Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Vývoj mobilní aplikace pro sportovní portál – HockeyOnline

Bakalářská práce

Autor: David Podzimek
Studijní obor: ai3

Vedoucí práce: Ing. Michal Macinka

Hradec Králové Březen 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29.4.2021 David Podzimek

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Michalu Macinkovi za metodické vedení práce. Dále pak své rodině za bezmeznou podporu.

Anotace

Práce se zabývá volbou technologie a následným vývojem mobilní aplikace HockeyOnline, která má být rozšířením webové aplikace HockeyOnline – ta slouží především k vyhodnocování a zobrazování výsledků turnajů pořádaných klubem HC Slavia Hradec Králové, stejný účel by měla mít i mobilní aplikace. V první části práce jsou porovnány technologie a na základě požadavků na technologii i mobilní aplikaci je vybrána technologie React Native. Druhá část práce je věnována vývoji aplikace a API, které je vyvíjeno současně a poskytuje data pro fungování mobilní aplikace. Během vývoje je kladen důraz na znuvupoužitelnost vytvořených komponent, aby se aplikace mohla v budoucnosti dále rozrůstat. API i mobilní aplikace jsou v průběhu vývoje testovány, aplikace je nasazena na Google Play.

Annotation

Title: Mobile Application Development for Sports Portal – HockeyOnline

The thesis deals with the choice of technology and development of the mobile application HockeyOnline, which should be an extension of the web application HockeyOnline - it is mainly used to evaluate and display the results of tournaments organized by the club HC Slavia Hradec Králové, the same purpose should have the mobile application. In the first part, the technologies are compared, and based on the requirements for the technology and the mobile application the React Native technology is selected. The second part of the thesis is devoted to the development of the application and the API that is developed simultaneously. API provides data for the functioning of the application. During the development process, emphasis is given to the usability of the developed components so that the application can grow in the future. Both the API and the app are tested during development, and finally, the app is deployed on Google Play.

Obsah

1 Úvod

V dnešní době digitálních technologií, kdy má již téměř každý k dispozici mobilní telefon s možností připojení k internetu, se toho stále více organizací snaží využít a umožnit přístup ke svému obsahu zaměstnancům, uživatelům, klientům i zákazníkům přímo na mobilním zařízení. Na rozdíl od počítače nebo laptopu mobilní telefon či tablet mívají jejich uživatelé vždy po ruce, využívají jej jak pro komunikaci, tak i pro zábavu. Obsahem můžou být články, informace, nabídky služeb či produktů, ale i firemní systémy, které mají za cíl zvýšit produktivitu svých zaměstnanců nebo jim práci maximálně zjednodušit a nabídnout přístup odkudkoliv.

Jsou dvě základní cesty, jak toto umožnit. Webové stránky optimalizované pro mobilní zařízení či webové aplikace, nebo mobilní aplikace pro konkrétní operační systém. Obecně platí, že vývoj mobilní aplikace je časově i finančně náročnější, na druhou stranu mobilní aplikace poskytuje více možností, a to i přes výrazný pokrok u technologií pro tvorbu webových aplikací na principu PWA (Progresivní webová aplikace) či WebAssembly.

Mobilní aplikace dnes mohou najít uplatnění v mnoha oblastech. Jednou z nich je sport, aktuálně velmi úzce spjatý s informačními technologiemi. Je tomu tak díky velkému množství dat, které se dají o sportovních aktivitách zaznamenávat. Tato data mohou být zpeněžena například sázkovými kancelářemi, mohou poskytnout sportovcům informace, v čem se zlepšovat, nebo být využita k vyhodnocování výkonů jednotlivců i skupin, ba dokonce i celých sportovních soutěží. Tato data mohou být zapsána na sportovištích více či méně složitými zařízeními; aktivity, jako je běh a cyklistika, jsou zaznamenávány přímo mobilními zařízeními, nejčastěji telefonem či chytrými hodinkami. Ve většině případů jsou informace po zanesení uloženy na servery v cloudu, kde jsou i zpracovány a uvedeny do kontextu s dalšími naměřenými daty. Mobilní aplikace i webové stránky pak slouží většinou k zobrazování těchto dat.

Touto cestou se vydal i klub pozemního hokeje v Hradci Králové, který při organizování svých sportovních akcí, především dětských a mládežnických turnajů, začal narážet na problémy u manuálního zpracovávání výsledků utkání. Akce se díky velké oblibě hradeckých turnajů a dobré organizaci začaly rozrůstat a při více jak padesáti utkáních

denně na třech hřištích zároveň začaly vznikat chyby, a to i s pomocí zpracování dat v excelovských tabulkách. Nebylo také možné sledovat jiné statistiky, například střelce gólů, vyloučení a další. V roce 2019 spustil klub první verzi vlastní webové aplikace zvanou HockeyOnline, která pomohla vyhodnocovat pořadí turnajů i další statistiky, zadávat výsledky jednotlivě pro každé hřiště a zobrazovat data na informačních obrazovkách téměř v reálné čase. Z interní aplikace se tak stala veřejná. HockeyOnline se později začal používat i pro menší akce pořádané klubem. V dalších verzích byly přidány i funkce pro tvorbu článků a komentářů jednotlivých turnajů i celých utkání, přibyly popisky a fotografie týmů včetně soupisek. Informační tabule již nestačily, a tak byly do obsahu na těchto obrazovkách přidány QR kódy, které po naskenování fotoaparátem mobilního zařízení otevřely webovou stránku s detailem turnaje obsahující veškeré důležité informace. I přes dosavadní optimalizaci pro mobilní zařízení není přehlednost zobrazených informací ani funkcionalita zcela ideální, navíc díky zvýšenému provozu na síti byl klub ve spolupráci s HKFree nucen několikanásobně navýšit v místech konání akcí kapacitu wifi sítě, která je i přes značné investice a ne vždy, kvůli špatnému připojení k internetu v místech konání akce, dostačující. Zajištění kvalitního a stabilního připojení k internetu je žádoucí i z důvodu živého vysílání.

Klub HC Slavia Hradec Králové tak nyní chce rozšířit HockeyOnline o mobilní aplikaci, která by pomohla tuto situaci vyřešit a umožnit tak další expanzi svých akcí. A právě na vývoj takové mobilní aplikace pro klub pozemního hokeje se zaměří tato bakalářská práce.

2 Cíl práce

V práci bude důležité zamyslet se nad technologiemi pro vývoj mobilních aplikací a vybrat tu nejvhodnější, v níž bude možné aplikaci realizovat. Bude nutné zjistit, která technologie umožňuje rychlý vývoj škálovatelných aplikací, aby bylo možné aplikaci do budoucna případně rozšiřovat. Nezbytné bude vyvinout a nasadit službu API, která bude zprostředkovávat aplikaci data z již existující MSSQL databáze. REST API bude tvořeno kvůli kompatibilitě s již existující webovou aplikací v technologii ASP.NET Core a vybraná technologie musí umět s touto API komunikovat. Poté bude třeba hotové API uvést do provozu a nasadit jej na hostingovou službu.

Cílem je vytvořit jednoduchou, vícejazyčnou (podpora češtiny a angličtiny) mobilní aplikaci pro přehledné zobrazování článků, akcí, turnajů a jejich výsledků. Výsledná aplikace by měla být připravena k publikování na Google Play.

3 Mobilní platforma

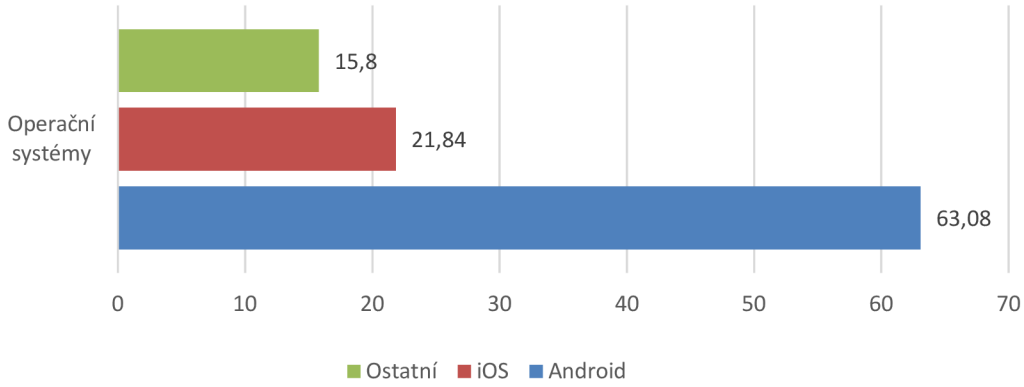
Mobilní platformy představují kombinaci hardwaru, jako je například chytrý telefon, chytré hodinky nebo tablet, a softwaru, operačního systému určeného pro tento hardware. Jednotlivé platformy se od sebe liší funkcemi, které poskytuje operační systém uživateli, rozhraním mobilního operačního systému pro vývojáře mobilních aplikací a také aplikacemi třetích stran podporovanými danou platformou.

3.1 Mobilní aplikace

Mobilní aplikace jsou specializovaným typem aplikací optimalizovaných pro mobilní zařízení, nejčastěji telefony nebo tablety. Podporuje tedy například dotykový display, jímž většina mobilních zařízení disponuje. Mobilní aplikace mají přizpůsobené uživatelské rozhraní tak, aby podporovaly specifické rozlišení mobilních zařízení. Zásadní pro vývoj aplikací je také podpora senzorů, jimiž je mobilní zařízení vybaveno, nejčastěji se jedná o GPS senzory, biometrické senzory, používané pro ověření identity uživatele či senzory, které mohou být užívány i k měření životních funkcí (např. pulzní oxymetr [1] používaný pro měření srdečního tepu a hladiny kyslíku v krvi); vybavení mobilního senzorického zařízení se liší na základě konkrétního modelu a konkrétního výrobce. Dále mají mobilní aplikace, z důvodu omezené kapacity dostupných akumulátorů, optimalizovanou spotřebu energie [2] tak, aby se prodloužila doba užívání daného zařízení s mobilním operačním systémem na maximum.

3.2 Operační systémy pro mobilní platformy

Na současném trhu mají největší podíl zastoupení, jak je vidět v grafu níže, operační systémy Android a konkurenční iOS, oba podporují dotykový display, senzory mobilního zařízení a slot pro SIM kartu i GSM modul sloužící k připojení do sítě operátora.



Graf 1 – Procentuální podíl zastoupení na trhu leden 2009 – únor 2021. Zdroj: [3]

3.3 Operační systém Android

Operační systém Android, který vznikl původně pro fotoaparáty [4], má na poli mobilních operačních systémů největší zastoupení. Je to dáno především tím, že na rozdíl od konkurenčních operačních systémů může běžet téměř na jakémkoliv mobilním zařízení nehledě na výrobce.

Startup společnost Android Inc. byla založena roku 2003 a v roce 2005 byla odkoupena společností Google Inc. Google v roce 2007 podle [5] zakládá organizaci Open Headset Alliance, složenou z výrobců mobilních zařízení, operátorů a dalších technologických firem, společně se tak podílejí jak na vývoji nadstaveb a optimalizací systému pro svá zařízení, tak na vývoji samotného systému, který je postavený na linuxovém jádře. Android v současné době běží na telefotech, tabletech, televizích, autorádiích, nositelné elektronice jako jsou například hodinky, zařízení pro virtuální realitu a dalších. Android je open source, což je kvůli své otevřenosti a možnosti úprav, a to ze strany výrobců i uživatelů, výhodou i nevýhodou zároveň. Aktualizace nejsou vydávané centrálně, ale každým výrobcem zvlášť. Nevýhodou takto otevřeného a univerzálního systému je, že ne vždy jde správně optimalizovat tak, aby poskytl úplný komfort svým uživatelům, tento problém nastává především u levnějších mobilních zařízení, která nemají dostatečnou podoboru výrobce či dostatečně výkonný hardware [6]. Oficiální službou pro instalaci aplikací je Google, avšak existují i další nezávislé zdroje aplikací, někteří výrobci dokonce sami aplikace poskytují a prodávají.

Jak bylo řečeno výše, Android je u mobilních aplikací nejvyužívanějším operačním systémem. Proto je právě Android primární cílovou platformou pro mobilní aplikaci HockeyOnline.

3.3.1 Vývoj aplikací pro Android

Oficiálními podporovanými jazyky pro Android jsou Java a Kotlin a oficiálním vývojovým prostředím je Android Studio, které nahradilo vývojářské prostředí Eclipse. Android Studio je multiplatformní, lze spustit na operačních systémech Windows, Linux, iOS i Chrome OS. Pro publikování aplikací na oficiálním obchodu je nutný vývojářský účet na Google Play, jednorázový poplatek je 25 dolarů [7].

