jako jsou například příkazy pro spuštění, sestavení, debuggování, popřípadě další funkce. Jde také o soubor, který je využíván nástrojem npm pro instalaci balíčků a závislostí projektu.
- capacitor.config.json – jedná se o hlavní konfigurační soubor pro Capacitor
- angular.json – jedná se o konfigurační soubor pro framework Angular

11.4 Ukázka zdrojového kódu aplikace

Následující zdrojový kód se nachází v projektu testovací aplikace, která bude využita pro otestování vyvíjeného capacitor pluginu. Následující kód je zobrazením toho, jak testovací aplikace vypadá ve chvíli, kdy je psán tento text. Výsledná aplikace se může v některých částech lišit, avšak funkcionalita zůstane velmi podobná.

11.4.1 Soubor home.page.html

Následující soubor se stará o zobrazení několika jednoduchých komponent, jedná se především o naskenovanou fotku, popřípadě obrázek vybraný z galerie, níže je text, který se povedlo vytáhnout z výše zmíněného obrázku, a ve spodní části je možné vidět tlačítka, které umožňují jak výběr obrázku z galerie, tak využití fotoaparátu zařízení pro zaznamenání libovolné fotografie.

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Ionic OCR plugin preview
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="false" class="ion-padding">
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Ionic OCR preview</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-grid class="ion-no-padding">
    <ion-row class="ion-justify-content-center ion-align-items-center">
      <ion-col size-md="6" class="ion-text-center ion-padding">
        <ion-img
          src="{{imageBase64$.value ? imageBase64$.value : placeholderImage}}"
          class="ion-margin-bottom"
        ></ion-img>
        <ng-container *ngIf="isLoading$.value; else loaded">
          <ion-spinner></ion-spinner>
        </ng-container>
        <ng-template #loaded>
          <ion-text>
            <p>{{result$.value}}</p>
          </ion-text>
        </ng-template>
      </ion-col>
    </ion-row>
  </ion-grid>

</ion-content>

<ion-footer [translucent]="true">
  <ion-grid class="ion-margin-vertical">
    <ion-row class="ion-justify-content-center">
      <ion-button (click)="chooseImage()">Choose Image</ion-button>
      <ion-button (click)="captureImage()">Capture Image</ion-button>
    </ion-row>
  </ion-grid>
</ion-footer>
```

11.4.2 Soubor home.page.ts

Uvedený kód je komponenta Angularu představující domovskou stránku. Jsou zde importovány potřebné závislosti, včetně knihovny RxJS, služby PhotoService a vyvíjené knihovny CapOcr. Komponenta dokáže uchovávat a aktualizovat výsledek rozpoznání textu, také uchovává base64 data a stav načítání. Obsahuje metody pro výběr nebo zachycení obrázků pomocí služby PhotoService a provedení detekce textu pomocí knihovny CapOcr. Text, který se podaří rozpoznat je následně zobrazen spolu s vybraným, případně zachyceným obrázkem na domovské stránce

```
import {Component} from '@angular/core';
import {BehaviorSubject} from "rxjs";
import {PhotoService} from "../services/photo.service";
import {CapOcr} from "cap-ocr";

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {
  result$ = new BehaviorSubject('placeholder …');
  imageBase64$ = new BehaviorSubject('');
  isLoading$ = new BehaviorSubject(false);

  public placeholderImage = 'https://placehold.jp/200x150.png';


  constructor(private _photoService: PhotoService) {
  }

  async chooseImage() {
    const image = await this._photoService.pickImageBase64();
    this.getText(image);
  }

  async captureImage() {
    const image = await this._photoService.captureImageBase64();
    this.getText(image);
  }

  getText(image: string | undefined) {
    if (image) {
      this.imageBase64$.next(this.addBase64Prefix(image));

      this.isLoading$.next(true);
      CapOcr.detectText({imageBase64: image})
        .then((result: any) => {
          if (result) {
            const extractedText = result.value;
            console.log('extracted text: ', extractedText);
            this.result$.next(extractedText);
            this.isLoading$.next(false);
          }
        })
    }
  }

