VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE VEDENÍ SPORTOVNÍHO ODDÍLU

APPLICATION FOR SPORT GROUP MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAROSLAV HOLCMAN**
AUTHOR

VEDOUCÍ PRÁCE — **MARTIN KOLÁŘ, M.Sc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Holcman Jaroslav, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Aplikace Vedení Sportovního Oddílu**
Application for Sport Group Management
Kategorie: Uživatelská rozhraní
Zadání:

1. Prostudujte postupy návrhu uživatelských rozhraní moderních mobilních aplikací.
2. Formou konzultací prostudujte konkrétní požadavky, a zakomponujte je do produktové studie podle principů vývoje UX.
3. Seznamte se s existujícími aplikacemi sloužícími pro usnadnění fungování neziskových organizací. Na základě tohoto studia definujte požadavky pro podobný systém a nedostatky existujících aplikací.
4. Vyberte vhodné metody a nástroje a navrhněte uživatelské rozhraní aplikace, která slouží jako databáze účastníků a umožňuje vytvořit a spravovat různé organizační činnosti.
5. Navržený systém implementujte s vámi vybraným frameworkem. Systém musí obsahovat docházkový systém, koordinační systém, komunikační systém, a zobrazování novinek oddílu.
6. Proveďte uživatelské experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
7. Vytvořte plakát A2 prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Nielsen, J.: Iterative User Interface Design. 1993
- Nielsen, J.: Parallel & Iterative Design + Competitive Testing = High Usability. 2015
- Preece, J.; Rogers, Y.; Sharp, H.; aj.: Human-Computer Interaction
- Google: Material Design
- Zendulka, J.: Databázové systémy IDS, Studijní opora. 2006

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kolář Martin, M.Sc.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Tato práce popisuje postup vytvoření mobilní aplikace určené pro podporu každodenního fungování sportovního oddílu. Hlavním cílem aplikace je umožnění digitalizace a sjednocení co největšího množství běžných organizačních činností v rámci oddílu. Současně je kladen důraz na vytvoření uživatelského rozhraní, které bude zejména intuitivní a jednoduché a umožní přímočaré použití při hlavních činnostech uživatelů. Práce zkoumá principy návrhu uživatelských rozhraní, analyzuje prostředí sportovního oddílu a definuje požadavky na softwarový nástroj pro podporu jeho vedení. Následně navrhuje podobu řešení na základě analyzovaných požadavků a popisuje implementaci výsledného produktu a jeho nasazení do cílového prostředí. Závěrem popisuje způsob testování reálnými uživateli aplikace a závěry tohoto testování, které validují splnění vytyčených cílů.

Abstract

This text describes the process of creating a mobile application for sport club management and its everyday operations. The main goal of the application is to digitize and unite most of common activities in the sport club. Another goal is to create the user interface which will be primarily intuitive and simple to use while providing straightforward ways of usage during most important user use cases. The thesis studies principles for designing user interfaces, analyzes sports team environment and defines the requirements for a software product to support sports club management. The design of the solution follows the analysis of requirements as well as the description of the implementation of the resulting product and its publishing to the real sports club environment. The thesis ends with description of user testing methods and results of the testing which validate accomplishment of defined goals.

Klíčová slova

mobilní aplikace, uživatelské rozhraní, uživatelská zkušenost, vedení sportovního oddílu, Xamarin Forms

Keywords

mobile application, user interface, user experience, sport group management, Xamarin Forms

Citace

HOLCMAN, Jaroslav. *Aplikace Vedení Sportovního Oddílu.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

Aplikace Vedení Sportovního Oddílu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana M.Sc. Martina Koláře. Další informace mi poskytli zástupci sportovního oddílu SK JUVENIS PŘIBYSLAV. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Holcman
16. května 2019

Poděkování

Na tomto místě bych rád poděkoval panu Martinu Kolářovi, M.Sc. za vedení a vstřícný přístup při tvorbě této práce a poskytnuté rady. Dále bych rád poděkoval všem zástupcům florbalového oddílu SK JUVENIS PŘIBYSLAV za poskytnuté informace, ochotnou spolupráci a zpětnou vazbu k praktické části této práce. V neposlední řadě děkuji své rodině a přítelkyni za veškerou morální a materiální podporu při celém studiu a vytváření této práce.

Obsah

1 Úvod **3**

2 Vývoj mobilní aplikace **4**
2.1 Vývoj softwarového produktu 4
2.2 Principy tvorby dobré uživatelské zkušenosti 7
2.3 Metody návrhu uživatelských rozhraní 10
2.4 Prvky uživatelských rozhraní 11
2.5 Specifika mobilních zařízení 14

3 Požadavky na výsledný produkt **18**
3.1 Principy získávání požadavků a komunikace s uživatelem 18
3.2 Požadavky na aplikaci vedení sportovního oddílu 21

4 Studium existujících řešení **27**
4.1 Týmuj 27
4.2 Team App 28
4.3 Teamstuff 29
4.4 Další aplikace 30

5 Návrh postupu a částí řešení **31**
5.1 Rozdělení iterací 31
5.2 Návrh podoby uživatelského rozhraní 32
5.3 Použité nástroje 38

6 Úvodní fáze vývoje **39**
6.1 Základní prvky uživatelského rozhraní 39
6.2 Sekce komunikace uživatelů 40
6.3 Formulářové elementy a deník 41
6.4 Docházka 46

7 Serverová část systému **49**
7.1 Architektura serverové aplikace 49
7.2 Využití služeb třetích stran 50
7.3 Automatizovaná správa 51
7.4 Nasazení serverové aplikace 52

8 Dokončení první verze **53**
8.1 Funkcionalita organizačních záležitostí 53
8.2 Notifikace na straně klientské aplikace 56

8.3 Doplnění uživatelských rolí . . . 57
8.4 Optimalizace výkonu aplikace . . . 59
8.5 Finalizace a nasazení aplikace . . . 60

9 Vyhodnocení a plán vývoje **61**
9.1 Testování během vývoje . . . 61
9.2 Finální uživatelské testování . . . 62
9.3 Rekapitulace a vyhodnocení vytyčených cílů . . . 64
9.4 Směr dalšího vývoje . . . 65

10 Závěr **66**

Literatura **67**

A Diagramy případů užití **69**

B Diagram struktury systému **72**

C Formulář uživatelských testů **74**
C.1 Testování sekce Docházka . . . 74
C.2 Testování formulářů (zaměřeno na Deník) . . . 75
C.3 Testování sekce Brigády . . . 77
C.4 Testování sekce Komunikace . . . 78
C.5 Zjištění celkové zkušenosti s používáním . . . 79

D Obsah přiloženého DVD **82**

Kapitola 1

Úvod

Vedení i poměrně malého sportovního oddílu vyžaduje aktivní zapojení mnoha lidí, kteří musí zodpovědně vykonávat různé činnosti vedoucí k hladkému každodennímu provozu oddílu. Veškeré činnosti je třeba řídit a kontrolovat, stejně tak je nutné mít dobrý přehled o všech osobách zapojených do chodu oddílu a jejich konkrétních rolích a činnostech, které vykonávají. Moderní technologie mohou při vhodném použití tento proces značně zjednodušit a zefektivnit, podobně jako téměř jakoukoliv oblast lidské činnosti.

Softwarový produkt vznikající v rámci této práce je určený právě pro podporu vedení takového sportovního oddílu. Jako hlavní cíle si klade digitalizaci velké části činností a aktivit uvnitř oddílu společně se všemi daty potřebnými pro jejich plnění. Výsledný produkt by měl umožnit především jednoduchou a pohodlnou práci všem budoucím uživatelům.

Text této práce se nejprve věnuje studiu popsaných metod a principů pro vytvoření uživatelského rozhraní s dobrou uživatelskou zkušeností při jeho používání s podrobnějším zaměřením na uživatelská rozhraní na mobilních telefonech. Následně popisuje prostředí sportovního oddílu, ve kterém by výsledná aplikace měla fungovat, analyzuje toto prostředí a současně definuje požadavky na aplikaci určenou pro vedení popsaného oddílu. Dále se zmiňuje o vybraných již existujících systémech s podobným zaměřením a hodnotí jejich případnou vhodnost pro definované cíle.

Druhá část textu se věnuje nejprve návrhu uživatelského rozhraní budoucí mobilní aplikace a následně jeho realizaci. Dále popisuje implementaci serverové části a doplnění veškeré potřebné funkcionality pro dokončení první verze aplikace určené pro nasazení v reálném sportovním oddílu. V závěru práce je popsáno právě nasazení do tohoto reálného prostředí, způsob testování se skutečnými uživateli systému a vyhodnocení výsledků získaných tímto testováním, stejně jako vyhodnocení vytyčených cílů a nástin dalšího vývoje celého systému nad rámec této diplomové práce.

Kapitola 2

Vývoj mobilní aplikace

Vývoj softwarového produktu, který má sloužit pro zefektivnění práce uživatelů v cílovém prostředí, je komplexní problém, který vyžaduje uskutečnění mnoha postupných kroků vedoucích k požadovanému výsledku. První část této kapitoly (kapitola 2.1) má za cíl uvést přehled principů a metodik užívaných při vytváření softwarových produktů.

Součástí vývoje je také snaha o to, aby byl používaný software dodáván v takové podobě, v jaké ho bude moci uživatel začít používat co nejrychleji, co nejefektivněji, co nejbezpečněji a ideálně také s oblibou. Drtivá většina dnešních aplikací komunikuje se svými uživateli pomocí grafických uživatelských rozhraní a návrh těchto rozhraní se tedy stal jednou z kritických částí vývoje nových aplikací. [13] Další části této kapitoly probírají principy návrhu kvalitních uživatelských rozhraní s hlavním zaměřením na uživatelskou zkušenost při jejich používání (kapitola 2.2) a následně specifika těchto principů pro uživatelská rozhraní mobilních aplikací (kapitola 2.5). Studium těchto témat poskytne základ pro zvolení vhodných postupů při realizaci softwarového výstupu této diplomové práce (viz kapitola 5.2).

2.1 Vývoj softwarového produktu

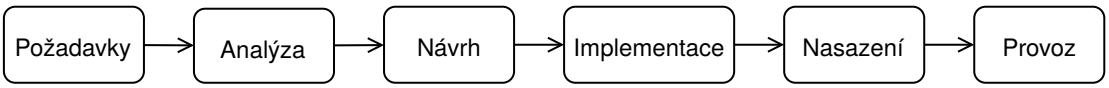
Základem pro softwarové systémy jsou zpravidla skutečné systémy okolo nás. Ve studijní opoře doc. Zendulky [20] je takovýto systém definován následovně: „*Systémem rozumíme kolekci komponent, které jsou ve vzájemném vztahu a které pracují společně k dosažení nějakého cíle.*“ Softwarový systém je potom jakýmsi modelem tohoto reálného systému a slouží nám pro zkvalitnění a usnadnění procesů v tomto systému. Vytvoření dobrého softwaru však není jednoduchý úkol a historicky mnoho projektů vytvářejících nějaký softwarový produkt končilo neúspěšně. Proto byly postupně popisovány a zaváděny určité metodiky do vývoje softwarových produktů, které společně s rozšířením internetu na konci minulého století výrazně přispěly k tomu, že poměr neúspěšných projektů značně poklesl. [20] Tato kapitola popisuje jednotlivé přístupy k tvorbě softwarových produktů a jejich postupný vývoj.