3.4 Operační systém iOS

Na rozdíl od Androidu je iOS v některých ohledech více optimalizovaný. Je to dáno tím, že iOS vyvíjí pouze jedna firma a pouze pro svá zařízení iPhone a iPad. Celý systém je také mnohem více uzavřený, což jej v konečném důsledku činí bezpečnějším. První verze byla vydána v roce 2007 pro iPhone. Instalace aplikací je možná pouze z oficiální služby App Store [8], kde aplikace podléhají mnohem větší kontrole než u konkurenčního Androidu.

3.4.1 Vývoj aplikací pro iOS

Oficiálně podporovanými jazyky pro vývoj aplikací spustitelných na iOS jsou Swift a Objective-C [9], oficiální vývojové prostředí je XCode, které je možné spustit pouze na macOS. Vývojářský účet je v podobě předplatného – Apple Developer Program stojí 99 dolarů ročně, Apple Developer Enterprise Program je za 299 dolarů ročně [10].

4 Požadavky

Nová mobilní aplikace by měla být rozšířením pro již funkční webový portál HockeyOnline – jedná se o webovou aplikaci klubu pozemního hokeje Slavia Hradec Králové, která původně vznikla jako jednoduchá aplikace k zobrazení dat z největšího mládežnického turnaje ve střední Evropě. HockeyOnline byl postupně rozšiřován a v září 2019 se stal celoklubovou aplikací, která krom vyhodnocování turnajových dat nabízí i evidenci členů, správu a nominace na klubové akce, hromadnou korespondenci a další.

Mobilní aplikace HockeyOnline by neměla nabízet celou funkcionalitu, nýbrž by měla být pouze odlehčenou, přesto v případě dalšího růstu snadno škálovatelnou, verzí webové aplikace.

Aplikace by měla implementovat především část týkající se článků a turnajových dat. Webová aplikace sice zobrazení dat již nabízí, ale pro mobilní zařízení není zcela optimální. Primárním úkolem aplikace je komunikace s API (Application Programming Interface) a následné zobrazení dat.

4.1 Funkční požadavky

Konkrétní požadavky na funkcionalitu aplikace:

- ukládání dat pro případ přechodu zařízení do módu offline,
- zobrazení dat článků, nadcházejících, uplynulých a v současnosti probíhajících turnajů, dále pak detailů daného turnaje, účastníků, soupisů jednotlivých utkání a průběžného pořadí ve skupinách i celkově,
- možnost uživatele označit si sledované týmy,
- možnost automaticky přepínat mezi cloudovým a lokálním serverem – není implementováno v první verzi,
- naskenování QR kódu:
    - na akcích, kde má být aplikace především využívána, generuje webová aplikace QR kódy, které jsou rozmístěné v rámci areálu, v němž se akce koná; data z QR kódu odkazují na detailní informace o akci,
- nastavení jazyka podle nastavení systému.

4.2 Non-funkční požadavky

Požadavky na kvalitu a provedení aplikace:

- podpora REST ASP.NET API:
  - webová aplikace včetně API je již napsána v ASP.NET, je tedy nutné, aby bylo možné používat stávající API,
- možnost aplikaci dále rozvíjet:
  - stejně jako se vyvíjí webová aplikace, je důležité, aby bylo do budoucna možné rozšířit i mobilní aplikaci, přidávat další funkcionalitu či rozšiřovat a případně upravovat další,
- minimální vytížení připojení k internetu:
  - během akcí, na kterých se bude aplikace primárně používat, je hodně uživatelů fyzicky na jednom místě, je tedy nutné optimalizovat komunikaci natolik, aby bylo velké množství spuštěných aplikací v jedné síti a příliš nevytěžovalo wifi připojení,
- podpora minimálně 80 % mobilních zařízení s operačním systémem Android:
  - důležité je, aby daná aplikace byla správně funkční na co největším množství zařízení, primární je operační systém Android, podpora iOS není aktuálně podmínkou, ale spíše výhodou,
- podpora minimálně dvou jazyků (čeština, angličtina).

4.3 Požadavky na výslednou technologii

Zvolená technologie, v níž bude následně aplikace napsána, by měla umožnovat psát škálovatelný znovupoužitelný kód, vyvíjet přehledné a vzhledné uživatelské rozhraní a obsahovat nástroje pro ladění aplikace v reálném čase a také nástroje na nasazení vyvíjené aplikace na testovací zařízení či emulátor – virtuální stroj, který představuje virtuální testovací zařízení, je tak možné aplikaci otestovat v různých konfiguracích zařízení bez nutnosti tato zařízení fyzicky vlastnit. Bude také záležet na tom, jaká vývojářská prostředí, ve kterých se vyvíjí aplikace, danou technologii podporují. Vybraná technologie by měla mít podporu, obsáhlou přehlednou dokumentaci včetně ukázek kódu od tvůrců, ale zároveň také komunitu vývojářů a knihoven, které mohou usnadnit či zrychlit vývoj.

5 Porovnání jednotlivých technologií

K porovnání byly vybrány tři v zásadě odlišné technologie. Všechny mají podporu při vývoji aplikací pro Android a je možné v nich realizovat požadavky na výslednou aplikaci.

5.1 Testovací aplikace

Za účelem porovnání jednotlivých zvažovaných technologií byla v každé naprogramována jednoduchá aplikace zobrazující články. Aplikace nejprve stáhne data z API, zpracuje je a poté zobrazí jednotlivé položky, v tomto případě ve vertikálně posuvném zobrazení. Když uživatel posouvá položky, aplikace postupně načítá další data a při kliknutí na položku se zobrazí její detail. Aplikace byly testovány na mobilním zařízení – telefonu Samsung Galaxy S7 Edge s operačním systémem Android verze 8.0.

5.2 Java (Android)

Java je striktně objektově orientovaný programovací jazyk vyvinutý společností Sun, tu později převzala společnost Oracle. Google si ji při vývoji Androidu zvolil jako oficiální programovací jazyk, pouze pro syntaxi[1], nikoliv knihovnu tříd, tu používá Android vlastní. [11]. V současné době se jako alternativa Javy jeví Kotlin, který byl dokonce v roce 2019 Googlem označen jako preferovaný jazyk pro vývoj aplikací pro operační systém Android [12].

5.2.1 Android Studio

Oficiální vývojářské prostředí pro vývoj aplikací pro operační systém Android podporuje vývoj jak v Javě, tak v Kotlinu. Obsahuje celou řadu užitečných nástrojů, které vývojář ocení. Jednou z hlavních předností Android Studia oproti konkurenci je IntelliSense[2], propracovaný editor grafického rozhraní optimalizovaný pro vývoj Android aplikací nebo také šablony pro nejčastěji používané třídy a další komponenty.

[1] Syntaxe – způsob zápisu kódu
[2] Automatická nápověda ve vývojovém prostředí.

5.2.2 Android Debug Bridge

Android Debug Bridge (ADB) je vývojářský nástroj, který umožňuje instalaci aplikací, debugging[3] aplikace v reálném čase a přístup k příkazovému řádku (Shell) Unixového jádra operačního systému Android [10]. ADB lze použít společně s Android Studiem k instalaci vyvíjené aplikace na testovací mobilní zařízení či emulátor. Nasazení včetně ladění může probíhat za pomoci USB kabelu či bezdrátově v rámci lokální sítě. Ke správné funkcionalitě ADB je nutné mít aktivovaný „vývojářský režim“ na daném zařízení. Je to užitečný nástroj při vývoji i řešení problémů s Android zařízením, lze jej používat nejen s Android Studiem, ale i s dalšími vývojářskými nástroji nebo také samostatně.

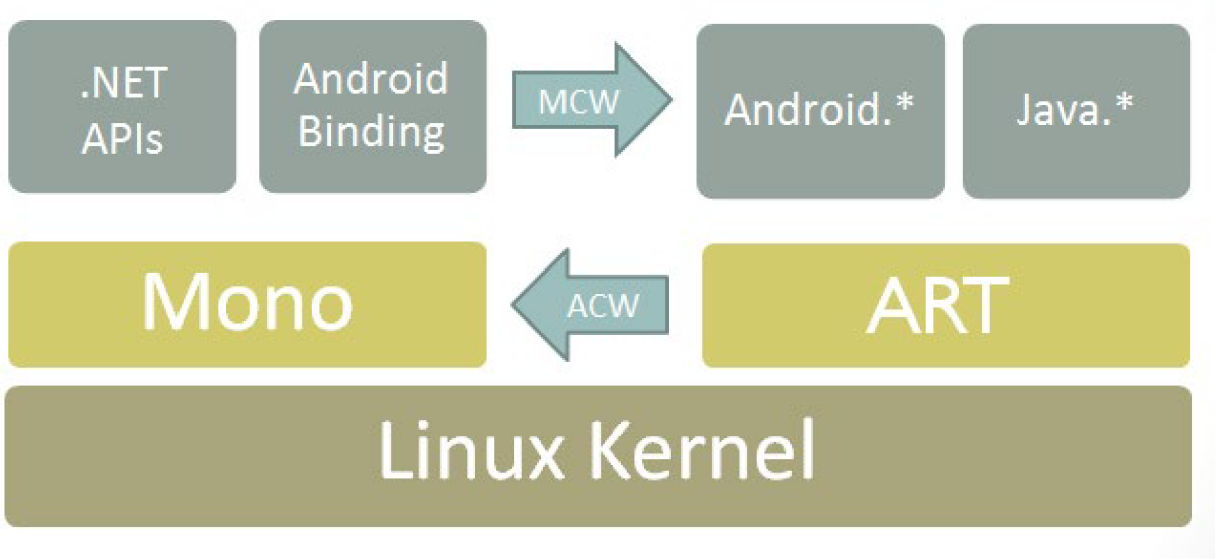
5.2.3 Vývoj testovací aplikace

Kód je, i přestože se jedná o jednoduchou testovací aplikaci, obsáhlý a vývoj, i přes použití předpřipravených šablon, zdlouhavý. Na druhou stranu běží aplikace plynule, dobře se v ní pracuje i s dokumentací. Kompilace a následná instalace aplikace přes USB kabel zabírá poměrně dlouhý čas, průměrně 50 sekund.

5.3 C# (Xamarin)

C# je objektově orientovaný programovací jazyk, vyvinutý společností Microsoft, s frameworkem .NET [13]. Xamarin je abstraktní vrstva, která spravuje komunikaci mezi sdíleným kódem v C# a nativním kódem dané platformy za pomoci Mono runtime[4] [14]. Umožňuje tak využívat téměř veškeré výhody frameworku .NET [15]. Krom aplikací přímo pro operační systém Android či iOS lze vytvářet i multiplatformní aplikace za pomocí Xamarin Forms [16].

[3] V překladu ladění, jedná se o běžně používaný vývojářský termín.
[4] Prostředí či knihovna umožňující běh programu.



Obrázek 1 – Architektura Xamarin Android. Zdroj: [17]

5.3.1 Visual Studio

Visual Studio je vyvíjeno společností Microsoft. Stejně jako v Android Studiu i ve Visual Studiu nalezneme velké množství užitečných funkcí, také je zde možné používat ADB. Přestože je Visual Studio mnohem univerzálnější a není určeno pouze pro mobilní vývoj, obsahuje také grafický editor uživatelského rozhraní, který ovšem nemá srovnatelnou funkcionalitu oproti Android Studiu.