  addBase64Prefix(image: string) {
    return '' + image;
  }
}
```

11.4.3 Soubor photo.service.ts

Následující kód je serviska Angularu, která využívá capacitor plugin „Capacitor Camera“. Poskytuje dvě metody, jež jsou „captureImageBase64()“, která je schopna zaznamenat obrázek pomocí fotoaparátu zařízení, zatímco „pickImageBase64()“ umožňuje vybrat obrázek z fotogalerie. Obě metody následně vracejí reprezentaci base64 zachyceného obrázku. Serviska zapouzdřuje funkce fotoaparátu, což usnadní další integraci funkcionality v projektu.

```
import {Injectable} from '@angular/core';
import {Camera, CameraResultType, CameraSource} from "@capacitor/camera";

@Injectable({
  providedIn: 'root'
})
export class PhotoService {

  public async captureImageBase64() {
    const capturedPhoto = await Camera.getPhoto({
      resultType: CameraResultType.Base64,
      source: CameraSource.Camera,
      allowEditing: true,
      quality: 100
    });

    return capturedPhoto.base64String;
  }

  public async pickImageBase64() {
    const image = await Camera.getPhoto({
      resultType: CameraResultType.Base64,
      source: CameraSource.Photos,
      allowEditing: true,
      quality: 100
    })