2.1.1 Životní cyklus softwaru

Každý softwarový produkt prochází určitým vývojem, který nazýváme životní cyklus. Životní cyklus softwaru začíná v momentě, kdy vznikne požadavek na jeho vytvoření, a končí v momentě, kdy již produkt přestane být používán. V čase mezi těmito dvěma okamžiky prochází software mnoha postupnými fázemi, které je možné zobecnit a shrnout do následujících pěti hlavních etap (viz také obrázek 2.1):

- Sběr a analýza požadavků
- Návrh systému a jeho součástí
- Implementace
- Integrace a nasazení systému
- Provoz a údržba

Každá z těchto etap sdružuje mnoho dílčích úkonů, které je třeba vykonat pro dosažení dobrého výsledku. Jedním z kritických úkonů, který stojí hned na začátku životního cyklu, je získání a analýza konkrétních požadavků na výsledný produkt, detailněji popsaný v kapitole 3.1. Na základě této analýzy reálného prostředí dojde k návrhu všech potřebných součástí systému, jejich implementace pomocí vhodných nástrojů a metod zvolených na základě definovaných vlastností systému, integrace a nasazení do cílového prostředí. Od této chvíle je produkt udržován, opravován a vylepšován až do okamžiku, kdy již tato činnost není možná nebo se stává příliš nákladná. Poté je software postupně vyřazován a typicky nahrazen novějším systémem, čímž dojde k jeho zániku. [20]



Obrázek 2.1: Etapy životního cyklu softwarového produktu

2.1.2 Modely životního cyklu

Modely životního cyklu software vznikaly postupně za účelem zdokonalení procesu vývoje nových produktů. Všechny modely životního cyklu jsou (více či méně) obecné a udávají pouze rámcový popis toho, co a kdy je vhodné během vývoje dělat. Při jejich aplikaci pro každý jednotlivý projekt je potřeba použitý přístup k řešení upravit pro konkrétní potřeby. Přestože tyto modely jsou definovány pro práci v týmech a mnohé z nich se částečně zaměřují také přímo na způsob spolupráce v těchto týmech, mnoho popsaných metod a principů je možné s úspěchem využít i pro realizaci samostatného projektu, jakým je například právě diplomová práce.

Základním a nejstarším modelem životního cyklu softwarového produktu je vodopád. Vodopád popisuje přímočarý způsob vývoje skládající se z pěti výše uvedených po sobě jdoucích etap. Každá etapa probíhá pouze jednou a je zcela dokončena, výstup každé etapy je poté použit v etapě další. Tento přístup byl v počátcích vývoje software hojně využíván (a i dnes může být užitečný, především u malých projektů), avšak postupem času se ukázal jako neefektivní, protože během vývoje produktu se takřka pokaždé vyskytují určité problémy, které zprvu nebyly zřejmé a na které je potřeba reagovat. Také požadavky cílového zákazníka typicky se začátkem využívání produktu dostávají určitých změn, které je nutné posléze znovu aplikovat. V případě vodopádu je poté potřeba předělat kompletně hotové výstupy dokončených částí, což značně znesnadňuje, prodražuje a prodlužuje vývoj produktu. Z tohoto důvodu se z vodopádového modelu vyvíjely různé modifikace, které spočívaly ve větším propojení jednotlivých částí, případně v prototypování. U malých projektů je možné vodopádový model s úspěchem použít i nadále, avšak u větších projektů postupem času

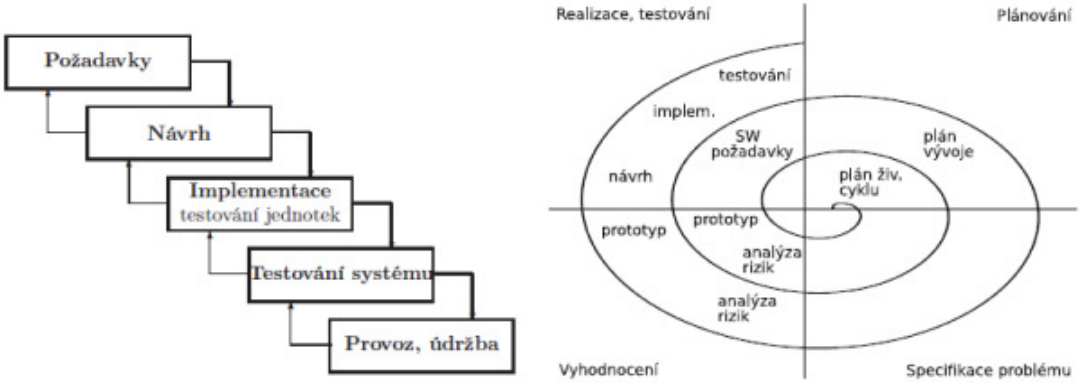
negativa tohoto modelu výrazně převážila pozitiva a v dnešní době již vodopádový model není hojně využíván. [13, 20]

Alternativou jsou iterativní modely. Podstatou iterativních modelů je opakování všech fází vývoje s postupnými přírůstky k výslednému řešení. Výsledkem každé iterace je typicky použitelný produkt, avšak s omezenou funkcionalitou. Tato funkcionalita je s každou další iterací postupně rozšiřována až do konečné podoby kompletního produktu. Jelikož každá iterace je vlastně jeden malý životní cyklus, je možné flexibilně reagovat na změny požadavků a nalezené chyby v návrhu a implementaci. To je také hlavní výhoda iterativních modelů.

Příklady iterativních modelů jsou model spirálový, Unified process, Architektura řízená modelem a dnes velmi oblíbené agilní metodiky. Každý z těchto modelů nabízí odlišný pohled na rozvržení iterací mezi jednotlivými etapami. Unified process a Architektura řízená modelem popisují celkový proces vývoje více podrobně než ostatní a jsou oba úzce spjaté s využíváním modelovacího jazyka UML.

Agilní způsob vývoje software klade velký důraz především na krátké a rychlé iterace, zaměřené na rychlý vývoj a velmi flexibilní reakce na změny v požadavcích na produkt. Jedním z typických rysů je upozadění většiny formálních procesů a dokumentů před lidmi, kteří systém vyvíjí. Dalším rysem je intenzivní napojení realizátora na cílového zákazníka, který je vyzýván k tomu, aby hodnotil dosavadní postup a co nejdříve reagoval ve změnách požadavků. Každá iterace je obvykle zakončena validací, jestli je vytvořený systém skutečně obrazem toho, co cílový zákazník potřebuje. [3, 20]

Spirálový model je iterativním modelem skládajícím se ze čtyř hlavních částí — plánování, analýza cílů a rizik, realizace a vyhodnocení. V každé iteraci je naplánováno, co je jejím cílem. Následně je provedena analýza vytvořeného plánu, jsou identifikovány klíčové součásti a hlavní rizika. Poté je naplánovaná práce realizována, přičemž konkrétní činnost se může v jednotlivých iteracích lišit. V první iteraci pravděpodobně půjde o vytvoření návrhu, v pozdějších iteracích pak o implementaci částí systému. V poslední fázi dojde k vyhodnocení výsledné realizace, je provedeno testování a vytvořena zpětná vazba použitá pro další iteraci. [13, 20] Na obrázku 2.2 je grafické znázornění vodopádového a spirálového modelu.



Obrázek 2.2: Znázornění vodopádového a spirálového modelu (převzato z [8])

2.2 Principy tvorby dobré uživatelské zkušenosti

Požadovaným výstupem návrhu uživatelského rozhraní je dobrá zkušenost uživatelů s používáním aplikace. Dobrý vizuální vzhled je důležitá součást této zkušenosti, nikoliv však jediná. Součástí dobré uživatelské zkušenosti je mnohem více — aplikace musí být pro uživatele dobře použitelná, musí správně a svižně reagovat na jejich vstupní podněty, uspokojivě plnit požadované akce a podobně. [14, 15] Právě o prvcích důležitých pro dobrou zkušenost uživatelů s používáním aplikace pojednává několik následujících odstavců.

2.2.1 Lidský faktor

Klíčovým parametrem pro dobrý návrh uživatelského rozhraní je samozřejmě skupina uživatelů, která s ním bude interagovat. Je důležité konkrétně specifikovat tuto skupinu, popsat její klíčové rysy a následně definovat požadavky, které by měly být při návrhu uživatelského rozhraní patřičně zohledněny právě vzhledem k uživatelské základně. Mnoho rysů je ovšem společných a je možné je uchopit jako obecné principy vztahující se pro tvorbu jakéhokoliv uživatelského rozhraní. Důvodem je, že cílovou skupinou jsou vždy lidé. Lidský mozek pomocí smyslů neustále vnímá a analyzuje okolní prostředí a obrovské množství vjemů, které z něho přichází. Postupem času se mozek naučí mnoho věcí zautomatizovat a bez přímého vědomí reaguje na určité vjemy určitým způsobem. Této skutečnosti je možné při návrhu uživatelského rozhraní aplikace využít a dosáhnout tak větší pohodlnosti pro uživatele a zároveň větší použitelnosti celé aplikace.

Stejně tak je možné využít lidské paměti. Mozek si napoprvé mnohem snáze zapamatuje věci, které mu dávají největší smysl. Pokud je tedy uživatelské rozhraní postavené takovým způsobem, že pro uživatele, který jej vidí poprvé, dává co největší smysl, potom je vyšší pravděpodobnost, že si uživatel při příští příležitosti mnohem snáze vzpomene, jak má s danou aplikací pracovat. Součástí dobrého návrhu uživatelského rozhraní by tedy měla být snaha umístit do něj prvky takovým způsobem, aby uživateli dávaly smysl. Konkrétními příklady mohou být názvy akcí nebo použité ikony. Každá ikona by měla uživateli při pohledu na ni co nejvíce asociovat skutečnou operaci nebo objekt, který na obrazovce zastupuje. Například s emailovým klientem, který by jako tlačítko pro odeslání emailu používal ikonu kočky, by se jeho uživatelům nejspíše moc dobře nepracovalo. Stejný princip platí samozřejmě také pro názvy akcí, které by měly taktéž být co nejintuitivnější a nejvýstižnější.

Speciální oblastí je také způsob oznamování uživateli, že došlo k chybě. Je prakticky nevyhnutelné, že uživatel se během interakce s aplikací dopustí nějaké chyby, která zabrání úspěšnému dokončení požadované akce. Stejně tak je velmi pravděpodobné, že uživatel narazí na nějakou chybu v samotném softwarovém produktu a akce nebude dokončena kvůli této chybě. V obou případech je důležité, aby byl uživatel o chybě informován takovým způsobem, který mu umožní pochopit, z jakého důvodu k této chybě došlo, a následně rozhodnout o dalším postupu. Výstižnost a vhodnost chybového sdělení pomůže uživateli jednak napravit chybu, která se stala, a do budoucna mu umožní se této chyby vyvarovat a zefektivnit tak svoji práci.

Správné použití takto přesných a výstižných prvků ovšem mnohdy nemusí být vůbec jednoduchý úkol. A právě na tomto místě se vracíme zpět k tomu, že je potřeba mít správně definovanou cílovou skupinu uživatelů. Právě na této skupině by výsledný výběr ikon, názvů a podobných prvků měl především záležet. Zároveň také záleží na konkrétním kontextu, v jakém budou uživatelé aplikaci používat, jak často ji budou používat, jaká je jejich zkušenost a znalost technologií a mnoha dalších faktorech. Jedním z faktorů je také skutečnost,

zda je produkt určený pro konkrétní, uzavřenou, předem známou skupinu uživatelů (například je vytvořen na míru pro určitou firmu) nebo se jedná o generický software, který bude k dispozici pro všeobecné využití a konkrétní skupiny uživatelů určujeme podle zaměření a cíle aplikace, nikoli podle konkrétní reálné skupiny lidí. Často se může stát, že uživatelé aplikace jsou velmi různorodí a není možné určit jednu konkrétní definici cílové skupiny. V takovém případě je nutné najít určitý kompromis, který bude co nejvíce vyhovovat všem, případně umožnit uživatelům postupovat různým způsobem (například s využitím různých způsobů interakce pro stejný případ použití systému — viz kapitola 2.4.1).