5.3.2 Vývoj testovací aplikace

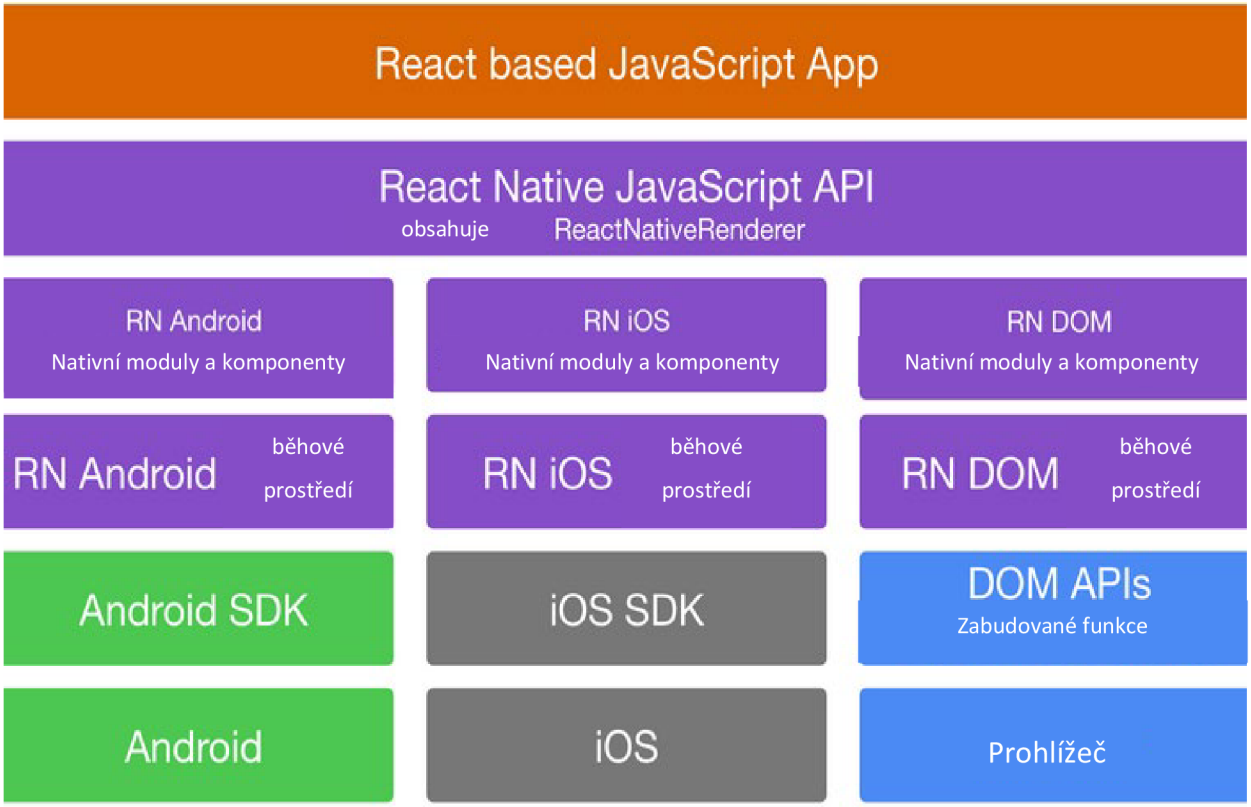
Mezi výhody se řadí možnost využívání rozsáhlého frameworku .NET, který je velmi propracovaný a pro programátora příjemný k implementaci, dále LINQ (Language Integrated Query) – to je implementace dotazovacího jazyka přímo do C#, což umožňuje dělat dotazování či vyhledávání přímo nad jakoukoliv kolekcí [18]. Narozdíl od Javy umožňuje C# díky své syntaxi psát kratší a přehlednější kód, což značně urychluje vývoj. Nevýhodou je ne zcela rozšířená komunita a podpora knihoven třetích stran. Dále je také vývojář odkázán při vývoji uživatelského rozhraní na editaci kódem, grafický editor slouží spíše jako náhled.

5.4 TypeScript (React Native)

TypeScript je programovací jazyk vytvořený společností Microsoft, jedná se o nadstavbu nad jazykem JavaScript – rozšiřuje ho o výhody objektového programování [19]. React Native je framework vyvinutý společností Facebook určený

pro multiplatformní vývoj mobilních aplikací, který umožňuje rychlý vývoj škálovatelných aplikací. Kód je možné vytvářet v JavaScriptu či TypeScriptu.

React Native využívají pro své mobilní aplikace nejenom tvůrce technologie Facebook, ale i další společnosti, jakými jsou Discord, Coinbase a další.



Obrázek 2 – Architektura React Native. Zdroj: [20]

5.4.1 Visual Studio Code

Visual Studio Code je odlehčená multiplatformní verze Visual Studia, opět z dílny společnosti Microsoft. Visual Studio Code je možné si přizpůsobovat za pomoci externích nástrojů a pluginů.

5.4.2 Framework Expo

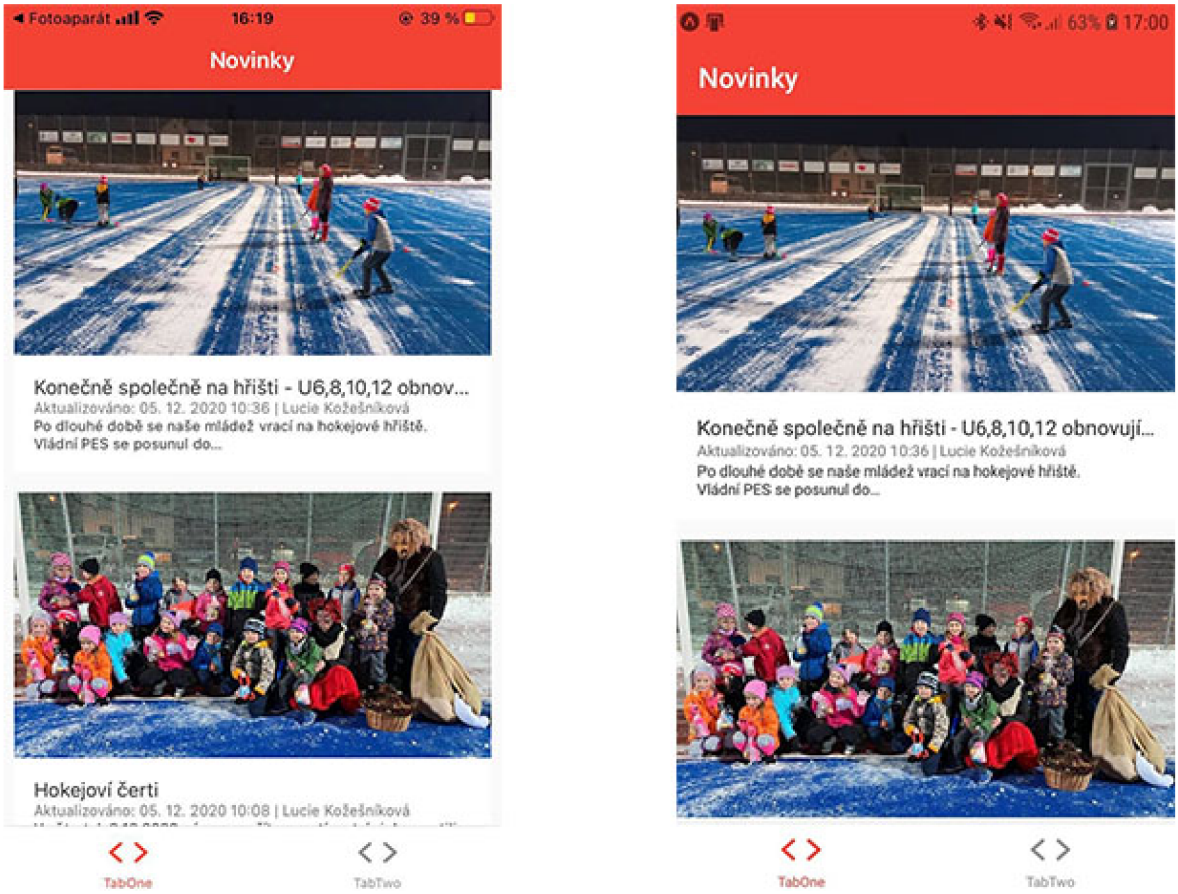
Expo je framework i sada nástrojů pro React Native usnadňující multiplatformní vývoj [21]. Díky Expo není nutné mít aktivovaný vývojářský režim na daném zařízení. Změny provedené v kódu se automaticky aktualizují i na testovacích zařízeních, změny tak programátor pozoruje téměř okamžitě. Expo funguje na bázi klient (testovací zařízení)

server (zařízení, na němž probíhá vývoj) v rámci lokální sítě. Pro testování aplikace na mobilním zařízení je nutné mít nainstalovanou mobilní aplikaci Expo Go, která je k dispozici pro Android i iOS zdarma.

5.4.3 Vývoj testovací aplikace

Vytvoření projektu začíná přes příkazový řádek – program NPM (Node.js package manager) a instalaci balíků. O sestavení programu se stará Expo, který promítá změny v kódu při každém uložení, tedy téměř v reálném čase. Ke stylování uživatelského rozhraní se používá modifikované CSS, které se běžně používá ke stylování HTML webových stránek. CSS nabízí obrovské množství možností stylování, snadno se implementuje, což umožňuje rychlý a kreativní vývoj. Uživatelské rozhraní se skládá z takzvaných komponent – komponenta obsahuje funkční kód, případně i část uživatelského rozhraní. Kolem React Native je obrovská komunitní podpora, je tedy možné pomocí NPM či dalších správců balíčků doinstalovat hodové komponenty k implementaci. Dále je zde výborná dokumentace včetně spustitelných ukázek. Obrovskou výhodou je multiplatformní vývoj pro Android, iOS i webovou aplikaci, a to s minimálními změnami v kódu. V testovací aplikaci, která byla původně navržena pro operační systém Android, vyvinuté v technologii React Native, nebylo nutné dělat žádné změny pro nasazení na iOS.

Na přiloženém obrázku je ukázka testovací aplikace, vlevo snímek obrazovky z iPhone 7 – iOS 14.4 a vpravo Samsung Galaxy S7 – Android 8.0, aplikace byla nasezena přes Expo bez jakýchkoliv úprav při zachování stejné funkcionality.



Obrázek 3 - Testovací aplikace v iOS (vlevo) a Androidu (vpravo). Zdroj: [autor]

5.5 Volba technologie a doporučení

Jako nejvhodnější varianta se po zvážení všech okolností ukazuje pro danou aplikaci využít React Native, a to díky své škálovatelnosti, rychlému vývoji a možnosti multiplatformního vývoje s minimálními změnami. Hlavním cílem aplikace je zobrazit v přehledné formě data o konaných akcích v mobilních zařízeních. Do budoucna je však počítáno s dalším rozšířením aplikace a přidání funkce propojení s klubovým účtem. I z tohoto důvodu byl zvolen React Native, nejen že bude snadné implementovat nové funkce bez narušení či úprav stávajících a již hotových komponent, ale díky jejich znuvupoužitelnosti bude aplikace rychlejší.

I když aktuálně není podpora iOS prioritou, určitě by ji bylo vhodné do budoucna, z důvodu používání i iOS zařízení návštěvníky,[5] minimálně zvážit. Díky React Native

[5] Podle analýzy dat návštěvnosti webové aplikace HockeyOnline byl za sledované období 1. 1. 2019 až 1. 1. 2021 podíl zastoupení návštěvnosti z mobilních zařízení 58,84 %, z toho s operačním systémem iOS 37,15 %, Android dosáhl 62,47 %.

a Expo by byly změny u takovéhoto typu aplikace minimální. Navíc lze aplikaci vytvořenou v React Native jednoduše upravit do podoby desktopové aplikace pro Windows a macOS.[6]

React Native jistě není zcela vhodná volba pro aplikace, které by měly obsahovat složitější funkcionalitu, chtěly přistupovat nebo obsluhovat hardware či byly použity ke tvorbě her. Zde se již mohou projevit značné rozdílnosti v jednotlivých platformách. V takových případech je namístě užití Kotlinu nebo Javy.

Mobilní aplikace psané v C# mají jistě budoucnost, avšak stále chybí dostatečná podpora komunity. Na druhou stranu Xamarin umožňuje propojení s dalšími technologiemi Microsoftu a může v mnohých případech vývoj značně usnadnit.

[6] Jedna z možných implementací pro Windows a macOS; dostupná na: https://microsoft.github.io/react-native-windows/ (citováno 4. 3. 2021).

6 Serverové služby

Většina mobilních aplikací není zcela soběstačná, ale potřebuje jakési zázemí pro jejich plnou funkcionalitu – od distribuce aplikace k uživatelům až po úložiště dat. Těmto službám říkáme obecně serverové nebo backendové. Uživatel málokdy ví o jejich existenci, ale jsou stejně důležité jako samotné aplikace.

6.1 API rozhraní

API, neboli Application programming interface, je aplikace umožňující vzájemnou komunikaci systémů nebo aplikací. Tato práce se zaměřuje na Web API, což je serverová služba, program, zajišťující bezpečný přístup aplikace k datům nebo dalším službám.

6.1.1 Druhy Web API dle způsobu komunikace

Existuje velké množství typů komunikací, v následující části budu představeny tři, které patří mezi ty nejrozšířenější. Model komunikace mají postavený na principu požadavek – odpověď (request – response).