    return image.base64String
  }
}
```

11.5 Přidání capacitor pluginu

V této chvíli je již připravena základní implementace Capacitor pluginu, která splňuje alespoň základní požadavky pro funkcionalitu.

To znamená, že vyvíjený Capacitor plugin dokáže přijmout data v podobě base64 textového řetězce, zpracovat je a následně v obrázku či pořízené fotografii vyhledá text. Ten následně vrátí zpět do aplikace, ze které je funkce pluginu volána.

Pro přidání pluginu do aplikace existuje více způsobu. První je instalace pomocí NPM, jelikož je ale plugin zatím v naprosto základní verzi, nebyl veřejně publikován pro NPM. To znamená že jej zatím není možné stáhnout pomocí node package manageru.

Z tohoto důvodu je využito druhé možnosti, a to přímé instalace balíčku pomocí NPM, avšak z lokálního repositáře.

To je provedeno pomocí následujícího příkazu:

- *npm install „path/to/package“*

11.6 Ukázka aplikace na simulovaném zařízení iOS

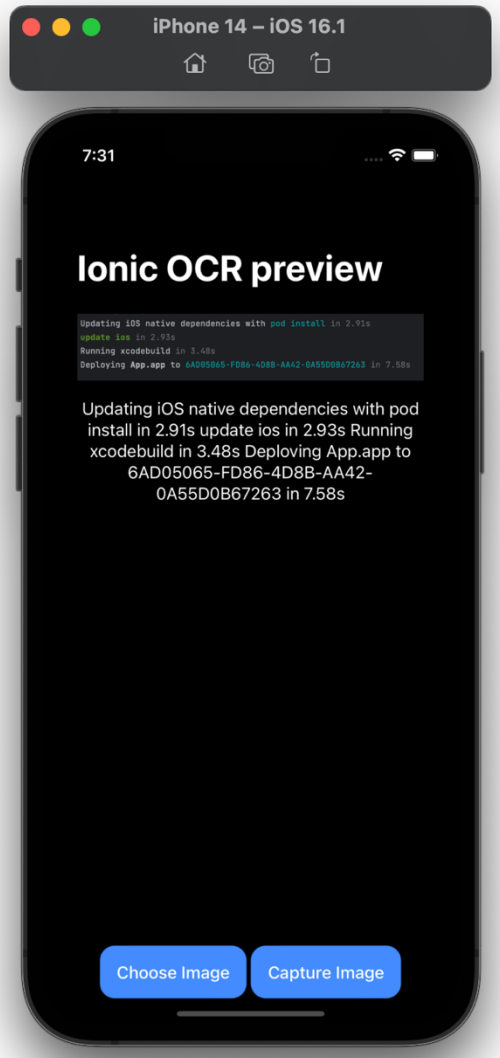
Domovská obrazovka testovací aplikace v defaultním stavu

- Nebyl vybrán ani zaznamenán žádný obrázek
- Text je nyní v původním stavu

Obrázek 12 - ukázka aplikace na simulátoru 1

Domovská stránka aplikace po rozpoznání textu

- Byl zvolen obrázek z galerie
- Je zobrazen vybraný obrázek
- Aplikace za pomoci capacitor pluginu pro rozpoznání textu úspěšně našla text
- Text byl zaměněn za nový, úspěšně extrahovaný z obrázku



Obrázek 13 - ukázka aplikace na simulátoru 2

ZÁVĚR

Tato bakalářská práce se zaměřuje na návrh, popis a implementaci Capacitor pluginu pro detekci textu za použití kamery mobilního zařízení a z dříve pořízených fotografií. Obsahuje detailní popis procesu vývoje mobilní aplikace s využitím frameworku Ionic/Capacitor a zhodnocuje výhody a nevýhody tohoto přístupu v porovnání s alternativními řešeními.

Práce rovněž zahrnuje analýzu technologií vhodných pro vývoj Capacitor pluginu a popisuje možnosti jeho využití. Praktická část obsahuje vytvoření prototypu mobilní aplikace, ve které je implementován navržený Capacitor plugin. Implementace byla provedena pomocí technologií TypeScript/Angular a frameworku Ionic. Aplikace spolu s Capacitor pluginem byla následně testována jak na simulovaném, tak na reálném zařízení.

Během vývoje byl kladen důraz na pečlivou implementaci a dodržování pravidel použitých technologií a frameworků. Tento přístup nejen zajišťuje kvalitu finálního produktu, ale také umožňuje jeho snadné a efektivní rozšíření v budoucnu.

Proces vývoje a implementace byl patřičně zdokumentován. Capacitor plugin byl zveřejněn na platformě GitHub a také pro node package manager, což zajišťuje jeho snadnou dostupnost a rozšiřitelnost pro vývojáře po celém světě.

Výsledkem práce je plně funkční Capacitor plugin, který splňuje všechna požadovaná kritéria. Plugin nabízí efektivní a uživatelsky přívětivé řešení pro detekci textu pomocí kamery mobilního zařízení nebo z dříve pořízené fotografie. Přináší tak nový příspěvek k aktuálnímu stavu technologií a otevírá nové možnosti pro budoucí vývoj a inovace v této oblasti.

Je taktéž vhodné zmínit, že během dvou dnů od publikace pro node package manager dosáhl plugin přes sto stažení. To zdůrazňuje poptávku po funkcích optického rozpoznávání znaků v dnešních aplikacích.

Jelikož se projevil alespoň nějaký zájem o tento plugin, jeho vývoj bude pokračovat jako open-source projekt. Již nyní existuje řada funkcionalit, které mohou v budoucnu zjednodušit a zefektivnit vývoj aplikací, kde se využívá rozpoznávání textu.

SEZNAM POUŽITÉ LITERATURY

[1] Ionic Framework - The Cross-Platform App Development Leader [online]. Dostupné z: https://ionicframework.com/docs

[2] [Příloha P2] Building Cross-Platform Apps with Capacitor. 17 [cit. 2023-05-18].

[3] [online]. Copyright © 2023 The Apache Software Foundation, Licensed under the [cit. 20.05.2023]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/index.html