Zároveň je ovšem také potřeba mít na paměti ještě jeden aspekt. Uživatelé jednak potřebují uživatelské rozhraní co nejintuitivnější, aby jej mohli začít využívat efektivně co nejrychleji. Po nějaké době používání však získají zkušenosti a mnoho úkolů pro ně začne být automatických a budou se v aplikaci orientovat mnohem lépe. Určité prvky, které zpočátku napomáhali intuitivnosti, však v takovou chvíli mohou začít uživatele spíše zdržovat. Mezi těmito dvěma aspekty je potřeba vytvořit určitou rovnováhu, která je typicky součástí definice použitelnosti (viz kapitola 2.2.2) konkrétního produktu. [13, 15]

2.2.2 Užitečnost a použitelnost

Pro každý softwarový produkt s uživatelským rozhraním je důležité, aby byl pro koncového uživatele užitečný. Užitečný se stává, pokud uživateli poskytuje takové funkce, které potřebuje, a zároveň je pro práci dobře použitelný. Použitelnost konkrétní aplikace se dá popsat pomocí několika atributů:

- Jak obtížné je pro uživatele zvládnout úkol napoprvé.
- Jak rychle dokáže uživatel plnit úkoly po zaučení s aplikací.
- Jak obtížné je pro uživatele znovu si vzpomenout na práci s aplikací po určité době.
- Kolik chyb a jak závažných uživatel při práci dělá.
- Jak příjemné je pro uživatele používání aplikace.

Ačkoliv všechny tyto atributy jsou důležité z hlediska použitelnosti, ne vždy je nutné, aby byly mezi sebou vybalancované. Vždy je potřeba přizpůsobit vyhodnocení těchto atributů konkrétnímu projektu a mnohdy stačí u některých atributů dodržet nějakou minimální akceptovatelnou mez, zatímco u jiných je nutné dosáhnout maximální kvality, a to třeba i na úkor některého z jiných atributů. Obecně lze považovat všech pět atributů za důležité a hodné dostatečné pozornosti, lze si ale představit i případy, kdy jeden z nich převáží nad všemi ostatními. Aplikace může například vyžadovat, aby zaučený uživatel mohl pracovat s maximální efektivitou, aniž by bylo důležité, kolik úsilí, času a prostředků bude nutné věnovat na jeho zaučení. V některých případech zase může záležet především na tom, aby množství uživatelských chyb bylo naprosto minimální, a tomuto cíli je celý návrh přizpůsoben. Je tedy vždy důležité zohlednit požadavky konkrétního produktu a podle nich klást důraz na jednotlivé atributy použitelnosti uživatelského rozhraní. [10, 11]

Použitelnost je důležitá jak pro aplikace, které se snaží zaujmout předem nekonkretizované koncové uživatele (jinak aplikaci nebudou používat), tak také pro uživatele, kteří s ní z nějakého důvodu pracovat musí (protože špatně použitelná aplikace je bude při práci zbytečně zdržovat). Použitelnost aplikace je velmi důležitou součástí uživatelské zkušenosti s používáním každého softwarového produktu a je tedy na místě věnovat jí dostatečnou pozornost a prostředky. [10]

2.2.3 Důležité rysy uživatelské zkušenosti

Kromě použitelnosti je nedílnou součástí zkušenosti s aplikací mnoho dalších faktorů. Jedním z nich je samozřejmě vizuální podoba uživatelského rozhraní. Existují situace, kdy je možné do značné míry upustit od snahy o líbivé použití barev, fontů a podobných čistě vizuálních prvků, aniž bychom tím uživatelům znepříjemnili práci. I v takovém případě je však důležité dbát na vhodné rozložení elementů na obrazovce, dobrou čitelnost, jasné odlišení jednotlivých prvků a podobně. Použité barvy, fonty, označení a další by měly být konzistentní v rámci celé aplikace. Stejně tak způsob zobrazování dat by měl být pro stejná data konzistentní a ve všech případech by měl zohledňovat charakter dat a zobrazovat je srozumitelně. Zobrazovaná data by měla vždy být pro uživatele relevantní a užitečná, není vhodné rozptylovat uživatelovu pozornost zbytečnými daty, která mu v práci nijak nepomáhají. V neposlední řadě by mělo zobrazení informací na obrazovce být takové, aby uživatel vždy měl kontext viditelných dat a nemusel si pamatovat informace při přecházení mezi obrazovkami.

Uživatelova pozornost může být také na obrazovce směrována směrem k určitým částem. K tomuto účelu je možné využít různých způsobů, například orámování, označení (šipkou, tečkou,...), výraznou barvou apod. Je vhodné využít různých barev a velikostí prvků pro odlišení částí uživatelského rozhraní, ale je třeba dbát opatrnosti — příliš mnoho různých stylů a jejich kombinací působí naopak velmi rušivě. Počet různých velikostí, barev a fontů by neměl překročit čtyři a mnohdy postačuje i menší počet. Prvky s podobným vzezřením jsou poté uživatelem vnímány jako související, což je efekt, kterého by si návrhář měl být vědom a využívat ho vhodným způsobem při tvorbě návrhu.

Z hlediska uživatelské zkušenosti je důležitá také očekávatelnost chování uživatelského rozhraní. Systém by měl reagovat na akce uživatele co nejočekávatelnějším způsobem. V opačném případě dochází ke zmatení uživatele a práce se systémem se pro něj stává mnohem náročnější a nepohodlnější. Podobně negativně na uživatele působí chyby aplikace, složitost a nepřehlednost uživatelského rozhraní zabraňující provedení požadované operace nebo například dlouhá odezva systému. Uživatelé mají zkušenosti s chováním jiných aplikací a mají přehled o dobách reakcí těchto aplikací v rámci různých situací. Pokud se z nějakého důvodu doba odezvy výrazně prodlužuje nad očekávanou mez, dochází opět k značnému poklesu spokojenosti na straně uživatele.

Jako dobrou praktiku lze označit poskytování zpětné vazby na uživatelské akce. Například drobné animace při přechodech mezi stránkami, po kliknutí na objekt, rozbalování a sbalování nabídek a další velmi přispívají k pocitu plynulosti a dobré uživatelské zkušenosti s používáním aplikace. Podobně lze vnímat i potvrzování provedených operací, kdy uživatel dostane od aplikace zřetelně najevo, že určitý úkon byl úspěšně dokončen. Toto se samozřejmě netýká každé akce, kterou uživatel provede, ale při dokončení nějaké posloupnosti akcí tvořících dohromady určitý komplexnější úkon (například vyplnění a uložení formuláře) je potvrzující zpětná vazba uživatelem velmi pozitivně vnímána.

K zlepšení uživatelské zkušenosti s používáním aplikace velmi napomáhá také aktivní snaha zabránit uživatelským chybám. Grafické odlišení neaktivních tlačítek a položek v nabídce je dobrým příkladem takového chování, stejně jako snaha zamezit uživateli ve vložení nevalidního znaku do vstupního pole. V případě, že k chybě skutečně dojde a operace nemůže být dokončena, získávají na důležitosti chybová hlášení. Ta by měla být pro uživatele co nejsrozumitelnější, měla by výstižně popisovat problém, ke kterému došlo, a v ideálním případě také navrhnout možnost řešení. Obecně je možné vždy preferovat pozitivně laděnou konstruktivní zprávu, která informuje uživatele o vzniklém problému a snaží se ho navést

k úspěšnému dokončení akce. Zachování uživatelem vyplněných dat při chybě a umožnění opravit pouze ta nevalidní bez nutnosti nového zadávání všech informací je taktéž důležitá praktika výrazně zlepšující zkušenost s aplikací. [13, 15]

2.3 Metody návrhu uživatelských rozhraní

Jelikož navržení kvalitního uživatelského rozhraní je komplexní a náročný úkol, byly popsány metody a postupy, které umožňují dosáhnout uspokojivého výsledku. Níže jsou vybrány hlavní principy těchto metod.

2.3.1 Iterativní návrh

Návrh uživatelského rozhraní takovým způsobem, aby odpovídal všem konkrétním požadavkům dané aplikace, byl dostatečně dobře vizuálně vypadající a zároveň dobře použitelný pro koncového uživatele, je velmi obtížný úkol, který je takřka nemožné splnit pomocí jediného procesu. Proto je v podstatě ve všech případech vhodné navrhovat uživatelské rozhraní iterativně. Iterativní způsob návrhu umožňuje po vytvoření prvotní verze postupně odhalovat a opravovat problémy v použitelnosti uživatelského rozhraní za pomoci uživatelských testů zaměřených právě na problémy v použitelnosti.

Každá iterace se skládá z vytvoření konkrétního návrhu, který je poskytnut vzorku uživatelů k provedení určitých úkolů, a sesbírání zpětné vazby během a po dokončení těchto úkolů. Následně je třeba provést analýzu této zpětné vazby a na jejím základě identifikovat problémy, navrhnout jejich řešení a toto řešení aplikovat v rámci další iterace. Konkrétní počet potřebných iterací nelze obecně stanovit a zákonitě se bude lišit produkt od produktu, lze ovšem říci, že iterací by mělo obecně proběhnout alespoň několik. První iterace běžně objeví kritické chyby v návrhu, které významně brání použitelnosti aplikace, a umožní jejich opravu. Další iterace jsou potřebné zejména ze dvou důvodů. Jedním je skutečnost, že po odstranění skutečně závažných chyb si testovací uživatelé začnou více všímat drobnějších nedostatků, které lze posléze také postupně odstraňovat. Druhým důvodem je fakt, že s každým zásahem do již vytvořeného systému podstupujeme určité riziko zanesení nového problému.

Se vzrůstajícím počtem iterací se snižuje úroveň zlepšování použitelnosti produktu, avšak žádný maximální počet iterací není stanoven. Lze ovšem předpokládat, že v reálném prostředí bude tento iterativní proces omezen určitými limity ve zdrojích, např. finančních nebo časových. U mnoha produktů (především méně rozsáhlých) je také pravděpodobné, že po několika iteracích se použitelnost dostane do uspokojivých mezí a není již zapotřebí provádět další. [11, 12]

2.3.2 Paralelní návrh

Využití samotného iterativního návrhu s sebou nese jedno riziko — iterativně se vylepšuje prvotně vytvořený návrh uživatelského rozhraní, který ovšem sám o sobě nemusí vůbec být ideální. Řešením je využití paralelního návrhu. Paralelní návrh spočívá ve vytvoření několika nezávislých verzí uživatelského rozhraní (nebo určitých kritických částí) a na základě jejich srovnání dosáhnout optimálního řešení. Toto řešení vzejde buď výběrem jednoho z vytvořených návrhů, nebo jejich sloučením a kombinací nejlepších částí a nejsilnějších stránek jednotlivých návrhů.

Podobně lze také využít tzv. kompetitivního testování, které je velice podobné paralelnímu návrhu, avšak testovány nejsou návrhy vytvořené pro požadovaný produkt, ale různé již existující podobné produkty. Cílem je především získat více znalostí o chování uživatelů a jejich způsobu práce s danou aplikací. Na základě těchto informací je poté možné lépe navrhnout odpovídající řešení pro daný produkt.

Pro vytvoření co nejlepšího konečného uživatelského rozhraní je vhodné zkombinovat oba popsané přístupy. Paralelní přístup je užitečný především v počáteční fázi, kdy je třeba vytvořit celkovou koncepci rozhraní a je zde nejvíce prostoru pro vytváření více odlišných řešení stejného problému. Na základě informací získaných prvním uživatelským testováním všech těchto řešení vykrystalizuje jedno finální řešení, které je posléze zdokonalováno pomocí iterativního přístupu. V určitých situacích je samozřejmě možné využít paralelních návrhů kdykoliv během dalších iterací, ale mělo by se jednat pouze o konkrétní dílčí části uživatelského rozhraní, je-li potřeba vyřešit například umístění nějakého funkčního prvku, a podobně. [12]

2.3.3 Uživatelsky orientovaný návrh

Uživatelsky orientovaný návrh je označení přístupu, kdy návrhář nechává reálným uživatelům dostatek prostoru pro vyjádření jejich skutečných potřeb namísto vytváření vhodného rozhraní pouze na základě popsaných požadavků samotným návrhářem. Cíloví uživatelé by se měli účastnit celého procesu návrhu, diskutovat vlastnosti systému, hodnotit vytvořené návrhy a případně navrhovat jejich pohled na řešení určitého problému.