Ke komunikaci je nejčastěji využíván HTTP protokol, který se nachází ve vrchní vrstvě ISO/OSI modelu, konkrétně v aplikační vrstvě. Základními metodami, které reprezentují elementární operace CRUD[7] v protokolu HTTP, jsou GET (pro získání dat), POST (používán k odeslání uživatelských dat na server), PUT (nahraje data na server), DELETE (na smazání dat), mezi další patří HEAD, TRACE, OPTIONS, CONNECT a PATCH; nejpoužívanějšími jsou GET a POST. Tyto metody HTTP protokolu jsou následně využívány ke komunikaci s API v rámci jednotlivých technologií. Pro bezpečnou komunikaci je používán protokol HTTPS, který je kombinací HTTP a SSL nebo TLS protokolu a zajištuje autentifikaci a důvěryhodnost přenášených dat [22].

REST

Representational state transfer (dále REST) umožňuje za pomoci HTTP dotazů snadnou práci s daty. Každý endpoint, koncový bod, představuje jednu entitu

[7] Create, read, update, delete – základní operace s daty.

a na základě HTTP metody je provedena požadovaná operace. Implementace do API je velmi snadná, existuje celá řada knihoven pro C#, Javu i další programovací jazyky. Vracená data jsou nejčastěji ve formátu JSON nebo XML. [23]

GraphQL

GraphQL, novější technologie vyvinutá společností Facebook, funguje pouze na jednom endpointu pro všechny entity. Dotazy jsou popisovány jednoduchým dotazovacím jazykem, díky tomu není nutné, na rozdíl od REST, volat více HTTP požadavků. Protože je možné zažádat si pouze o data, která v dané situaci skutečně potřebujeme, lze optimalizovat objem přenášených dat. Vrácená data jsou ve formátu JSON.

SOAP

Simple Object Access Protocol (dále SOAP) je starší technologie založená opět na HTTP protokolu, může však využívat i SMTP protokol. Nevýhodou je datová náročnost i náročnost na zpracování. [24]

6.2 Databáze

Databáze je povětšinou serverová služba umožňující uchovávání dat. Vedle serverových databází jsou i databáze lokální, například SQL Lite, užívané v mobilních i desktopových aplikacích. V kontextu této práce se bude jednat o serverové služby.

Aplikace mají přímý přístup k databázi zřídkakdy, protože by se jednalo o bezpečnostní riziko. Při dekomprimaci programu by se totiž případný útočník mohl dostat k přístupovým údajům k databázi a odcizit tak uložená data, poškodit je nebo pozměnit. Právě proto má k databázi přístup pouze API, které zajišťuje poskytování dat aplikacím, u API běžící na serveru je menší šance na kompromitování přístupových údajů k databázi.

Relační databáze

Model relační databáze, který dnes používá většina moderních databázových systémů, byl vyvinut již v sedmdesátých letech minulého století, je intuitivní, pro člověka snadno pochopitelný a data jsou systematicky seskupována do již dále nedělitelných celků. Relační databáze ukládají data do tabulek, v nichž jsou jednotlivé sloupce vázány na sloupce v jiných tabulkách; takto propojená datová poje jsou na sobě závislá. Tato

propojení neboli vztahy jsou realizována takzvanými klíčovými hodnotami, které jsou uloženy v příslušných sloupcích.

Jazykem pro práci s relačními databázemi je SQL – Structured Query Language, jedná se o standardizovaný strukturovaný dotazovací jazyk používaný k dotazům, aktualizaci a správě relačních databází [25].

Jedním z nejrozšířenějších databázových systému je MySQL – rychlý a univerzální systém, který byl vyvinut společností TcX. Důvodem jeho obliby je především to, že je na rozdíl od konkurence poskytován ve většině případů zdarma [25]. Pro větší komerční projekty bývají používány systémy Microsoft SQL Server, Oracle database a další.

NoSQL

U takzvaných NoSQL databází se jedná o koncept, v němž nejsou data uložena v tabulkách, jak je tomu u tradičních relačních databází; obvykle je zde pro ukládání dat používána stromová či grafová struktura. Jsou to například síťové databáze, objektové databáze nebo hierarchické databáze. NoSQL může být kvůli svému názvu interpretován jako Non only SQL namísto non SQL, nicméně některá řešení dokonce SQL jazyk pro dotazování podporují [26]. Mezi NoSQL databázové systémy řadíme například Redis a MongoDB.

7 Vývoj a testování

Vývoj aplikace lze rozdělit na dvě části, které spolu velice úzce souvisí, jedná se o vývoj API, které zajišťuje přístup k datům, a samotnou mobilní aplikaci. Vývoj obou těchto částí probíhal paralelně.

7.1 API

Jak bylo již zmíněno, API pro mobilní aplikaci HockeyOnline bylo vyvíjeno v technologii ASP.NET Core pomocí programovacího jazyka C# a způsob komunikace REST s daty probíhal v textovém formátu JSON. API byla vyvíjena tak, aby ji bylo možné i do budoucna rozšiřovat, případně je zde počítáno i s využitím pro jiné další aplikace. Výhodou vývoje s ASP.NET Core frameworkem je velké množství předpřipravené funkcionality. Další velká výhoda ASP.NET Core se projeví při nasazení, jelikož je možné sestavit program pro téměř jakékoliv běhové prostředí.

Prvním krokem vývoje bylo vytvořit a spustit alespoň první základní část API, aby jej následně bylo možné používat už při výběru technologie pro mobilní aplikaci v rámci testovacích aplikací. Dále již bylo API vyvíjeno vždy po částech, souběžně s mobilní aplikací. Autorizaci uživatele API obsahovat nebude, nyní není potřeba, do budoucna při rozšiřování aplikace o klubovou sekci to bude však nezbytné.

Controller `ArticleController` předzpracovává a zajišťuje data o článcích, konkrétně kolekci článků pro úvodní obrazovku, a také detailní obsah jednotlivých článků, který se zobrazí po kliknutí na článek na úvodní obrazovce.

Controller `TournamentController` slouží pro předzpracování a poskytování dat o turnajích, utkáních i o týmech, které se turnajů účastní.

7.1.1 Instalace balíčků

V průběhu vývoje byl pro instalaci balíčků do projektu použit NuGet. Jedná se o správce balíčků určený pro .NET projekty, integrovaný přímo do Visual Studia. Pro ovládání správce balíčků NuGet lze využít uživatelské rozhraní či příkazy v konzoli.

7.1.2 Struktura API

API je rozděleno do dvou controllerů – `ArticleController` a `TournamentController`. Každý je v kódu representován jako třída, jejíž metody označené atributem `[HttpGet]`, `[HttpPost]` a další představují jednotlivé endpointy. Controllery na této API umí obsluhovat pouze GET požadavky pro získání dat; vkládání ani úprava dat není pro funkcionalitu aplikace potřeba. Každá metoda vyžaduje parametr `apiKey`, dvacet čtyři znaků dlouhý textový řetězec, který slouží jako identifikátor aplikace, jež o data žádá. Pro vývojové účely mobilní aplikace HockeyOnline je to:
`sdkjcvsf45dfbv86s6d5v469d.`

7.1.3 Databáze

API bude poskytovat data z již existující databáze. Webová aplikace HockeyOnline využívá databáze dvě – jednu s názvem HockeyOnlineDatabase pro záznamy o turnajích a článcích, druhou ClubDatabase sloužící k uchovávání informací o členech klubu, nominacích, docházce a komunikaci. ClubDatabase do API nebude zatím implementována, pro aktuální požadovanou funkcionalitu postačí první databáze. Pro ukládání API klíčů a statistik o využívání API byla vytvořena samostatná databáze. Pro práci a návrh s databází a manipulaci s daty bylo využíváno nástroje Microsoft SQL Server Management Studio 2018.

7.1.4 Entity Framework

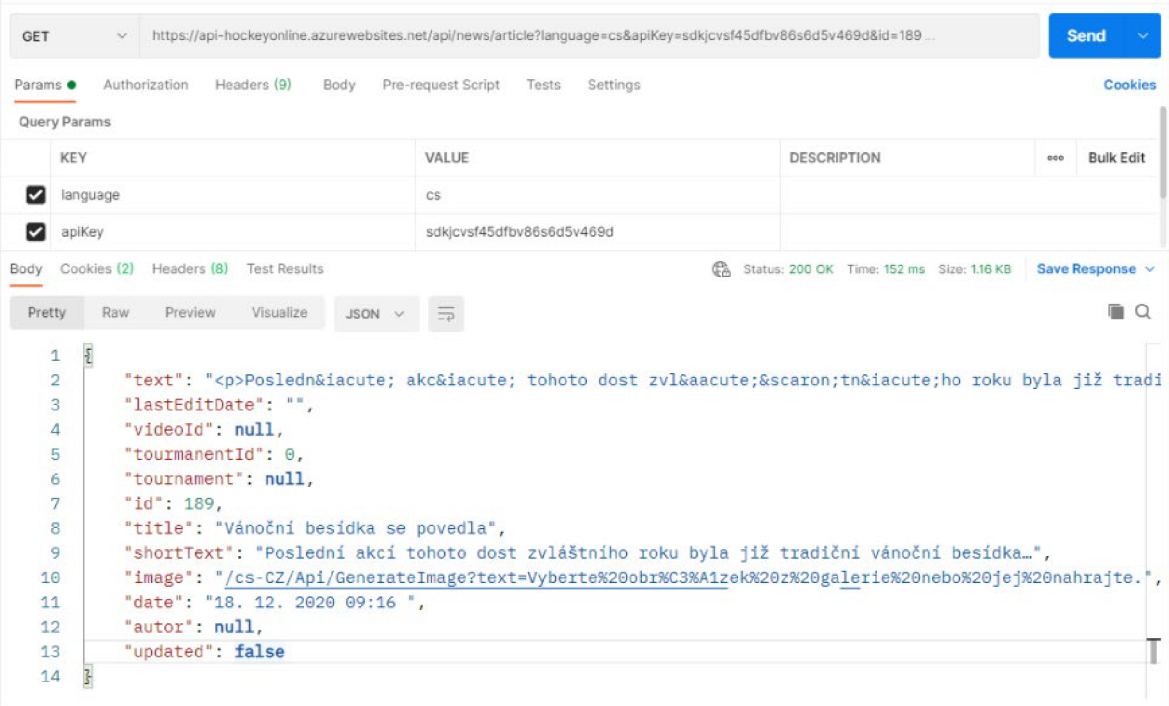
Nedílnou součástí a jednou z dalších předností ASP.NET Core je Entity Framework (dále jen EF) umožňující objektově relační mapování. Jsou dva způsoby užívání EF: prvním je Code First, utvořený objektový návrh, z něhož je za pomoci nástrojů EF vygenerována databáze, druhým způsobem je DB (Database) First, který spočívá ve vygenerování objektového modelu z navržené databáze. Oba způsoby lze i kombinovat. Pro vývoj API byla využita varianta DB First, protože API pracuje s již existující databází.

7.1.5 Testování

V průběhu vývoje byla API průběžně testována nástrojem Postman (nástroj určený k testování endpointů API) i přímo z mobilní aplikace. V rámci vývoje byl Postman použit k otestování správnosti dat přijatých před implementací z endpointů do samotné mobilní aplikace.

Na obrázku je ukázka testu endpointu Article, který zajišťuje poskytování detailních informací o článku na základě ID, což je jedinečný číselný identifikátor článku. Dalšími parametry jsou `language`, tedy v jakém jazyce má daný článek být, a `apiKey`, který je zde využíván jako identifikátor aplikace pro statistiky o užívání API.

URL adresa endpointu:
`/api/news/article?language=cs&apiKey=sdkjcvsf45dfbv86s6d5v469d&id=12`



Obrázek 4 – Aplikace Postman testování API. Zdroj: [autor]

Na obrázku je ukázka z testování API nástrojem Postman – nejprve je nutné zadat URL adresu a typ požadavku, poté parametry, případně do hlavičky i další atributy. Po odeslání požadavku a následném přijetí odpovědi od API naformátuje Postman odpověď tak, aby bylo možné se v datech dobře orientovat.

7.1.5.1 Microsoft Azure

Microsoft Azure není testovací nástroj, nýbrž cloudová platforma poskytující obrovské množství placených služeb. Jednou z těchto služeb je i hostování ASP.NET Core aplikací, které je navíc pro testovací účely zdarma. Nasazení je velmi rychlé, hotové na dvě kliknutí přímo z Visual Studia. Navíc tato služba podporuje vzdálené ladění, odchytávání a logování chyb, což v konečném důsledku velmi zrychlilo a usnadnilo vývoj. Nicméně pro finální nasazení není Azure z pohledu klubu výhodný, hlavně kvůli

jeho vysoké ceně. Hotové verze API běží spolu s dalšími službami HockeyOnline na virtuálním serveru, který má klub dlouhodobě pronajatý.