[4] Getting started with the web - Learn web development | MDN. [online]. Copyright ©1998 [cit. 20.05.2023]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web

[5] [online]. Dostupné z: https://www.makeuseof.com/hybrid-app-development-best-frameworks/

[6] Creating Capacitor Plugins | Capacitor Documentation. Capacitor - build cross platform apps with the web [online]. Dostupné z: https://capacitorjs.com/docs/v3/plugins/creating-plugins

[7] Ionic vs. React Native: Performance Comparison - Ionic Blog. Ionic: Enterprise App Platform [online]. Copyright © [cit. 20.05.2023]. Dostupné z: https://ionic.io/blog/ionic-vs-react-native-performance-comparison

[8] Stack Overflow Developer Survey 2022. [online]. Dostupné z: https://survey.stackoverflow.co/2022/

[9] Ionic vs. Xamarin: Mobile App Development Frameworks. Ionic: Enterprise App Platform [online]. Copyright © [cit. 20.05.2023]. Dostupné z: https://ionic.io/resources/articles/ionic-vs-xamarin

[10] Ionic vs Flutter | Flutter Alternative & Performance Comparison. Ionic: Enterprise App Platform [online]. Copyright © [cit. 20.05.2023]. Dostupné z: https://ionic.io/resources/articles/ionic-vs-flutter-comparison-guide

[11] Comparison to Java | Kotlin Documentation. Kotlin Programming Language [online]. Dostupné z: https://kotlinlang.org/docs/comparison-to-java.html

[12] Smith, R. (2007). An Overview of the Tesseract OCR Engine. In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol. 2 (pp. 629-633).

[13] Tesseract.js | Pure Javascript OCR for 100 Languages!. Tesseract.js | Pure Javascript OCR for 100 Languages! [online]. Dostupné z: https://tesseract.projectnaptha.com/

[14] Vision Framework for iOS: The Basics | by Ilana Concilio | Academy@Eldorado-CPS | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://medium.com/academy-eldoradocps/vision-framework-for-ios-an-introduction-78d02c3e1ef

[15] Apple Developer [online]. Dostupné z: https://developer.apple.com/documentation/vision

[16] ML Kit | Google for Developers. Google for Developers - from AI and Cloud, to Mobile and Web [online]. Dostupné z: https://developers.google.com/ml-kit

[17] Intelligently Extract Text & Data with OCR - Amazon Textract - Amazon Web Services. Cloud Computing Services - Amazon Web Services (AWS) [online]. Copyright © 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved. [cit. 20.05.2023]. Dostupné z: https://aws.amazon.com/textract/

[18] Azure Cognitive Service for Vision with OCR and AI | Microsoft Azure. Object moved [online]. Copyright © Microsoft 2023 [cit. 20.05.2023]. Dostupné z: https://azure.microsoft.com/en-us/products/cognitive-services/vision-services

[19] cordova-plugin-mobile-ocr - npm. npm [online]. Dostupné z: https://www.npmjs.com/package/cordova-plugin-mobile-ocr#cordova-plugin-mobile-ocr

[20] GitHub - NeutrinosPlatform/cordova-plugin-mobile-ocr: A cordova plugin that can accept image URI or Base64 data and returns the text present in the image as string without need for network. You can also try :- https://github.com/NeutrinosPlatform/cordova-plugin-ml-text. GitHub: Let's build from here · GitHub [online]. Copyright © 2023 GitHub, Inc. [cit. 21.05.2023]. Dostupné z: https://github.com/NeutrinosPlatform/cordova-plugin-mobile-ocr

[21] GitHub - bendyworks/cap-ml: Text Detection Capacitor Plugin. GitHub: Let's build from here · GitHub [online]. Copyright © 2023 GitHub, Inc. [cit. 21.05.2023]. Dostupné z: https://github.com/bendyworks/cap-ml

[22] Fakulta aplikované informatiky - Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky - Univerzita Tomáše Bati ve Zlíně [online]. Copyright © 2023 Univerzita Tomáše Bati ve Zlíně [cit. 22.05.2023]. Dostupné z: https://fai.utb.cz/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

NPM node package manager

PWA progressive web application

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

P1 CD s diplomovou prací a elektronickou knihou *Ebook_Building cross-platform apps with Capacitor*

P2 CD obsahující zdrojové kódy capacitor pluginu a testovací aplikace