Ke správnému pochopení a vyložení uživatelských požadavků je potřeba vést diskuzi s uživatelem o tom, jaké jsou jeho potřeby, co od aplikace očekává a jaké úkony potřebuje provádět. Je třeba pochopit jeho motivaci pro používání aplikace a také studovat již existující podobná řešení, jejich hodnocení od uživatelů a způsob, jakým se dané aplikace postupně rozvíjely. Pro vytvoření dobré uživatelské zkušenosti s používáním aplikace je při návrhu uživatelského rozhraní důležité skutečně porozumět cílovým uživatelům a jejich motivaci pro práci se systémem. Neméně důležité je definovat různá omezení, která aplikaci mohou ovlivňovat.

Jedním z principů dobrého návrhu řešení interakce uživatele se systémem je pochopení celkového systému, ve kterém se interakce odehrává. Kromě motivace uživatele k použití systému je třeba pochopit širší souvislosti, ve kterých k této interakci dochází. Stejně tak je třeba vzít v úvahu všechny účastníky systému, a to jak aktivní uživatele, tak i pasivní účastníky, kterých se činnost také dotýká. Tyto znalosti poté napomáhají vytvoření co nejpřesnějšího produktu, který se bude v dané situaci pro uživatele chovat co nejpříhodněji. [13, 19]

2.4 Prvky uživatelských rozhraní

Při tvorbě interaktivních uživatelských rozhraní se opakovaně setkáváme s prvky, které taková rozhraní potřebují řešit. Typicky se jedná o navigaci aplikací a způsob komunikace mezi uživatelem a systémem, který uživatelské rozhraní představuje.

2.4.1 Typy interakcí

Zvolený typ interakce uživatele se systémem je velmi důležitý pro výslednou použitelnost systému. Výběr stylu interakce probíhá na základě definice uživatelů, jejich úkolů, celko-

vých požadavků na systém a dat, se kterými pracuje. Můžeme popsat pět základních stylů interakce:

- přímá manipulace — objekty jsou reprezentovány grafickými ekvivalenty, se kterými uživatel manipuluje (ikony, náhledy fotografií,...)
- výběr z menu — uživatel vidí svoje možnosti ve formě položek menu a vybírá požadovanou akci
- vyplnění dat — uživatel určitým způsobem vkládá data na odpovídající místo (vyplňuje textové pole, vybírá z položek,...)
- příkazy — uživatel používá definovanou množinu (typicky textových) příkazů pro provádění požadovaných operací
- přirozený jazyk — uživatel používá přirozenou řeč, kterou popisuje systému svůj požadavek a systém na něj reaguje

Ve většině reálných situací se neobejdeme bez kombinování těchto stylů dohromady. Je proto vždy důležité správně odhadnout, pro jakou situaci se který styl nejvíce hodí. Zároveň je možné také použít více typů interakcí pro stejný úkol, typicky například zavedením možnosti vkládání textových příkazů namísto ovládání grafického uživatelského rozhraní můžeme umožnit expertům výrazně efektivnější práci se systémem. [15]

2.4.2 Navigace

Jednou z nejdůležitějších částí větších aplikací je navigace, neboli menu. Menu by mělo být navržené takovým způsobem, aby uživatel mohl rychle přecházet mezi různými částmi systému, a zároveň by měl mít při navigaci systémem přehled o tom, kde se nachází. Položky v menu by měly být jednoznačné a měly by výstižně popisovat, kam odkazují. Konkrétní navigace je samozřejmě vždy tvořena s ohledem na konkrétní aplikaci a její potřeby, nicméně je možné identifikovat čtyři nejvýznamnější typy navigace:

- hlavní navigace — uživatel má na výběr mezi několika položkami, které představují všechny části aplikace
- sekvence — navigace je tvořena posloupností několika výběrů, které jsou závislé na předchozích volbách
- stromová struktura — menu je tvořeno stromovou strukturou uspořádaných položek
- cyklické a acyklické sítě — dosáhnout jednotlivých destinací je typicky možné pomocí více cest a opakovatelných možností

Jelikož menu jako takové neslouží přímo k efektivní práci uživatele, ale pouze jako prostředek, jak se dostat k požadované části aplikace, nastává problém v tom, že menu velkou část času pouze zbytečně zabírá místo na obrazovce. Tento problém je typicky řešen pomocí různě rozbalovacích menu, kdy je po celou dobu viditelné pouze tlačítko, které menu zobrazuje (případně lišta tlačítek zobrazujících různá podmenu), a zbytek menu je skryt.

Problém začlenění menu do zbytku aplikace tak, aby nepůsobilo rušivě a zároveň dobře plnilo svou funkci, ovšem není ten největší problém při tvorbě menu, který musíme řešit. Hlavním problémem je struktura a uspořádání položek, zejména pokud je položek mnoho a mají více úrovní zanoření. Typické způsoby uspořádání položek v menu jsou:

- abecední
- kategorické — v případě, že položky lze rozdělit do logických kategorií
- konvenční — položky nelze rozdělit do kategorií, ale přesto lze najít logické uspořádání (např. dny v týdnu)
- podle frekvence použití

Výběr správného zanořování a řazení položek v menu potom vždy závisí na konkrétním případu a není možné stanovit jedno univerzální optimální řešení. Je ovšem možné říci minimálně to, že pokud existuje nějaký logický způsob vytvoření menu jinak než abecedně (typicky kategoricky), ve většině případů bude vhodnější využít tohoto způsobu. V případě řazení položek menu podle frekvence použití lze předpokládat zvýšení efektivity při vybírání nejvýše umístěných položek s nejčastějším využitím. Nicméně sestupné řazení níže umístěných položek působí na uživatele spíše rušivě a rychlost nalezení konkrétní položky snižuje. Řešením je vybrání několika důležitých nejčastěji používaných položek, které budou umístěny prioritně na vrcholku struktury menu, a organizace zbytku položek například abecedně. Princip vybírání nejčastěji použitých voleb uživatele svádí k dynamickému přizpůsobování menu podle statistických dat získaných od konkrétního uživatele, tato praktika se však také ukázala jako rušivá a matoucí. [13, 15]

Důraz na úsporné a efektivní menu je ještě mnohem výraznější na zařízeních s menšími obrazovkami, jako jsou tablety, a především mobilní telefony. O problému navigace v aplikacích na mobilních zařízeních pojednává kapitola 2.5.3.

2.4.3 Zobrazení dat

Jedním z typických cílů uživatelských rozhraní je prezentace aplikačních dat uživateli. Správné zobrazení těchto dat je jednou z klíčových částí pro efektivní práci a celkovou dobrou zkušenost uživatele s používáním aplikace. Je nutné prezentovat všechna potřebná data v takovém formátu, aby uživatel dokázal rychle identifikovat pro něj důležité informace a naložit s nimi dle potřeby. Pro dosažení tohoto efektu je nutné, aby zobrazení dat bylo přehledné a všechna data byla zobrazena v odpovídajícím formátu — například zobrazení data ve formátu *01-01-2018* je mnohem vhodnější než *01012018*, přestože oba zápisy reprezentují stejnou informaci.

Dalším pravidlem, které je nutné dodržovat, je poskytování kontextu a významu informací. Zobrazené datum samo o sobě nemá žádnou vypovídající hodnotu, pokud nevíme, co toto datum znamená. Každá zobrazená informace by měla mít přesný popis, například formou vysvětlujícího popisku, názvu sloupce tabulky a podobně. Data se stejným významem je vhodné zobrazovat stejným vizuálním stylem a podobně dobrou praktikou je i shlukování souvisejících dat. [15]

2.4.4 Vkládání informací

Jedním z kritických bodů interakce uživatele se systémem je vkládání dat. V téměř každém systému bude vyžadován nějaký druh vstupu od uživatele. Proces vkládání tohoto vstupu může ovšem být značně zdlouhavý a vést k chybám. Tomu lze zabránit vhodným návrhem těchto vstupů. Jednak je dobré snažit se eliminovat vstupní akce na minimum, zejména v případě vyplňování redundantních informací. Nutné údaje samozřejmě uživatel vyplnit musí, ale měl by mít situaci co nejjednodušší. Zejména v případě textových vstupů

v nějakém očekávaném formátu je velmi vhodné nahradit tento textový vstup adekvátní formou výběru z připravených možností (typickým příkladem této situace je zadávání data nebo času). Výběr z několika možností je možný například pomocí několika vzájemně se vylučujících tlačítek (tzv. *radio-buttons*), k vybrání možnosti ano nebo ne zase můžeme využít zaškrtávacího tlačítka (*check-box*) a podobně. Tímto způsobem je možné velmi zpříjemnit práci se systémem a stejně tak dosáhnout i menší pravděpodobnosti chyby ze strany uživatele.

Ke zlepšení zkušenosti s typickým formulářovým vkládáním dat je však zapotřebí více než pouze správné použití způsobu získání dat od uživatele. Uživatel musí mít jasné instrukce o tom, proč a kam má jaká data umístit. Rozložení formuláře musí být logické a přímočaré, povinné položky by měly být jasně vyznačeny, celkový vzhled formuláře by měl být pro uživatele příjemný. Pro vstupy ve specifickém formátu je vhodné konkrétní formát uživateli naznačit. [15]

2.5 Specifika mobilních zařízení

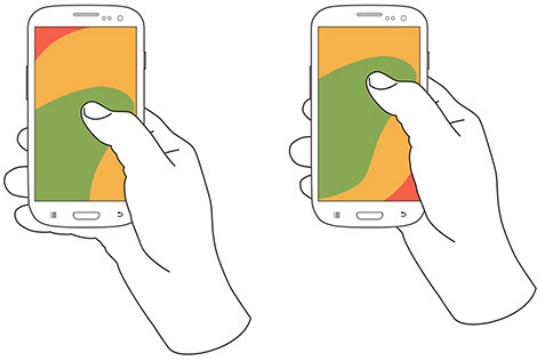
Mobilní zařízení jsou specifická díky velikosti jejich obrazovky a také způsobem ovládání, kdy uživatel využívá dotykové obrazovky k provádění požadovaných akcí. Tyto vlastnosti s sebou nesou potřebu navrhovat uživatelské rozhraní aplikací poněkud odlišným způsobem. Aplikace se snaží využít každého místa obrazovky k zobrazení užitečné informace, zároveň musí zajistit dobrou čitelnost a dobrou použitelnost při ovládání dotykem.

2.5.1 Interakce s uživatelem

Specifický způsob komunikace mezi mobilním zařízením a uživatelem je nutné při návrhu uživatelského rozhraní zohlednit. Typickým způsobem interakce je dotykové ovládání přímo prostřednictvím obrazovky, kdy uživatel využívá primárně své prsty — a ve většině případů pouze palec své dominantní ruky. Díky tomu může často docházet k situaci, že uživatel má problémy palcem pokrýt celou plochu obrazovky (viz obrázek 2.3). Důležité prvky (například menu) poté může být výhodné umístit ke spodnímu okraji obrazovky, aby k nim měl uživatel co nejpohodlnější přístup. Je třeba zohlednit skutečnost, že uživatelé (především uživatelé s většími prsty) mohou mít problémy s používáním příliš malých objektů na obrazovce. Zobrazené elementy musí mít dostatečnou velikost a také musí být umístěny dostatečně daleko od sebe, aby nedocházelo k mylným stisknutím špatného elementu. Toto pravidlo však jde přímo proti snaze dostat na malou obrazovku mobilního zařízení co nejvíce informací a aktivních prvků. Jako obecné vodítko lze použít poučku, že na cílovém zařízení by každý aktivní prvek měl zabírat plochu přibližně 1 cm^2. Je však na citu návrháře, aby na základě cílové skupiny uživatelů posoudil ideální velikosti prvků. [9]

Z důvodu velikosti obrazovky se většina aplikací nevyhne nutnosti posouvání obrazovky k zobrazení veškerého dostupného obsahu (anglicky *scrolling* — dále budeme používat počeštěného výrazu *skrolování*). Vertikální skrolování je zaběhnutým zvykem většiny aplikací snad ve všech oblastech (web, mobilní aplikace,...), pro uživatele je přirozené a intuitivní. Naproti tomu horizontální skrolování není využíváno příliš často a nehodí se pro posouvání zobrazeného obsahu stránky, typicky na webu (výjimkou jsou například široké tabulky).