7.1.6 Formát dat

Při testování API s mobilní aplikací byla odhalena chyba u vykreslování HTML obsahu v mobilní aplikaci. Text článků je ukládán jako HTML. V některých částech HTML záznamů byl nastaven CSS styl Font nekompatibilní s fonty daného zařízení, proto bylo tedy nutné tuto část CSS stylu odebrat. Nevhodnější je tento problém vyřešit již na úrovni API.

Za tímto účelem byla napsána metoda `RemoveHTMLStyle` (viz ukázka níže), která jako parametr vyžaduje textový řetězec představující styl k odstranění. Metoda je univerzální a znovupoužitelná.

```
public static string RemoveHtmlStyle(this string html, string style)
        {
            Regex regex = new Regex(style + "\\s*:.*?;?");

            return regex.Replace(html, string.Empty);
        }
```

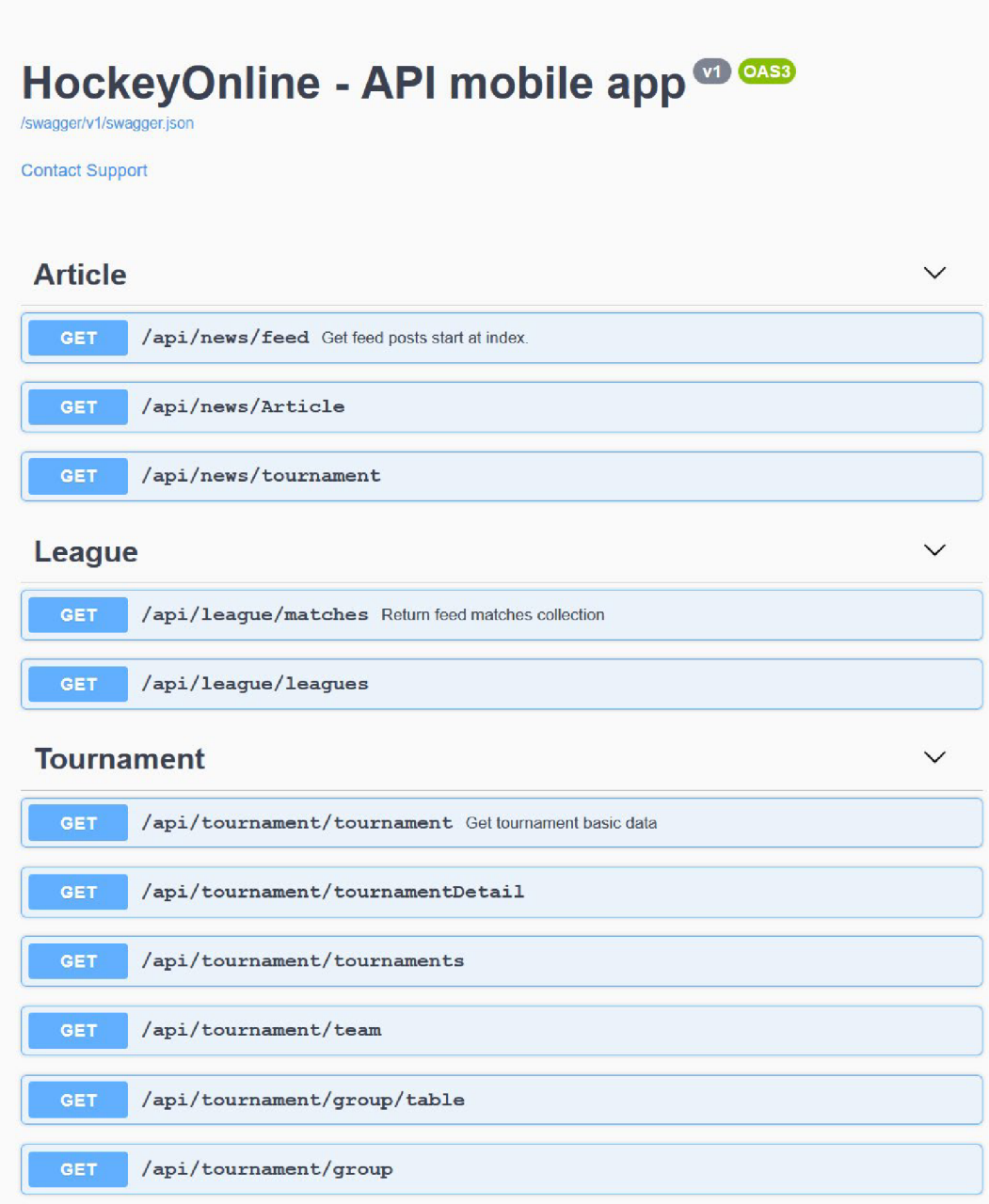
Ukázka kódu 1 – Odstranění stylů z HTML kódu. Zdroj: [autor]

7.1.6.1 Sledování chyb

Stejně jako webová aplikace HockeyOnline i API implementuje Sentry.io, řešení pro reportování chyb, které umožňuje sledovat chování aplikace v reálném čase a zachytávat výjimky, jež mohou za běhu aplikace nastat. Je zde možné efektivněji reagovat na chyby neodhalené při běžném testování, což se projeví až v produkční verzi; chyby je možné vidět v celém kontextu včetně dalších statistik. Implementace opět probíhá přes instalaci NuGet balíčku `Sentry.AspNetCore` a následná konfigurace v souboru `Startup.cs`. Setry.io podporuje celou řadu technologií, do nichž je možné toto reportování implementovat. Výhodou je tedy kromě podpory ASP.NET Core také podpora React Native, a tak bylo Sentry.io pro lepší doladění mobilní aplikace implementováno i tam. Data jsou odesílána na API Sentry.io, kde jsou zpracována. Je možné využívat servery Sentry.io nebo si celé řešení hostovat na vlastních strojích. Pro účely HockeyOnline je zcela dostačující využívat služeb poskytovaných Sentry.io.

7.1.7 Dokumentace

Důležitou součástí API umožňující efektivní implementaci, další rozvoj i použití API více vývojáři, je dokumentace, která popisuje jednotlivé endpointy i jejich parametry. Tradičním přístupem je popsat API ručně, existuje ale i software pro automatické generování dokumentace API. Jedním z nich je open source Swagger, jenž byl implementován pomocí NuGet balíčku `Swashbuckle.AspNetCore`. Dále bylo nutné upravit soubor `Startup.cs`, jedná se o pouhé přidání služeb do metod `ConfigureServices` a `Configure`. Dokumentace se pak generuje automaticky z metod v controllerech, které jsou označeny atributem `[HttpGet]`, případně dalšími typy HTTP metod. Komentáře k těmto metodám jsou použity jako popisky jednotlivých endpointů. Celá dokumentace je dostupná na: `/swagger/index.html`.



Obrázek 5 – Swagger, automaticky generovaná dokumentace API. Zdroj: [autor]

7.2 Mobilní aplikace

K vývoji samotné mobilní aplikace HockeyOnline byla použita, na základě předchozího porovnaní technologií, technologie React Native s pomocí Typescript. Vývoj probíhal ve vývojářském prostředí Visual Studio Code s pluginem React Native Tools rozšiřujícím IntelliSence a další funkce pro pohodlný vývoj React Native[8].

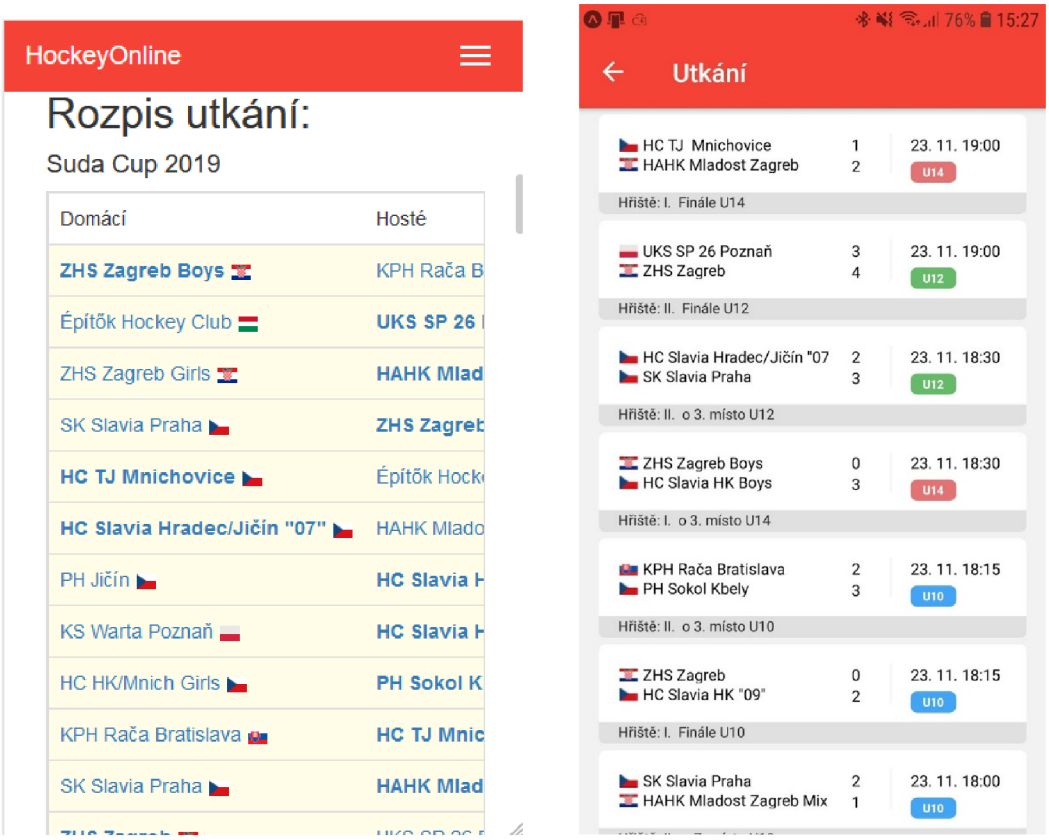
7.2.1 Uživatelské rozhraní

Uživatelské rozhraní je jednou z velmi podstatných částí mobilní aplikace, protože s ním interaguje uživatel. Pro mobilní aplikaci HockeyOnline je uživatelské rozhraní zásadní, jelikož zajišťuje přehledné zobrazení informací, což je primárním účelem aplikace.

Hlavním problémem ve webové verzi aplikace je nepřehledné zobrazování seznamů. Jeden z důležitých seznamů je tabulka utkání, která je sice optimalizována pro mobilní zařízení a lze ji horizontálně posouvat, ale přehlednost není zcela ideální. V mobilní aplikaci byl tedy zvolen zcela odlišný přístup k zobrazování jednotlivých utkání. Jména týmů jsou zobrazena pod sebou, stejně tak scóre utkání, název hřiště a skupina je zobrazena v šedé liště, jak je vidět v obrázku níže. Po kliknutí na jméno týmu se otevře obrazovka s detailními informacemi o týmu. Jednotlivá utkání se načítají vždy po šesti, automaticky po dosáhnutí konce seznamu. Seznam je seřazen podle času utkání a také podle toho, zda již bylo utkání odehráno. Funkcionalita tedy zůstala a podařilo se zvýšit přehlednost.

V obrázku 6 vlevo je snímek obrazovky webové verze, vpravo snímek obrazovky z mobilní aplikace. Problém webové verze na mobilních zařízeních je, že na první pohled neposkytuje potřebné informace uživateli; v mobilní aplikaci přepracovaný design nabízí přehledně veškeré podstatné informace o utkání (názvy obou týmů, název skupiny i kategorii a název hřiště) hned na první pohled bez nutnosti horizontálně posouvat obsah obrazovky.