V mobilních aplikacích ovšem horizontální skrolování může najít velmi dobré uplatnění. Zejména při zobrazení většího počtu položek stejného významu (jako jsou fotografie, zboží určité kategorie,...) může být velice výhodné umožnit listování těmito položkami horizontálním způsobem. Tento přístup ušetří vertikální prostor aplikace pro zobrazení jiných dat



Obrázek 2.3: Ukázka dostupnosti částí obrazovky pomocí palce jedné ruky (převzato z [2])

a udržuje celkový vzhled stránky kompaktní. Při tomto použití ovšem hrozí určitá rizika — uživatelé nejsou zvyklí využívat horizontální skrolování, je tedy nutné vhodně naznačit tuto možnost na odpovídajícím místě. Stejně tak je potřeba naznačit konec skrolovatelného obsahu. [16]

2.5.2 Sbírání uživatelského vstupu

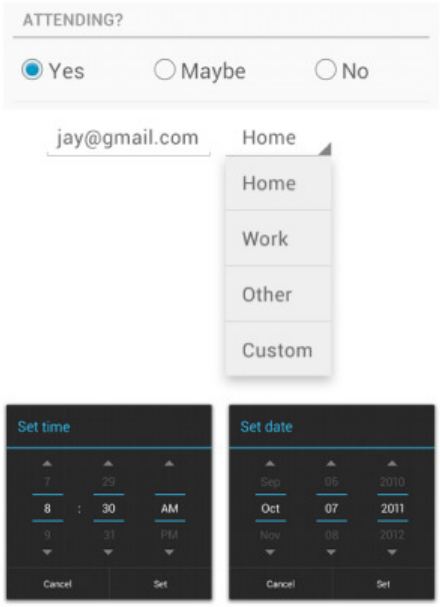
Na mobilních zařízeních nastává několik problémů v oblasti vkládání vstupu ze strany uživatele. Samotné psaní na nepříliš velké dotykové obrazovce je pro mnoho lidí poměrně nepříjemná a zdlouhavá práce. V kapitole 2.4.4 je zmíněna potřeba vybrat vhodný způsob získání informace od uživatele. Na mobilních zařízeních toto platí obzvlášť. Vhodně zvolený způsob výběru z předpřipravených možností může velice urychlit vyplnění i poměrně krátkého formuláře.

Může se ovšem projevit jiný problém — vzhledem k malé obrazovce jsou nativní prvky určené pro výběr z možností typicky vytvořeny tak, že prvek vystoupí do popředí a překryje zbytek formuláře. Uživatel v takovém případě ztrácí kontext, což v určitých situacích může být velice nežádoucí. Dalším problémem těchto prvků je také potřeba několika akcí pro zadání jedné hodnoty — nejprve je nutné stisknutím určené položky otevřít výběrový prvek, následně odpovídajícím způsobem vybrat zvolenou hodnotu a poté celý prvek zavřít. Při použití více takovýchto komponent v jednom formuláři se značně zvyšuje počet jednotlivých akcí, které musí uživatel uskutečnit. Ukázka několika typických vstupních prvků v operačním systému *Android* je na obrázku 2.4. Prvky na obrázku jsou převzaté z *Android API Guide* stránek *radio-buttons*[1], *spinners*[2] a *pickers*[3].

Zajímavou možností jak nahradit nativní komponenty pro výběr je využití klasického textového pole, které při zadávání textu od uživatele začne našeptávat možné volby, které částečně odpovídají vloženému řetězci. U číslicových polí je zase možné využít speciálního vstupního pole, které disponuje tlačítky pro inkrementaci a dekrementaci jeho hodnoty. Stejně tak lze s výhodou využít různých způsobů výběru stisknutím jednoho z několika tlačítek. Ne vždy je však možné či vhodné snažit se použití typických prvků nahrazovat.

Vyplnění formuláře může být zefektivněno předvyplněním možných dat. U některých položek formuláře lze odhadnout, jaká by mohla být uživatelem vyplněná hodnota (například

[1] https://developer.android.com/guide/topics/ui/controls/radiobutton.html
[2] https://developer.android.com/guide/topics/ui/controls/spinner.html
[3] https://developer.android.com/guide/topics/ui/controls/pickers.html



Obrázek 2.4: Některé vstupní prvky na platformě *Android*.

dnešní datum, nejčastější volba,...) a tuto hodnotu automaticky předvyplnit. Uživatel poté může pole upravit podle svých představ nebo pouze zkontrolovat, že hodnota je v pořádku a posunout se dále.

Z důvodu snahy o ušetření co nejvíce místa na obrazovce se často může stát, že návrhář odstraní popisky vstupních polí a nahradí je dočasnými hodnotami uvnitř těchto polí, které naznačují očekávaný vstup, a po vyplnění jsou tímto vstupem od uživatele nahrazeny (princip označovaný jako *placeholder*). Úskalí tohoto přístupu spočívá ve faktu, že dočasná hodnota uvnitř vstupního pole po vyplnění zmizí a tím uživatel přichází o informaci, jaký je význam vyplněných dat. Z tohoto důvodu je preferovaný způsob umístění popisku vstupního pole mimo něj a využití *placeholderu* pro ilustraci očekávaného vstupu (například *Jan Novák*, *email@domena.cz*,...). [9]

2.5.3 Menu

Vzhledem k omezené ploše aplikací na mobilních zařízeních jsou principy vytváření menu poněkud odlišné od aplikací na velkých obrazovkách. Navigace musí ustoupit do pozadí z důvodu potřeby efektivní prezentace dat. Důraz je kladen především na jednoduchost, intuitivnost a rychlost provedení nejčastějších akcí. Často zde proto hraje roli frekvence využití jednotlivých položek. Velmi často používané akce by měly být jednoduše přístupné, zatímco méně využívané operace je možné (a často velice žádoucí) umístit někam stranou hlavního dění, typicky do podmenu. [1, 15]

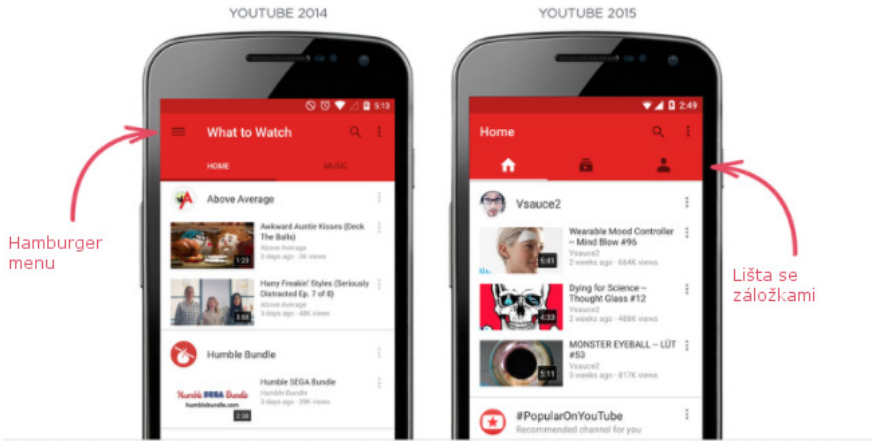
Často používaným typem navigace je takzvané *Hamburger menu*. Principem tohoto stylu navigace je ponechání celého prostoru obrazovky pro hlavní obsah aplikace a odsunutí celé navigace pryč. Pomocí k tomu určeného tlačítka je možné kdykoliv zobrazit celou navigaci, která překryje obsah stránky a umožní uživateli výběr akce. Výhodou tohoto způsobu je právě ponechání celé obrazovky aplikačním datům a také možnost umístit do navigace velké množství položek, přičemž je umožněno pohodlné vyčlenění prioritních položek. Negativem

je pak především ztráta kontextu v rámci aplikace, zhoršení intuitivnosti a potřeba jedné akce od uživatele navíc při snaze o přesunutí do jiné části aplikace.

Jiný způsob navigace, který v současnosti velmi získává na oblibě, je navigace pomocí lišty záložek (anglicky *Tab bar navigation*). Několik hlavních částí aplikace je reprezentováno pomocí záložek, které jsou umístěny v jedné společné liště na vrchním nebo spodním okraji obrazovky (z důvodu použitelnosti pomocí palce popsaném v kapitole 2.5.1 je doporučováno využití spodního okraje). Lišta je pro uživatele viditelná po celou dobu používání aplikace a umožňuje jak udržení kontextu, tak také velmi rychlý pohyb mezi klíčovými částmi aplikace. Počet záložek by se měl pohybovat mezi třemi a pěti, případné další možnosti lze schovat do podmenu reprezentovaného pomocí jedné z nich. Pro reprezentaci jednotlivých záložek se typicky používají ikony. Ikony by měly být co nejvýstižnější, nicméně je vhodné, aby byly vždy doplněny jednoznačným popiskem, který vystihne význam použité ikony. Aktuálně aktivní záložka by měla být vizuálně zvýrazněna tak, aby uživatel na první pohled poznal, kde se v aplikaci nachází.

Dva výše popsané způsoby řešení menu jsou nejčastěji používané v dnešních aplikacích (ukázka na obrázku 2.5). Nicméně ve vhodných případech lze využít i mnoha dalších způsobů, jak uživateli umožnit navigaci skrze aplikaci. Jednou z možností může být využití dlaždic, tlačítek nebo ikon, které tvoří hlavní stránku aplikace a přemístí uživatele na jinou odpovídající stránku, ze které se může kdykoliv vrátit zpět a pokračovat jinam. Tento princip navigace je typicky označován jako *dashboard*. Další možností může být úplné vynechání grafického menu a navigace aplikací pomocí gest. Tento způsob však vyžaduje zvýšenou snahu uživatele získat přehled o způsobu navigace a naučit se aplikaci dobře ovládat bez vizuální pomoci, což může v mnoha případech způsobit problémy.

V určitých aplikacích může být velice výhodné navrhnout zcela specifický způsob navigace, který bude pro tuto aplikaci vhodný. Ve většině případů však není na škodu držet se zaběhnutých zvyklostí, které uživatelé znají, a vybrat si dobře známé řešení, které je možné upravit pro potřebu konkrétního uživatelského rozhraní. V mnoha případech je také možné kombinovat různé způsoby navigace. Základem je ale vždy vytvořit takovou navigaci, která bude pro koncové uživatele nejpříjemnější a nejlépe použitelná. [1, 2, 5]



Obrázek 2.5: Ukázka dvou častých řešení navigace použitých v různých verzích aplikace *Youtube* (převzato z [1])

Kapitola 3

Požadavky na výsledný produkt

Sběr a analýza dostupných informací a požadavků na cílový produkt je první, a kritickou, fází životního cyklu softwaru (viz kapitola 2.1.1). Právě nedostatečné, nepřesné nebo často se měnící požadavky patří mezi hlavní důvody prodlužování a prodražování, případně celkového neúspěchu realizace softwarových projektů. Důvodem obtížnosti této etapy jsou především dvě skutečnosti. První z nich spočívá v tom, že není jednoduché správně a přesně definovat reálný systém a také všechny konkrétní požadavky na jeho softwarovou abstrakci. Druhou je poté problém v komunikaci mezi člověkem popisujícím tyto požadavky (typicky neprogramátorem) a člověkem realizujícím výsledný produkt (tedy programátorem). Stejně jako pro ostatní etapy vývoje produktu, i pro získání a analýzu požadavků na software vznikly určité osvědčené metody a dobré praktiky, které má za cíl přiblížit kapitola 3.1. Požadavky na cílový produkt vznikající v rámci této diplomové práce získané a popsané pomocí těchto metod jsou poté náplní kapitoly 3.2. [15, 19]

3.1 Principy získávání požadavků a komunikace s uživatelem

Cílem fáze sběru a analýzy požadavků je pochopení a popsání uživatelů, kteří budou s produktem pracovat, jejich motivaci, očekávání, potřeby, které má produkt naplnit, úkoly, které budou pomocí produktu vykonávat, a mnoho dalších více či méně důležitých informací. Výstup této části je velice důležitý pro další fáze vývoje, které z tohoto studia reálné situace vychází.

3.1.1 Požadavky na software

Fáze získávání požadavků je fáze, kde se střetávají očekávání a potřeby jednak zákazníka, který potřebuje nasazení požadovaného produktu, uživatelů, kteří budou systém skutečně používat, a vývojáře (či vývojářů), kteří systém vytvářejí. Je důležité najít průnik mezi očekáváním těchto tří pohledů na systém a nalézt společnou představu o tom, co je potřeba vytvořit.

Jako požadavky chápeme takové informace, které mají vliv na výslednou podobu systému. Může se jednat o informace o požadované funkčnosti či chování systému, podmínky, které musí systém naplňovat, splnitelné cíle při používání aplikace a mnoho dalších. Požadavky na systém můžeme rozdělit do tří kategorií — funkční, datové a požadavky na použitelnost.

Funkční požadavky se dají popsat jako soubor toho, co musí systém umět. Definujeme, jaké úkony musí být možné se systémem provést ze strany uživatele, jaké úkony má systém

provádět automaticky a podobně. Stejně tak je důležité popsat i všechna omezení a úkony, které naopak systém umožnit nesmí.

Datové požadavky popisují strukturu systému a definují data, která se v systému nachází. Tato část požadavků je důležitá pro pochopení významu dat v systému a jejich vzájemných vztahů.

Zatímco předchozí dva typy požadavků se soustředí především na systém a jeho vlastnosti, požadavky na použitelnost určují, co vše je nutné splnit z hlediska efektivní a příjemné práce uživatelů se systémem (viz kapitola 2.2.2). Tyto požadavky by měly popisovat míru, s jakou je nutné dodržet intuitivnost ovládání, jednoduchost učení při práci se systémem, rychlost provádění potřebných akcí, spolehlivost a výkon systému a dalších. Kromě toho se soustředí také na popis prostředí a kontextu, v jakém uživatelé produkt používají, a jaký je jeho vliv na práci s produktem. [13, 19, 20]

3.1.2 Zdroje informací a požadavků

Existuje několik typických možností, jakými jsou potřebné informace o prostředí a požadavky na systém shromažďovány. Primárním způsobem získávání požadavků je přímý rozhovor se zákazníkem a uživateli. Jako podpůrný prostředek pro získání dat přímo od uživatelů je možné využít dotazníky. Dotazníky by však neměly nahrazovat osobní rozhovory, protože se jedná o pasivní způsob komunikace. Výhodou však může být, že respondenti mají dostatek času na rozmyšlení odpovědi.

Důležitým zdrojem informací může být pozorování reálného systému, které může dobře posloužit pro pochopení všech souvislostí a motivací pro zavedení produktu do tohoto systému. Pozorovatel se může aktivně zapojit do procesu a tím získá náhled a zkušenost, ze které může významně těžit při návrhu systému. Je třeba dát pozor na to, aby uživatelé skutečně pracovali běžným způsobem a neupravovali svoje chování na základě toho, že je vnější osoba pozoruje.

Pro získání náhledu do reálné situace je dobré i studium prostředí jako takového včetně studia různých dokumentů a dalších materiálů z cílové oblasti, které pomáhají dokreslit celkový obraz cílového systému. Výhodné mohou být také konzultace s některými lidmi, kterých se vyvíjený systém přímo netýká, ale mohou poskytnou užitečné informace o prostředí, ve kterém bude systém provozován (typicky odborníci na danou oblast, lidé se zkušenostmi s vývojem podobných produktů,...). [13, 20]

3.1.3 Získávání požadavků

Hned na začátku získávání informací je vhodné odhalit a popsat určitou vizi, kterou má požadovaný systém naplnit a hlavní cíle tohoto systému. Vznikne tak jakýsi základní model specifikace požadavků, ze kterého je možné dále vycházet a hledat konkrétní a úplné požadavky, které bude nutné splnit.

Další informací, kterou je nutné získat co nejdříve, je definice cílové skupiny nebo skupin uživatelů. Je nutné pochopit charakteristické rysy těchto skupin, jejich zkušenosti s používáním podobných systémů a další faktory, které tyto skupiny reprezentují. Zároveň je třeba určit role jednotlivých skupin uživatelů a také pravomoci a omezení jednotlivých rolí. Při pozdějším získávání dalších informací od potenciálních uživatelů je zapotřebí zahrnout zástupce všech těchto skupin. [20]

Po pečlivém určení cílových uživatelů je nutné objevit také způsoby jejich interakce s budoucím produktem. Je důležité identifikovat jednotlivé úkoly, které budou uživatelé vykonávat, pochopit uživatelovu motivaci za těmito úkoly a frekvenci, s jakou tyto úkoly

bude vykonávat. Na základě těchto údajů je potom možné rozhodnout mnoho otázek v rámci návrhu uživatelského rozhraní. Například způsob, jakým je vytvořena navigace uživatele v aplikaci, bude velmi záležet na povaze a frekvenci jednotlivých úloh, stejně tak na jejich podobě. [15]

Po získání základních informací a vytvoření prvotního náhledu na systém je možné dále pokračovat a postupně nalézt všechny případy užití, data v systému, širší souvislosti a všemožné detaily, stejně jako vnější rozhraní a formu komunikace s dalšími systémy. Je nutné mít na paměti, že v této fázi není možné efektivně odhadnout, které informace je nebo není možné zanedbat. Veškeré shromážděné materiály by proto měly být považovány za potenciálně důležité a rozhodnutí o jejich skutečné relevanci by mělo být ponecháno až na pozdější analýzu těchto materiálů (tomu se věnuje kapitola 3.2.3).

Při rozhovorech s uživateli je vhodné klást jednak rozmyšlené předpřipravené otázky, tak také využít nestrukturovaného způsobu rozhovoru, kdy si povídáme o systému a potřebách zákazníka či uživatele. V rámci rozhovorů je dobré kromě výše popsaných požadavků (v kapitole 3.1.1) zjistit zejména konkrétní úkony, které provádějí se stávajícím systémem, co se jim na aktuálním řešení líbí a co nelíbí, jaké mají při provádění úkonů problémy a jaké návrhy mají oni sami ohledně vylepšení tohoto systému. Tyto informace mohou být velmi užitečné při vytváření komplexního pohledu na požadované řešení.

V případě vedení rozhovorů s více různými osobami, které navíc mohou zastávat různou funkci v rámci systému (typicky vedení, které chce zavést systém a uživatel, který ho používá), může dojít k rozporům a nesrovnalostem v získaných požadavcích. V takovém případě je nutné provést validaci požadavků, kdy dojde k usměrnění všech požadavků správným směrem.

Důležitým aspektem rozhovorů s uživateli je komunikace *jejich řečí*. Je třeba vést rozhovor takovým způsobem, aby byl průměrný uživatel schopný odpovídat co nejpřesněji. K tomu je zapotřebí používat jazyk, kterému uživatel rozumí a ptát se ho na jeho potřeby a jeho zkušenosti. Pokud budeme uživatele nutit mluvit řečí informatika (nebo návrháře,...) a klást odborné otázky ohledně fungování budoucího systému, je velmi pravděpodobné, že uživatel nebude schopen adekvátně odpovídat a získané informace nebudou ideální.

Při sběru požadavků je nutné dávat jednotlivé informace od uživatelů do kontextu, snažit se odhalit nevyslovené požadavky a takové požadavky, které logicky vyplívají z poskytnutých informací. Stejně tak je třeba poukazovat na nesrovnalosti a rozpory a snažit se vést uživatele takovým směrem, aby získané informace byly co nejúplnější a nejvýstižnější. Při hledání všech důležitých informací je užitečné hledat drobné detaily. Mnoho zdánlivě skrytých požadavků je možné získat otázkami na důvody nějakého chování, faktory bránící dokončení úkolů, hledání alternativních cest provedení úkolů a podobně. Je vhodné se snažit již ve fázi sbírání požadavků promýšlet návrhy možných řešení některých problémů nebo cílů systému a získávat názory uživatelů na tyto návrhy. [7, 19, 20]

3.1.4 Analýza požadavků

Po sesbírání všech dostupných informací a požadavků ze všech dostupných zdrojů je nutné provést jejich analýzu. V rámci analýzy je třeba projít získané materiály, strukturovat a prioritizovat informace a převést je do takové formy, která je poté využitelná pro technický návrh a implementaci řešení. Informace získané od uživatelů typicky nejsou ve stručné a přehledné formě a je tedy nutné jim tuto formu dát v rámci analýzy.

V rámci analýzy požadavků je užitečnou metodou definování takzvaných případů užití. Případy užití popisují jednotlivé akce, které můžou různí uživatelé (neboli *aktéři*) se systé-

mem vykonávat. U některých (zejména komplexních) případů užití je poté vhodné vytvořit jejich detaily, které popisují způsob provedení daného úkonu, vstupní a výstupní podmínky, možné problémy a omezení při jeho realizaci a další informace.

Během analýzy je dobré rozdělit jednotlivé požadavky podle jejich priority. Prioritní požadavky poté tvoří jakési jádro systému, minimální model, který musí systém splnit a ze kterého je možné vycházet pro pozdější upřesnění požadavků a doplnění nových. Tento přístup je výhodný zejména při iterativním způsobu vývoje (viz kapitola 2.1.2), kdy je možné dokončit minimální potřebnou část co nejdříve a umožnit uživatelům její používání. Následně je možné získat zpětnou vazbu uživatelů na dokončenou část systému a podle potřeby zareagovat v následující iteraci.

Výstupem fáze analýzy požadavků je obvykle dokument popisující tyto požadavky. Formát tohoto dokumentu závisí na potřebě konkrétního projektu. U menších projektů často stačí pouze slovní popis přirozeným jazykem. U velkých projektů je vhodné využití nějakého standardizovaného způsobu, který umožňuje vytvořit mezi požadavky určitou hierarchii, která udržuje požadavky přehledné.

Často mohou v rámci analýzy požadavků také vznikat různý typy diagramů, které napomáhají orientaci v nich a slouží jako základ pro pozdější fáze, především pro návrh řešení systému. Typicky se jedná například o diagramy případů užití (tzv. *use-case diagram*), diagram vztahů datových entit v systému (*ER diagram*), diagramy toků dat (*dataflow diagram*) a další. [19, 20]

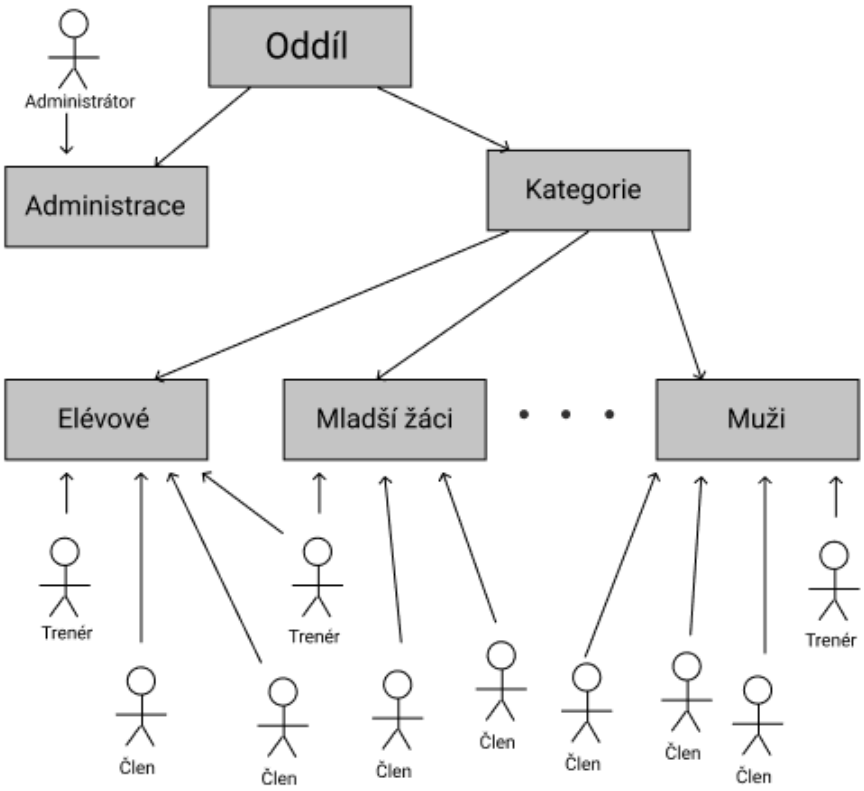
3.2 Požadavky na aplikaci vedení sportovního oddílu

Hlavním výstupem této diplomové práce bude aplikace sloužící jako nástroj pro vedení sportovního oddílu. Pomocí konzultací s vedením tohoto oddílu, dalšími potenciálními uživateli této aplikace a také sesbíráním znalostí pomocí osobního pozorování provozu oddílu byly sesbírány dostupné informace a požadavky, které má aplikace splňovat. Sesbírané informace o prostředí, popis a analýza požadavků a také vize do budoucna jsou obsahem následujících odstavců.