[8] Rozšíření dostupné na oficiálním webu:
https://marketplace.visualstudio.com/items?itemName=msjsdiag.vscode-react-native



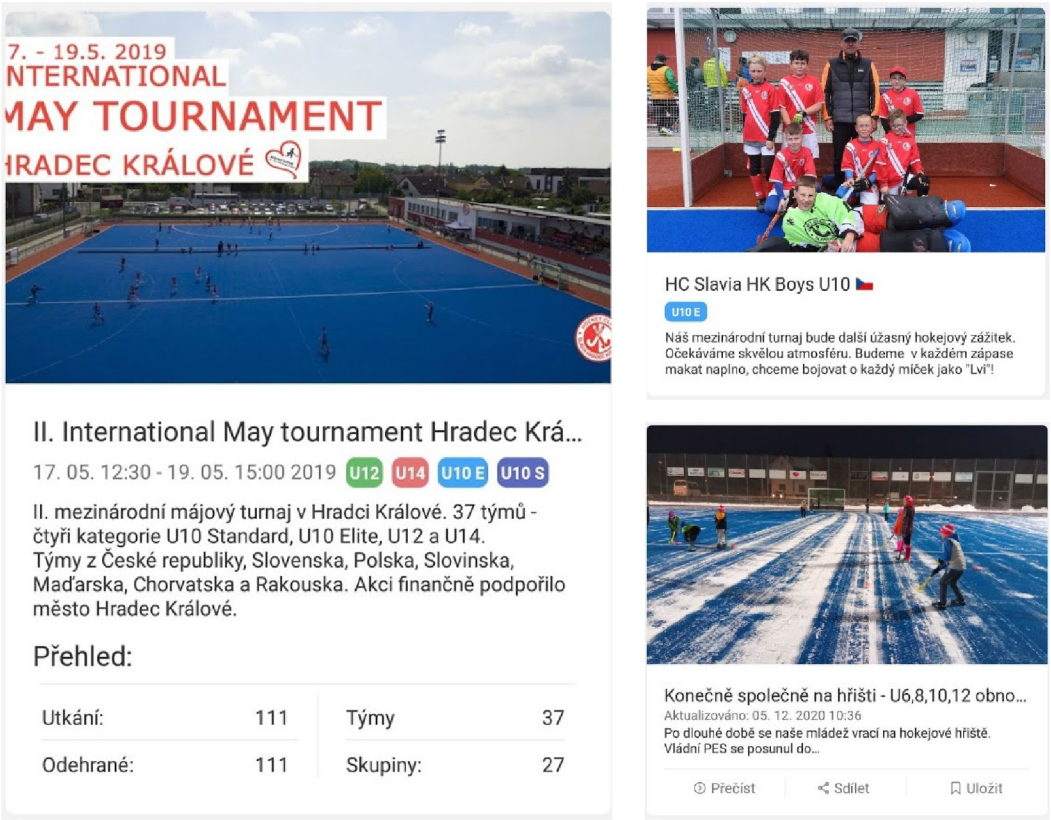
Obrázek 6 – Porovnání webové (vlevo) a mobilní (vpravo) aplikace HockeyOnline. Zdroj: [autor]

V aplikaci jsou často používané horizontálně posuvné seznamy, které se na dotykových obrazovkách příjemně ovládají. Umožnují tak umístit přehledně velké množství informací na jednu obrazovku. Horizontálně posuvné seznamy byly použity například u přehledu turnajů či v detailu týmu nebo turnaje pro zobrazení jednotlivých skupin. Prvky uživatelského rozhraní jsou tvořeny jako komponenty, je proto možné je používat opakovaně. Více ukázek uživatelského rozhraní aplikace se nachází v kapitole 7.3.1.

7.2.2 Komponenty

Celá aplikace je dělena do komponent, většina je znovupoužitelná, což je u používání komponent hlavní výhoda. Každá komponenta obsahuje funkční část, některé obsahují i uživatelské rozhraní. Příkladem komponenty bez uživatelského rozhraní je `IMLocalized`, které slouží k vícejazyčnému načítání textů pro uživatelské rozhraní.

Nejuniverzálnější komponentou s uživatelským rozhraním, vytvořenou při vývoji, je `MainCard`, která je díky své jednoduchosti používána téměř na všech obrazovkách. Skládá se z obrázku, nadpisu, druhého nadpisu a textu; lze ji rozšířit o další komponenty. Po kliknutí na komponentu se provede navigace. Vstupními parametry komponenty jsou typ `MainCardContent`, dále pak volitelně `secondOther` a `body`, kterými lze předat další komponenty pro rozšíření té základní. V obrázku níže můžete vidět ukázku některých využití.



Obrázek 7 – Různé použitá použití kompoinrenty MainCard. Zdroj: [autor]

7.2.3 Komunikace s API

Pro komunikaci s API je volána funkce `fetch`, jejímž parametrem je URL adresa. Funkce je volána asynchronně, aby nedošlo k zamrznutí uživatelského rozhraní. Konstanty jako endpointy nebo API klíč jsou načítány pomocí komponenty s názvem API.

```
import * as api from '../constants/Api';
import { IMLocalized} from '../components/IMLocalized';

…
      setLoading(true);
        fetch(api.endPoint + '/tournament/tournament?id=' + id + '&te
xtLength=200&language=' + IMLocalized('language') + '&apiKey='+api.ap
iKey)
            .then((response) => response.json())
            .then((json) => setData(json))
            .catch((error) => console.error(error))
            .finally(() => {
                setLoading(false);
            });
```

Ukázka kódu 2 – Komunikace stažení dat z API. Zdroj: [autor]

Kód je ukázkou stažení základních informací o turnaji, konkrétně z komponenty `QRCodeReader`, která slouží k načítání QR kódu. Po načtení a zpracování QR kódu, z něhož je získáno identifikační číslo turnaje, je zavolán `fetch`, ten se pokusí z API stáhnout data. V URL adrese si můžete všimnout parametru `textLength=200` – ten říká, jaký je maximální počet znaků popisu turnaje, protože se jedná pouze o náhled, popis je tedy redukován již na úrovni API služby. Tím se ušetří množství přenášených dat a zvýší rychlost načínání. Tato úspora se projeví například při načínání většího množství náhledů turnajů či článků v podobě seznamů, parametr `textLength` je obsažen ve všech endpointech API, v nichž je přenášen text či popis. Dalším zajímavým parametrem je language, kterým je popsáno, v jakém jazyce chceme data i API získat.

7.2.4 Vícejazyčnost

Mobilní aplikace HockeyOnline je určena pro mezinárodní akce, je tedy nezbytné, aby byl obsah aplikace vícejazyčný. Prioritou je podpora češtiny a angličtiny, další jsou zvažovány, ale prozatím nevyžadovány. Aplikace obsahuje dva textové soubory s texty uživatelského rozhraní, tyto texty jsou uloženy data ve formátu JSON. Jeden soubor obsahuje české překlady, druhý anglické. Podle toho, jaký jazyk má nastavený operační systém, jsou poté načítány překlady. Ke zjištění, jaký je výchozí jazyk systému, jsou používány komponenty balíčku `expo-localization`. Pokud je rozpoznána čeština, jsou využívány české překlady, v ostatních případech je používána angličtina.

Obsah, jako jsou články, popisky turnajů či týmů, je uložen v databázi také ve dvou jazycích – v originálním českém i v anglickém překladu. Při kontaktování API mobilní

aplikací je vždy zmíněn parametr language, který popisuje, v jakém jazyce chceme data získat. Výhodou tohoto řešení je, že veškerý překlad je tvořený člověkem, je tak povětšinou kvalitnější, než by byl strojový automatizovaný překlad.

7.2.5 Knihovny

Pro instalaci balíčků, knihoven, je použit NPM, avšak namísto klasického příkazu pro instalaci balíku je doporučeno používat `expo install <název>`, díky čemuž je zajištěna kompatibilita všech balíčků a SDK.

Vybrané balíčky:

- react-native
  - obsahuje základní prvky uživatelského rozhraní,
- react-navigation/native
  - umožňuje navigaci mezi obrazovkami,
- expo/vector-icons
  - sada ikon používaných v aplikaci,
- expo-barcode-scanner
  - poskytuje přístup k fotoaparátu a vyhodnocuje QR kód.

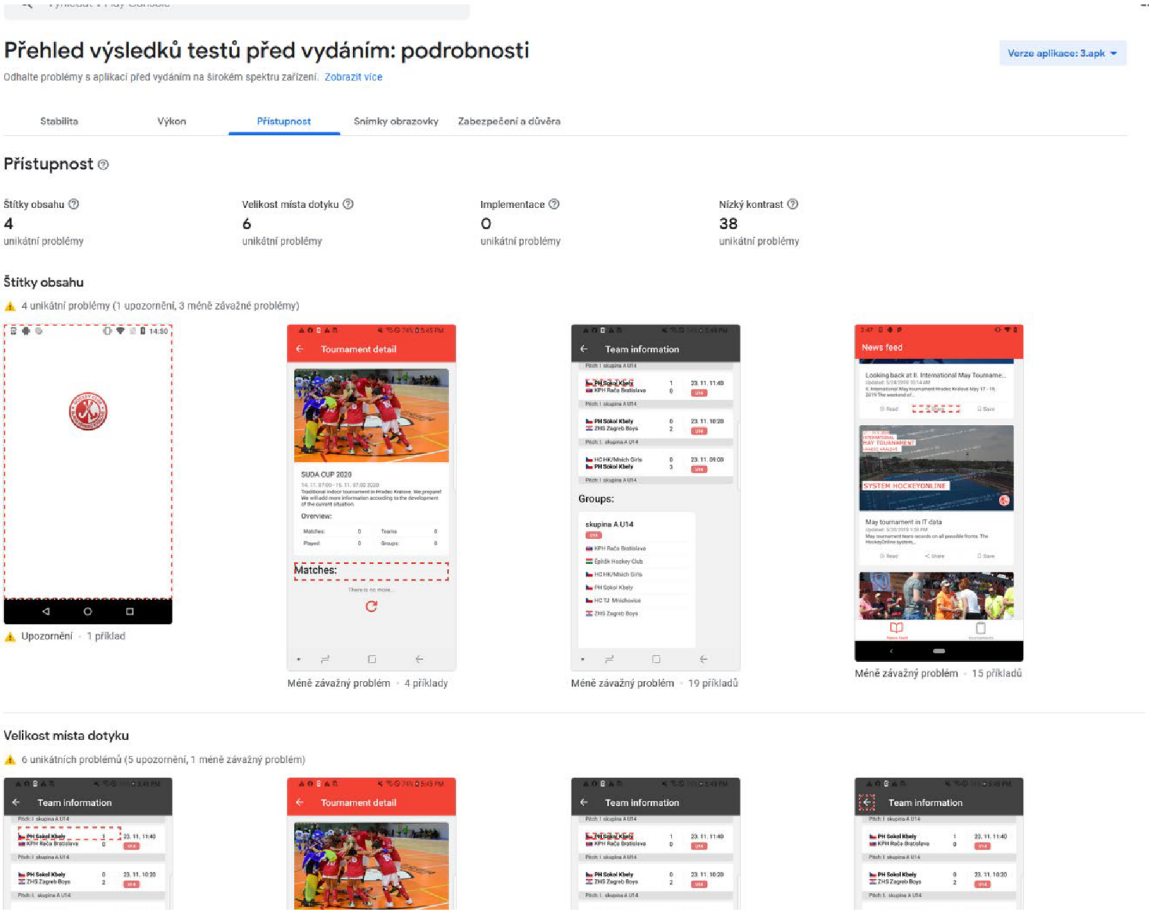
7.2.6 Publikování na Google Play

Pro publikování aplikace na Google Play bylo nejprve nutné zkontrolovat veškerou funkcionalitu a zjistit vyžadovaná práva k přístupu k operačnímu systému a hardwaru. V případě aplikace HockeyOnline jsou to oprávnění INTERNET pro komunikaci na internetu, CAMERA pro přístup k fotoaparátu při skenování QR kódů, READ_CONTACTS využívané při sdílení a READ_PHONE_STATE k určení, v jakém stavu se mobilní zařízení nachází. Poté se tyto požadavky zadají do souboru s popisem aplikace `app.json`. Při sestavení aplikace jsou data ze souboru `app.json`. předána do metadat aplikace, u operačního systému Android konkrétně do souboru `AndroidManifest.xml`. Dalším krokem bylo vyexportovat aplikaci do APK. Součástí Expo jsou nástroje pro exportování, které nahrají soubory projektu na cloudovou službu, kde proběhne sestavení a podepsání digitálním certifikátem APK[9]. Tato

[9] Odkaz na sestavení https://expo.io/@podzimekdavid/projects/HockeyOnline

procedura zabere v průměru třicet minut. Dále následovalo nahrání a vytvoření aplikace v Google Play Console, zde bylo nutné nejprve vyplnit celou řadu dotazníků ohledně chování aplikace a soukromí uživatelů, nahrát popis a snímky obrazovky aplikace. Posledním krokem bylo vytvořit vydání, nahrát APK soubor a vyčkat na schválení aplikace, to trvalo pět dní. Aplikace je vydána jako tester v modu předběžného vydání, kde se uživatelé na Google Play musí nejprve registrovat, než si aplikaci budou moct naistalovat.