3.2.1 Popis cílového prostředí a stávající situace

Cílovým prostředím pro výslednou aplikaci je malý florbalový oddíl. Oddíl je složený z několika věkových kategorií od předškolních dětí až po dospělé hráče. Každá z kategorií má určitý počet členů adekvátního věku, přičemž u dětských kategorií je nutné zapojení rodičů, kteří zastupují daného hráče (své dítě). V rámci klubu funguje několik trenérů, kteří jsou různým způsobem spojení s různými kategoriemi. Za každou kategorii je určený zodpovědný trenér, který má za úkol řešit veškeré záležitosti s danou kategorií spojené. Celý oddíl je potom zastřešen jedním člověkem (dále označovaným jako administrátor), který se stará o vedení oddílu jako celku a dohlíží na chod jednotlivých kategorií. Zjednodušená struktura oddílu je zobrazená pomocí obrázku 3.1. Typický chod oddílu můžeme ilustrovat pomocí příkladu na jedné z kategorií.

Kategorie má po celou sezonu vymezené časové úseky, které má každý týden k dispozici k uskutečnění svých tréninků ve sportovní hale. Účast všech členů kategorie na těchto trénincích je povinná a v případě neúčasti člena je vyžadována odůvodněné omluvení z konkrétního tréninku. Docházka jednotlivých členů je sledována a zaznamenávána trenéry dané kategorie. V případě jakýchkoliv problémů v době tréninků mají přítomní trenéři povinnost zapsat takovou událost do klubového deníku, který slouží jako evidence těchto neoče-



Obrázek 3.1: Zjednodušené znázornění struktury oddílu

kávaných událostí. Mezi takové události patří například zranění některého z členů v době tréninku, poškození vybavení sportovní haly, zrušení tréninku v daném termínu a podobně.

Kromě tréninků se hráči dané kategorie účastní také soutěžních zápasů, které se typicky konají ve víkendových dnech formou takzvaných turnajů, kdy se více různých týmů v dané kategorii dostaví do sportovní haly jednoho z nich a odehrají několik utkání podle konkrétního losu dané soutěže. Z pohledu konkrétního oddílu lze tedy turnaje rozdělit na dva typy — domácí (oddíl je zodpovědný za organizaci turnaje) a venkovní (oddíl dojíždí na turnaj organizovaný jiným oddílem). V případě venkovního turnaje je zodpovědností trenéra, aby zajistil dostatečný počet řidičů (typicky z řad rodičů), kteří pomohou s přepravou dětí na místo konání turnaje. Dále má na starost vyplnění soupisky týmu podle přítomných členů k danému utkání a stejně jako u tréninků kontrolu docházky jednotlivých členů.

V případě domácího turnaje musí trenér kromě docházky členů vyřešit také obsazení pořadatelské služby, která se typicky skládá z členů a trenérů jiných kategorií a také několika dalších lidí spolupracujících s oddílem. Stejně tak má trenér na starosti kontrolu vyplněného zápisu o utkání (vyplňuje pořadatelská služba a externí rozhodčí) a odeslání tohoto zápisu do sídla *České florbalové unie*. S konkrétním domácím turnajem pak mohou být spojené ještě další specifické povinnosti, které se mohou v čase lišit.

Administrátor má zodpovědnost za administrativní úkony spojené se všemi členy napříč kategoriemi. Jedná se zejména o vybírání členských příspěvků, aktivaci hráčských a trenérských licencí v rámci systému *České florbalové unie* (případně jejich kontrolu, po-

kud si aktivaci řeší přímo členové), přihlašování týmů do soutěží odpovídajících kategorií a vyplnění jejich soupisek a dalších potřebných informací.

Velká část výše popsaného chodu oddílu je závislá na komunikaci mezi zapojenými osobami. Komunikace uvnitř oddílu (tedy mezi trenéry a administrátorem) probíhá v rámci vytvořené komunikační skupiny v aplikaci *WhatsApp*. Komunikace mezi lidmi uvnitř oddílu a ostatními (členové, rodiče členů a lidé podílející se na organizaci) probíhá pomocí přímé komunikace (typicky za využití mobilních telefonů), případně alternativními přímými kanály (chat na *Facebooku*, *Viberu*,...). Kategorie mužů používá pro své potřeby skupinu na *Facebooku*. Přihlašování členů na organizační práce při pořádání turnajů a také předběžné přihlašování rodičů jako řidiči na venkovní turnaje probíhá pomocí tabulek dostupných na webu, do kterých se mohou zapsat na požadovaný termín. Potvrzení účasti těchto osob a případné změny nebo doplnění chybějících osob řeší zodpovědný trenér pomocí běžně využívaných způsobů komunikace popsaných výše.

Pro šíření informací směrem od klubu k členům a rodičům (a také případné veřejnosti) slouží také webové stránky klubu, na kterých se objevují důležité informace pro jednotlivé kategorie, povinnosti členů a trenérů, důležité termíny a podobně. Docházka členů na tréninky a turnaje je řešena pomocí papírových docházkových sešitů. Soupisky jednotlivých kategorií a informace o hráčích této kategorie jsou taktéž řešeny papírovým způsobem. Klubový deník sloužící jako evidence událostí je veden jako sdílená složka s dokumenty ve službě *Google docs*.

3.2.2 Neformální popis požadavků

Na samém začátku definice požadavků na produkt je vhodné popsat základní vizi, kterou má produkt naplnit (viz kapitola 3.1.3). V tomto případě má vytvořená aplikace sloužit všem lidem účastnících se běžného chodu popsaného oddílu. Hlavní cíle aplikace jsou sjednocení způsobu komunikace mezi účastníky, umožnění šíření informací od vedení klubu k ostatním aktérům vhodnou formou, digitalizovat způsob vedení docházky členů včetně zautomatizování evidence omluvenek a také převedení klubového deníku mimo dosavadní sdílený prostor a usnadnění jeho vyplňování.

Uživateli aplikace by měli být všichni lidé popsaní v kapitole 3.2.1 — tedy členové a rodiče členů, lidé napomáhají organizaci turnajů, trenéři a administrátor. Aplikace by měla počítat s budoucím zavedením funkce *sekretář*, což by měl být člověk, který převezme část pravomocí administrátora.

Komunikace v aplikaci by měla být možná jak mezi dvěma uživateli, tak také v hromadných skupinových konverzacích, rozdělených do nadřazených kategorií. Administrátor by měl mít možnost vytvářet kategorie a konverzace v nich, stejně jako spravovat přiřazení ostatních uživatelů do těchto konverzací. Aplikace by také měla zobrazovat novinky a důležité informace zveřejněné na webových stránkách oddílu a vhodnou formou také případná krátká upozornění vytvořená administrátorem. Jedním z hlavních problémů současného řešení je neochota osob zjišťovat si informace a pravidelně sledovat dění na webových stránkách, které tyto informace poskytují. Tento problém by aplikace též měla vhodným způsobem adresovat.

Další důležitou součástí je vedení docházky a klubového deníku. V rámci docházky by měl každý člen mít možnost vyplnit omluvenku pro určitý termín a trenéři poté zkontrolují a případně doplní docházku na daném termínu a potvrdí do systému. Záznamy do klubového deníku by měly být vytvářeny pomocí předpřipravených formulářů.

Výše popsané funkce jsou vyžadovány jako základ aplikace spolu s administrátorskými možnostmi, zejména v rámci vytváření a editace komunikačních skupin a také přístupu k záznamům v deníku a jejich správu. Administrátor by také měl mít možnost správy členů a přiřazování je do jednotlivých rolí.

Po otestování tohoto základního nástroje a zjištění dojmů uživatelů by pak měla být funkcionalita postupně rozšiřována na základě pozdější domluvy s vedením oddílu. Předběžně je očekáváno přidání přehledu povinností jednotlivých uživatelů a hlídání jejich plnění ze strany systému, vytvoření procesu pro přihlašování osob na termíny (organizace turnajů, řidiči na venkovní turnaje), zobrazování dat o týmech ze systému *České florbalové unie* (zápasy a výsledky, tabulka soutěže, statistiky hráčů,...), zahrnutí různých manuálů (například pro pořadatele turnajů), správa uživatelského profilu včetně případného nastavení aplikace a případné další požadované funkce, doplněné během vývoje.

3.2.3 Analýza požadavků

Na základě sesbíraných informací a požadavků shrnutých v kapitole 3.2.2 vznikla úvodní představa o podobě výsledného systému. Na přání vedení oddílu bude klientská část systému vytvořená jako mobilní aplikace. Hlavním důvodem je snaha o zjednodušení přístupu pro uživatele, možnost pracovat alespoň omezeným způsobem i bez připojení k internetu a také možnost notifikací, které mají cílit především na zajištění lepšího šíření informací. Serverová část (bude-li to možné) vznikne jako součást již fungující webové stránky klubu. Pro administrátorské potřeby by také mohlo být užitečné vytvoření (alespoň zjednodušeného) webového či desktopového rozhraní, které by umožnilo pohodlnější plnění funkcí administrátora systému.

Cílová skupina uživatelů je poměrně různorodá. Vzhledem k zahrnutí rodičů hráčů v dětských kategoriích mezi budoucí uživatele a také aktuálnímu složení trenérského týmu v oddílu je nutné brát skupinu uživatelů jako osoby téměř libovolného věku (při zahrnutí hráčů dostáváme skupinu uživatelů od nejnižšího věku po přibližně 55 let) s různorodou zkušeností s podobnými aplikacemi a moderní elektronikou jako takovou. V rámci zachování použitelnosti aplikace pro všechny uživatele je tedy třeba předpokládat, že uživatelé jsou v tomto směru spíše méně zdatní, a na toto při návrhu pamatovat. Uživatelské rozhraní aplikace by mělo klást hlavní důraz na intuitivnost a jednoduchost používání, přímočarost hlavních případů užití a snadnou orientaci v rámci aplikace. Uživatelské role je možné vyčlenit do následujících šesti skupin:

- Nepřihlášený uživatel — automaticky přiřazená role pro každého nového uživatele. Má právo prohlížet si veřejně dostupný obsah (výsledky týmů, novinky z webu,...). Přiřazení do jiné role provede administrátor.
- Člen nebo rodič člena — má přístup k funkcím určených pro členy jako je docházka, přehled povinností člena, možnost přihlášení na organizaci turnaje a podobně.
- Trenér — má přístup k funkcím trenéra, například ke kontrole vyplnění docházky členů přidělené kategorie, přidání záznamu do deníku, správa organizačních záležitostí přiděleného turnaje a další.
- Organizátor — osoba podílející se na pořádání turnajů (například zdravotníci). Má přístup k funkcím spojeným s organizačními záležitostmi.