Před zveřejněním Google otestuje aplikaci automatizovanými testy na několika zařízeních. Toto testování má za cíl odhalit potencionálně nebezpečné chování aplikace a také poskytnout zpětnou vazbu vývojářům. Výstupem těchto testů jsou reporty, dále má však vývojář k dispozici snímky obrazovky i videa, viz obrázek 8 níže. Během prvního vydání aplikace nebyly zjištěny žádné závažné problémy, pouze upozornění a méně závažné problémy týkající se kontrastu obsahu a velikosti klikatených částí, které byly v další verzích upraveny. Testy byly podle reportu provedeny na třech zařízeních – Nokia 1, Google Pixel 3, Samsung Galaxy S9.



Obrázek 8 – Výsledek testování aplikace na Google Play. Zdroj: [autor]

7.3 Rozšíření funkcionalit

Po zhotovení první verze mobilní aplikace byla proti původním požadavkům na aplikaci přidána celá řada rozšiřujících funkcionalit, nebo byla upravena stávající funkcionalita. Rozšiřování aplikace bylo již poměrně rychlé díky znuvupoužitelnosti již dříve vytvořených komponent.

7.3.1 Ligová utkání

Jedná se o funkci, kterou ani původní webová aplikace HockeyOnline nedisponuje, a to je zobrazování utkání z dlouhodobých ligových soutěží, kterých se klub v rámci České republiky účastní. Nové funkce bylo nutné přidat do API i do mobilní aplikace. Rozšíření mobilní aplikace bylo velmi rychlé, byly použity komponenty, které zobrazují utkání v turnajích.

Rozšiřování API bylo složitější, bylo třeba vytvořit nový controller LeagueController, který obstarává poskytování dat o utkání. Tato data jsou uložena v databázi, kam jsou stahována z API svazu Českého pozemního hokeje (dále ČSPH). API HockeyOnline v pravidelných intervalech automaticky stahuje změny z API ČSPH. Nelze použít přímo API ČSPH, protože přenáší v rámci jednoho požadavku zbytečně velký objem dat a odpověď API ČSPH trvá v průměru 6543 ms, což je doopravdy hodně dlouhá doba pro uživatele, který by čekal šest sekund na zobrazení jednoho utkání. Data jsou tedy předpřipravena v databázi HockeyOnline, odkud je přes API stahuje a následně zobrazuje mobilní aplikace. Do budoucna je plánováno nasadit zobrazování ligových utkání i ve webové verzi přes API HockeyOnline.

7.3.2 Světlý a tmavý režim

Přepínání mezi světlým a tmavým režimem je dnes ve webových a mobilních aplikacích velmi populární. Tmavý režim šetří spotřebu baterie mobilního zařízení i zrak uživatele. Přepínat režimy je možné v nastavení aplikace. Zda má uživatel zvolený tmavý nebo světlý režim, je uloženo v lokálním uložišti zařízení, pro práci s lokálním uložištěm slouží balíček `@react-native-async-storage/async-storage`. Nastavení z lokálního uložiště jsou načtena v komponentě `ColorThemeContext`, pro předávání tohoto nastavení mezi komponentami a promítnutí okamžitých změn při přepnutí barevného režimu je používáno takzvané `ContextApi.` To umožňuje sdílet si globální

instanci přes komponentu `ContextProvider`. V podřazených komponentách je tak možné zažádat o instanci `Contextu`, v tomto případě `ColorThemeContext`.

```
import { ColorThemeContextState } from './types'
import React, { useEffect, useState, createContext, useContext } from
 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';

const contextDefaultValues: ColorThemeContextState = {
  themeName: '',
  toogle: () => { }
};
export const ColorThemeContext = createContext<ColorThemeContextState
>(contextDefaultValues);

const ColorThemeContextProvider = ({ children }: { children: any }) =
> {
  const [themeName, setTheme] = useState<string>('');
  const colorScheme = useColorScheme();

  const toogle = () => {
    setTheme(themeName == 'light' ? 'dark' : 'light');
  };

  useEffect(() => {
    if (themeName != '')
      AsyncStorage.setItem('HO::theme', `${themeName}`)

  }, [themeName]);

  useEffect(() => {
    AsyncStorage.getItem('HO::theme').then((value) => {
      if (value) {
        setTheme(value);
      } else {
        setTheme(colorScheme);
      }
    });
  }, []);

  return (
    <ColorThemeContext.Provider value={{ themeName, toogle }}  >
      {children}
    </ColorThemeContext.Provider>
  );
};
export default ColorThemeContextProvider;
```

Ukázka kódu 3 – Komponenta ColorThemeContext, ColorThemeContextProvider. Zdroj: [autor]

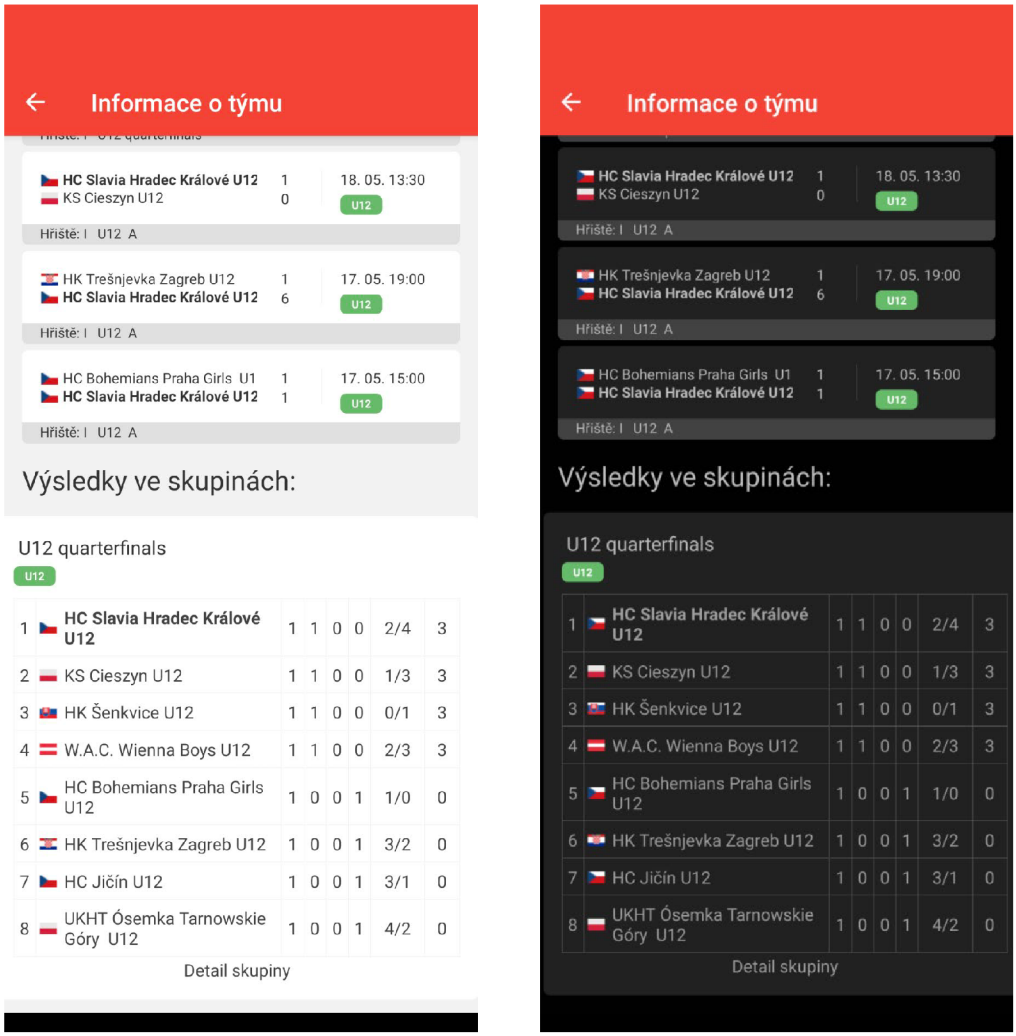
Použití `ColorThemeContext` je ukázáno v ukázce kódu 4. Přes funkci `useContext`, ta je součástí základního balíčku `react`, je zažádáno o `ColorThemeContext`, poté je na základě nastaveného barevného režimu zvoleno, jaká paleta barev definovaných v komponentě `Color` bude používána.

```
…
import Colors from '../constants/Colors';
import React, { useEffect, useState, useContext } from 'react';
import { ColorThemeContext } from '../ColorContext'

export default function Settings() {
    const { themeName, toogle } = useContext(ColorThemeContext);
    const theme = themeName === 'dark' ? Colors.dark : Colors.light;

…
```

Ukázka kódu 4 - Užití ColorThemeContext. Zdroj: [autor]



Obrázek 9 - Porovnání světlý (vlevo) a tmavý (vpravo) režim. Zdroj: [autor]

7.3.3 Ověřování API klíče

Zde se nejednalo přímo o novou funkcionalitu, nýbrž přepracování stávající. Doposud byl API klíč posílán jako parametr `apiKey` HTTP požadavku v url adrese, po úpravách je zasílán v hlavičce požadavku pod názvem `x-hockey-online`, podoba API klíče se nezměnila. Tato změna umožňuje jednodušší zpracování na API. V ukázce kódu 5 je upravený kód, kde je již, na rozdíl od ukázky kódu 2, API klíč přidávaný do hlavičky.

```
…
      fetch(api.endPoint + '/tournament/tournament?id=' + id + '&text
Length=200&language=' + IMLocalized('language'), {
        method: 'GET', headers: {
          'Accept': 'application/json',
          'Content-Type': 'application/json',
          'x-hockey-online': api.apiKey
        }})
            .then((response) => response.json())
            .then((json) => setData(json))
            .catch((error) => console.error(error))
            .finally(() => {
                setLoading(false);
            });
…
```

Ukázka kódu 5 – Upravená komunikace s API – API klíč v hlavičce. Zdroj: [autor]

Ověření API klíče na straně API je pak již poměrně snadné. Pro tyto účely byl naprogramován atribut `[ApiKeyAttribute]`, ten se umístí nad definici controlleru. Atribut tak funguje jako filtr a všechny požadavky na takto označený controller musí obsahovat platný API klíč, v opačném případě je zaslána odpověď, že požadavek nebyl v pořádku. V atributu je načten API klíč z hlavičky požadavku a následně vyhledán v databázi. Pokud je API klíč v pořádku, pokračuje se dále již na konkrétní metodu controlleru.

```
…
[AttributeUsage(validOn: AttributeTargets.Class)]
    public class ApiKeyAttribute : Attribute, IAsyncActionFilter
    {
        private const string APIKEYNAME = "x-hockey-online";
        public async Task OnActionExecutionAsync
               (ActionExecutingContext context, ActionExecutionDelegate next)
        {
            if (!context.HttpContext.Request.Headers.TryGetValue
                (APIKEYNAME, out var extractedApiKey))
            {
                context.Result = new JsonResult(new ErrorResponse()
                {
                    Error = "Api key not provided"
                })
                {
                    StatusCode = 401,
                };
                return;
            }
            var database =

context.HttpContext.RequestServices.GetRequiredService<ApplicationDbContext>
();
            var  apikey  =  database.Apikeys.FirstOrDefault(x  =>  x.Key  ==
extractedApiKey.ToString());

            if (apikey == null)
            {
                context.Result = new JsonResult(new ErrorResponse()
                {
                    Error = "Api Key is not valid"
                })
                {
                    StatusCode = 401,
                };
                return;
            }

            if (!apikey.Enabled)
            {
                context.Result = new JsonResult(new ErrorResponse()
                {
                    Error = "Api Key is not valid",
                    Message = "The key has expired or the key is not enabled"

                })
                {
                    StatusCode = 401,
                };
                return;
            }

            await next();
        }
    }
```

Ukázka kódu 6 – Atribut ApiKeyAtribute. Zdroj: [autor]

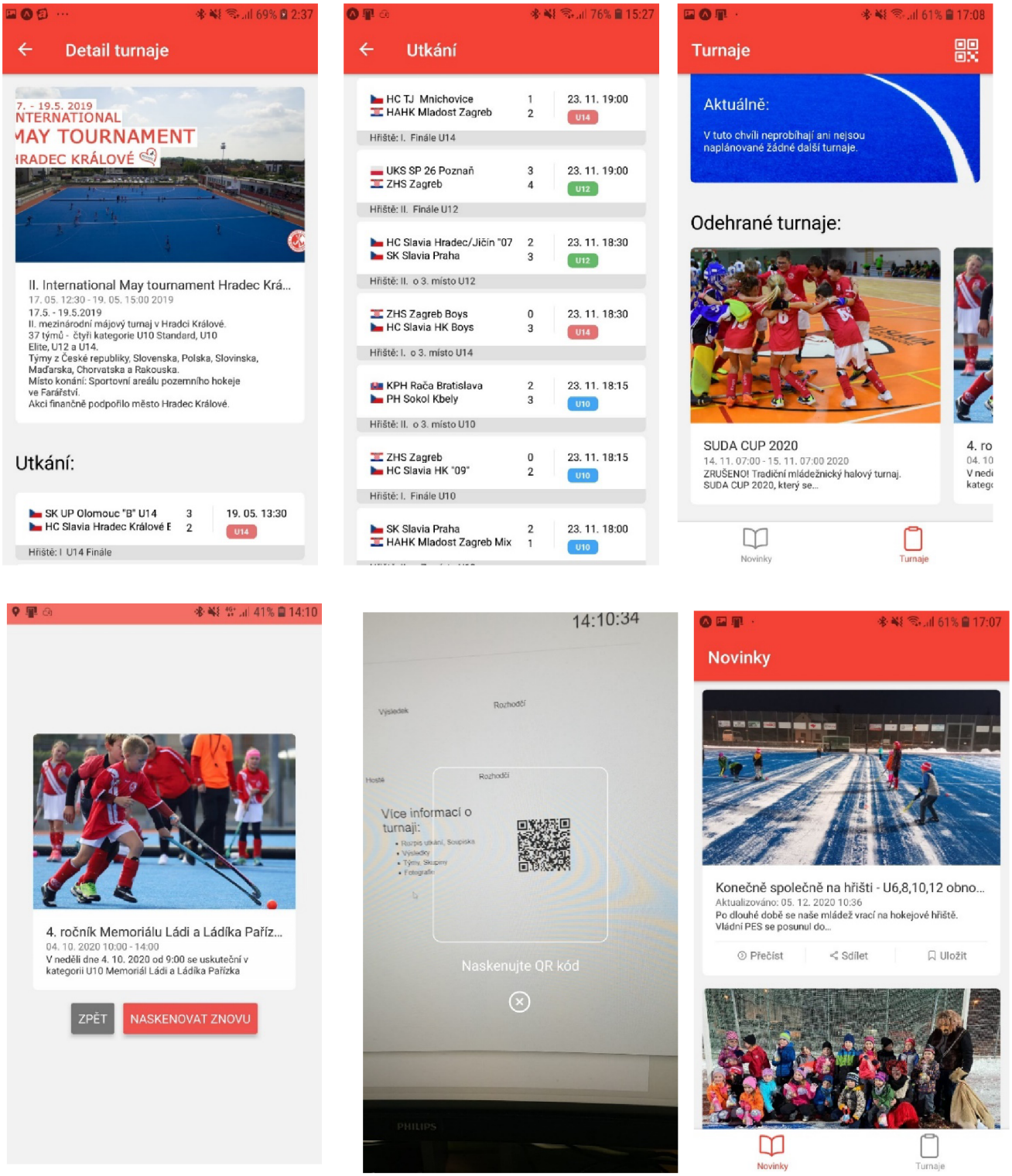
7.4 Shrnutí výsledků

V rámci práce byla v technologii React Native vytvořena mobilní aplikace HockeyOnline. Technologie React Native byla zvolena jako nejvhodnější po předchozím porovnání na základě požadavků na technologii i požadavků na aplikaci. Aplikace je přehledná, rychlá, umožňuje skenování QR kódů a podporuje více jazyků – češtinu a angličtinu. Aplikaci je možné naistalovat ze služby Google Play, kde se nyní nachází v předběžném vydání.

Kromě mobilní aplikace bylo vytvořeno i API, které mobilní aplikaci poskytuje veškerá potřebná data. API je napsáno v technologii ASP.NET Core a stejně jako aplikaci i API je možné dále rozšiřovat. Součástí API je i automaticky generovaná dokumentace. Díky dobře navrženým komponentám byla aplikace rozšířena o další funkce jako je například světlý a tmavý režim či zobrazování ligových utkání.

7.4.1 Ukázka aplikace

Na následující obrázcích jsou zachyceny snímky aplikace.



Obrázek 10 - Snímky obrazovky mobilní aplikace HockeyOnline. Zdroj: [autor]

8 Závěr

Tato práce se zabývala volbou nejvhodnější technologie a následným vývojem mobilní aplikace HockeyOnline, která má být rozšířením webové aplikace HockeyOnline – ta slouží především k vyhodnocování a zobrazování výsledků turnajů pořádaných HC Slavia Hradec Králové, stejný účel by měla mít i mobilní aplikace.

V první části byla rozebrána problematika mobilní platformy, mobilních aplikací a byly popsány základní informace o nejrozšířenějších mobilních operačních systémech Android a iOS. V práci jsou pak napsány požadavky (např. přehlednost, ukládání a zobrazování dat, možnost automaticky přepínat mezi cloudovým a lokálním serverem, skenování QR kódu, umožnění psaní ve škálovatelném znovupoužitelném kódu, podpora REST ASP.NET API a podpora dvou jazyků, mezi nimiž bude možné přepínat…) na funkcionalitu mobilní aplikace, které přímo ovlivňují výběr technologie pro vývoj, což je jedním z cílů práce. Dále jsou vypsány zvažované technologie – Java (Android), Xamarin a React Native, které jsou následně porovnávány, v každé z nich je vytvořena testovací aplikace a na základě získaných informací jsou ke každé z technologií vydána doporučení a učiněn konečný výběr.

Vhodnou technologií byl díky své škálovatelnosti, podpoře multiplatformního vývoje, rychlému a příjemnému vývoji, který zajišťuje framework Expo, zvolen React Native. Tato technologie má velmi dobře zpracovanou dokumentaci, dále obsahuje velké množství předpřipravených komponent a nástroje pro ladění včetně promítání změn v kódu do testovacího zařízení v reálném čase. V rámci práce byla také vypracována další doporučení ohledně volby vhodné technologie i pro jiné typy aplikací. Tato volba se jeví jako správná a vybraná technologie umožnila naplnit požadavky na mobilní aplikaci. Ta je nyní nasazena na Google Play a dostupná uživatelům v předběžném přístupu. Plné nasazení je plánováno na září 2021. Aplikaci je však možné dále rozšiřovat, a to právě díky zvolé technologii i způsobu vývoje. Komponenty, které byly při vývoji vytvořeny, jsou univerzální a je možné je znovu použít a rychle tak přidávat další funkcionality. Již nyní byla aplikace rozšířena o další funkcionality, které v původních požadavcích popsány nebyly. Jedná se například o vylepšení ověřování API klíče, uloženého v hlavičce požadavku či zobrazování ligových utkání nebo přepínání mezi tmavým a světlým režimem, které je dnes velmi populární.

Aplikace byla otestována na několika zařízeních s Androidem i iOS, kde fungovala bez problémů. Kvůli pandemii covid-19 bohužel nebylo zatím možné otestovat aplikaci na žádné sportovní akci.

9 Seznam použité literatury

[1] JORDAN, Taylor B. – MEYERS, Cody L. – SCHRADING, Walter A. – DONNELLY, John P., The utility of iPhone oximetry apps: A comparison with standard pulse oximetry measurement in the emergency department, *The American Journal of Emergency Medicine*, 2020. [online]. [cit. 2021-03-04]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S073567571930467X

[2] Optimize for battery life, *Google Developers.* [online]. [cit. 2021-03-03]. Dostupné z: https://developer.android.com/topic/performance/power

[3] *Statcounter: OS market share* [online]. [cit. 2021-04-15]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202102-bar

[4] BRŮCHA, Filip, Android měl být původně systémem do fotoaparátů, *Computer world,* 2013. [online]. [cit. 2021-03-03].

[5] Android (operační systém), *Wikipedia: the free encyclopedia.* [online]. [cit. 2021-03-03].
Dostupné z: https://cs.wikipedia.org/wiki/Android_(opera%C4%8Dn%C3%AD_syst%C3%A9m)

[6] LACKO, Ľuboslav, *Vývoj aplikací pro Android*, 1. vydání, Brno: Computer Press, 2015, s. 15–65.

[7] *Jak používat službu Play Console: Registrace účtu vývojáře Google Play* [online]. [cit. 2021-03-04]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/6112435?hl=cs#zippy=%2Ckrok-zaplacen%C3%AD-registra%C4%8Dn%C3%ADho-poplatku

[8] IOS, *Wikipedia: the free encyclopedia.* [online]. [cit. 2021-03-04]. Dostupné z: https://cs.wikipedia.org/wiki/IOS

[9] Swift, *Apple Developer.* [online]. [cit. 2021-03-04]. Dostupné z: https://developer.apple.com/swift/

[10] *Purchase and Activation* [online]. [cit. 2021-03-04]. Dostupné z: https://developer.apple.com/support/purchase-activation/

[11] Java (programovací jazyk), *Wikipedia: the free encyclopedia.* [online]. [cit. 2021-03-05].
Dostupné z: https://cs.wikipedia.org/wiki/Java_(programovac%C3%AD_jazyk)

[12] LARDINOIS, Frederic, Kotlin is now Google's preferred language for Android app development, *TechCrunch*, 2019. [online]. [cit. 2021-03-05]. Dostupné z: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

[13] Android Debug Bridge (adb), *Google Developers.* [online]. [cit. 2021-03-05]. Dostupné z: https://developer.android.com/studio/command-line/adb

[14] KENT, Jeffrey A, *Visual C# 2005 - bez předchozích znalostí: průvodce pro samouky*, 1. vydání, Brno: Computer Press, 2007, s. 5-18.

[15] Co je Xamarin? *Microsoft.Docs.* [online]. [cit. 2021-03-06]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/get-started/what-is-xamarin

[16] Dokumentace ke Xamarin.Forms, *Microsoft.Docs.* [online]. [cit. 2021-03-06]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/

[17] *Architektura* [online]. [cit. 2021-03-04]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/android/internals/architecture

[18] LINQ (Language Integrated Query), *Microsoft.Docs.* 2017[online]. [cit. 2021-03-06]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/programming-guide/concepts/linq/

[19] TypeScript, *Wikipedia: the free encyclopedia.* [online]. [cit. 2021-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/TypeScript

[20] React Native DOM, *Use React Native*, 2018. [online]. [cit. 2021-03-04]. Dostupné z: http://www.reactnative.com/react-native-dom/

[21] *Introduction to Expo.* [online]. [cit. 2021-03-17]. Dostupné z: https://docs.expo.io/

[22] *HTTPS.* [online]. [cit. 2021-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/HTTPS

[23] *Representational State Transfer* [online]. [cit. 2021-03-04]. Dostupné z: https://cs.wikipedia.org/wiki/Representational_State_Transfer

[24] *SOAP* [online]. [cit. 2021-03-04]. Dostupné z: https://www.ibm.com/docs/cs/rsm/7.5.0?topic=standards-soap

[25] MASLAKOWSKI, Mark, *Naučte se MySQL za 21 dní*, 1. vydání, Praha: Computer Press, 2001, str. 15-122.

[26] What is NoSQL?, *Mongodb.live.* [online]. [cit. 2021-03-29]. Dostupné z: https://www.mongodb.com/nosql-explained

10 Seznam obrázků

11 Seznam grafů

12 Ukázky kódu

13 Seznam zkratek

ADB – Android Debug Bridge

API – Application Programming Interface

APK – Android Application Package

C# – C Sharp

DB – Database

GPS – Global Positioning Systém

GSM – Groupe Spécial Mobile

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

LINQ – Language Integrated Query

NoSQL – No Structured Query Language

NPM – Node.js package manager

REST – Representational State Transfer

SDK – Software Development Kit

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

SSL – Secure Sockets Layer

TLS – Transport Layer Security

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**David Podzimek**
Studium:	I1800680
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Vývoj mobilní aplikace pro sportovní portál**
Název bakalářské práce AJ:	Mobile app development for a sports portal

Cíl, metody, literatura, předpoklady:

Cílem práce je navrhnout a vytvořit mobilní aplikaci pro sportovní portál HockeyOnline. Práce se zaměří na tvorbu API, aplikace pro platformu Android a optimalizace aplikace.

Osnova:

1. Úvod
2. Mobilní aplikace
3. Vývoj
4. Analytika a optimalizace
5. Zpětná vazba uživatelů
6. Závěr

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Michal Macinka
Datum zadání závěrečné práce:	14.1.2018