- Sekretář — aktuálně neexistující role v rámci oddílu, která by měla být výhledově doplněna. Jedná se o osobu (případně osoby), která převezme část pravomocí od administrátora. Konkrétní podoba této role bude upřesněna později.

- Administrátor — správce celého oddílu a systému. Má přístup k veškeré funkcionalitě, zejména správě ostatních uživatelů, správě deníku, správě kategorií v klubu, skupin a konverzací atd.

Přihlašování uživatelů by podle požadavků administrátora mělo být ideálně vyřešeno pomocí telefonního čísla uživatele, které bude sloužit jako identifikátor konkrétního uživatele, a podle kterého poté administrátor může přiřadit uživatele do požadovaných rolí.

Funkční požadavky na systém jsou podle sesbíraných informací následující:

- Komunikace uživatelů — komunikace mezi různými osobami v rámci oddílu, a to jednak přímá osobní komunikace, a pak také skupinové konverzace. Skupinové konverzace by mělo být možné dělit do kategorií a jednotliví uživatelé mají na základě příslušnosti k dané kategorii přístup ke konverzacím v této kategorii. Kromě textových zpráv by aplikace měla umožňovat také sdílení souborů, případně další moderní prvky obvyklé u podobných komunikačních systémů.

- Šíření informací — zobrazení novinek a informací publikovaných na webových stránkách klubu. Aplikace by měla reflektovat rozdělení těchto informací do kategorií a umožnit uživatelům přepínání mezi těmito kategoriemi, stejně jako zvolení kategorií, u kterých požadují přijímání upozornění na publikované novinky.

- Docházka členů — členové a rodiče členů by měli mít možnost omluvit svoji neúčast na nadcházejícím termínu tréninku nebo turnaje. Trenéři poté docházku zkontrolují, doplní a potvrdí.

- Klubový deník — trenéři (případně organizátoři) vyplňují předem definované formuláře o nastalém problému nebo události. Informace jsou uloženy a přístupné pro administrátora.

- Přihlašování členů na výpomoc — aplikace má umožnit členům, rodičům a organizátorům přihlásit se na výpomoc v rámci organizace (případně výjezdu) turnajů. Všichni uživatelé by měli mít přehled o termínech konání a obsazenosti těchto termínů. Trenér zodpovědný za konkrétní turnaj může tyto záznamy upravovat dle případné potřeby, typicky v případě výpomoci nějaké osoby mimo oddíl.

- Přehled povinností — každý uživatel má mít přehled o svých povinnostech a o stavu jejich plnění. V případě členů se jedná zejména o platbu členských příspěvků, aktivaci hráčské licence, dodání všech administrativních dokumentů a výpomoc na přihlášených turnajích. U trenérů tato část poté zahrnuje splnění všech činností spojených s turnaji, jako je zajištění obsazenosti všech organizačních funkcí, odeslání zápisů o utkání a dalších povinností, které budou definované později.

- Notifikace — *push notifikace*[1] upozorňující uživatele na příchozí zprávy, nové informace z webu, které se jich týkají, blížící se turnaje, potřebu splnit určitou povinnost

[1] Anglický výraz *push notification* označuje takové notifikace (neboli upozornění), které přichází zvenčí bez přímého vyžádání. Tento počeštěný tvar bude v dalším textu používán pro sémantické oddělení právě upozornění typických pro mnohé aplikace na mobilních zařízeních (které zprostředkovává operační systém zařízení) od upozornění zobrazovaných uvnitř samotné aplikace.

daného uživatele atd. Zároveň by měla v rámci aplikace fungovat stránka zobrazující přehled těchto notifikací, včetně možnosti administrátora vytvářet jím definovaná upozornění.

- Přístup k výkonnostním datům — zobrazení zápasů, výsledků, statistik, tabulek a dalších informací jednotlivých týmů převzatých ze systému *České florbalové unie*.
- Zobrazení dat o kategorii určených pro trenéry — jedná se zejména o soupisku dané kategorie a všech informací o hráčích v této kategorii, které trenér potřebuje například pro vyplnění soupisky týmu ke konkrétnímu utkání.
- Manuál — různé manuály pro trenéry a organizátory (v případě potřeby bude doplněno i pro členy), například návod na zprovoznění časomíry ve sportovní hale, návod na vyplnění zápisu o utkání, seznam sportovních hal ostatních týmů a jejich adres a podobně.
- Napojení na *FIS* (informační systém *České florbalové unie*) — možnost pro administrátora vyplňovat potřebné údaje do *FISu*, například registrace týmů na sezonu, vyplnění soupisek a údajů těchto týmů a podobně.
- Profil a nastavení — možnost úpravy uživatelského profilu a případné uživatelské nastavení aplikace.

Datové požadavky zahrnují zejména evidenci uživatelů systému a veškerá data potřebná pro plnění výše popsané funkcionality (perzistentní uložení záznamů v deníku, docházky,...), která nebudou detailně popisována. Kromě dat vzniklých v rámci používání aplikace je plánováno využití dat třetích stran (případně dat oddílu umístěných v rámci služeb třetích stran) — *České florbalové unie* a jejího informačního systému, klubových událostí dostupných pomocí služby *Google kalendář* a v budoucnu případně také data oddílu na jeho *Facebookové stránce*. Požadavky na použitelnost jsou popsané výše v rámci definice skupiny uživatelů.

Kapitola 4

Studium existujících řešení

Nápad na vytvoření aplikace usnadňující vedení sportovního oddílu (případně jiného týmu lidí) samozřejmě není naprostou novinkou. V současné době již existují různá volně dostupná řešení tohoto problému. Tato kapitola postupně popisuje některé vybrané aplikace sloužící tomuto účelu, analyzuje jejich silné a slabé stránky a především vhodnost jejich využití pro splnění cílů a požadavků popsaných v kapitole 3.2.

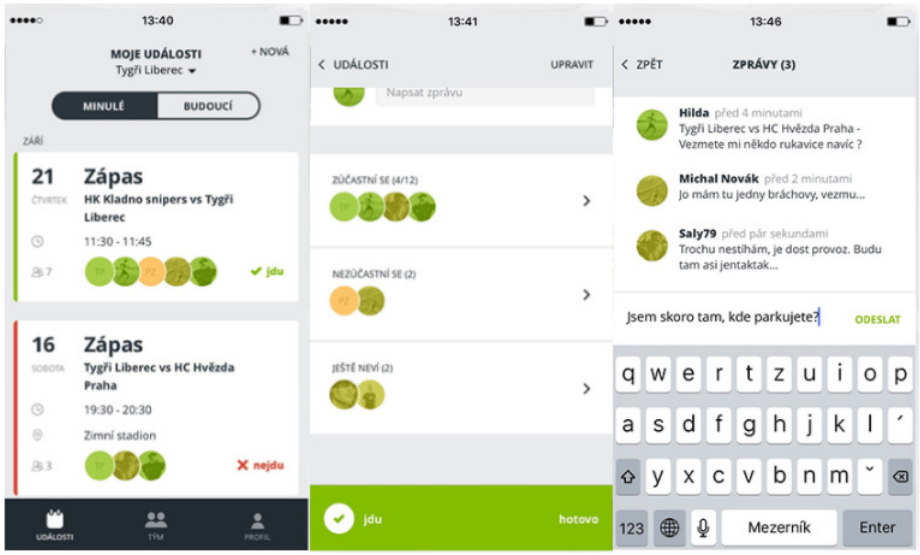
4.1 Týmuj

Týmuj[1] je česká služba určená právě pro vedení sportovních týmů (výběr sportu je dokonce povinnou součástí registrace nového týmu). Služba umožňuje vytvářet a spravovat jednotlivé sportovní týmy na úrovni konkrétních týmů, nikoliv oddílů. Hlavními funkcemi jsou správa soupisky týmu a informací o jeho členech, možnost vytváření událostí týmu (s možností oddělit soutěžní zápasy od ostatních událostí), vytváření úkolů a jejich přiřazení jednotlivým členům a také komunikace mezi členy týmu — a to jak přímo, tak také pomocí skupinové konverzace na úrovni celého týmu. Dále služba umožňuje sledovat docházku členů na jednotlivé události a zápasy (ovšem pouze pomocí vyplnění účasti samotnými hráči — chybí možnost úpravy docházky pověřenou osobou), zveřejňovat zprávy a spravovat galerii týmu. Je možné nastavit u některých částí aplikace veřejný přístup a tím umožnit libovolným uživatelům systému číst informace v dané části (využitelné například pro publikování informací). S docházkou je spojená možnost sledovat dostupnost jednotlivých členů týmu na budoucí termíny.

Aplikace splňuje několik kritérií pro využití v rámci florbalového oddílu popsaného v kapitole 3.2.3, zejména požadavky na správu členů a soupisku kategorií, sledování docházky (i když se značným omezením), komunikaci členů a možnosti zveřejňování zpráv. Naopak zde chybí mechanismy pro vedení popsaného klubového deníku, přístupu k výsledkům a dalším výkonnostním datům týmu, přihlašování na organizaci turnajů a další. Vzhledem k striktnímu rozdělení jednotek v aplikaci na jednotlivé týmy by také mohlo dojít ke ztížení práce osob ve vedení oddílu (administrátor, budoucí sekretáři, někteří trenéři), kvůli jejich zapojení ve velkém množství jednotlivých týmů. Další nevýhodou je nemožnost propojení s oddílovým webem, což je jeden z požadavků zadavatele.

Služba *Týmuj* funguje především jako webová aplikace. V současnosti je k dispozici i mobilní verze této služby (viz. obrázek 4.1) na platformách *Android* a *iOS*, ovšem prozatím disponuje pouze omezenou funkcionalitou, která by měla být v budoucnu rozšiřována.

[1] https://tymuj.cz/



Obrázek 4.1: Náhled aplikace *Týmuj* (převzato z webových stránek)

4.2 Team App

Team App[2] je služba, která umožňuje administrátorům sportovních týmů vygenerovat na základě dostupné šablony mobilní aplikaci určenou přímo pro jejich tým. Administrátor má možnost upravovat určité vzhledové prvky výchozí šablony takových způsobem, aby výsledný vzhled odpovídal jeho představám. Následně si vybere ty z připravených funkcí, které si přeje v aplikaci použít, a také může definovat různé skupiny lidí (tedy různé role v rámci oddílu), které budou mít k aplikaci přístup. Členové oddílu se poté podle jména mohou k oddílu přihlásit a zažádat i o přidání do určité skupiny.

Aplikace *Team App* poskytuje poměrné velké množství funkcí, ze kterých si může administrátor vybrat. Mezi základní funkce patří rozesílání novinek (s možností specifikace, kterým skupinám uživatelů jsou novinky určeny), správa konkrétních týmů a jejich soupisek, komunikace mezi členy i v rámci skupin členů, kontrola docházky a vytváření událostí. Dalšími funkcemi mohou být například zobrazování žebříčku či tabulek, nahrávání a sdílení dokumentů, vytváření anket pro členy, prodej klubových předmětů, možnost vložení internetové stránky a další.

Primárním účelem aplikace je její používání na mobilních zařízeních, ale k dispozici je i webové rozhraní umožňující plnohodnotnou práci všech členů. Z výše popsaných funkcí vyplývá, že při správném nastavení aplikace a vhodném využití nabízených funkcí by bylo možné vytvořit takovou aplikaci, která by pokryla většinu požadovaných funkcí pro cílovou aplikaci florbalového oddílu. Nevýhodou aplikace je ovšem skutečnost, že některé z funkcí jsou řešeny pomocí zobrazování webových stránek na základě vložených adres (například výsledky) nebo zobrazování vložených dokumentů ve formátu *PDF*, případně tabulek. Další nevýhodou je přítomnost reklam uvnitř aplikace, protože příjem ze zobrazování těchto reklam uživatelům je způsob příjmu firmy stojící za projektem *Team App*. Na zařízeních se systémem *iOS* je možné tyto reklamy za poplatek vypnout, na zařízeních s operačním systémem *Android* tato funkce zatím dostupná není.

[2] https://www.teamapp.com/

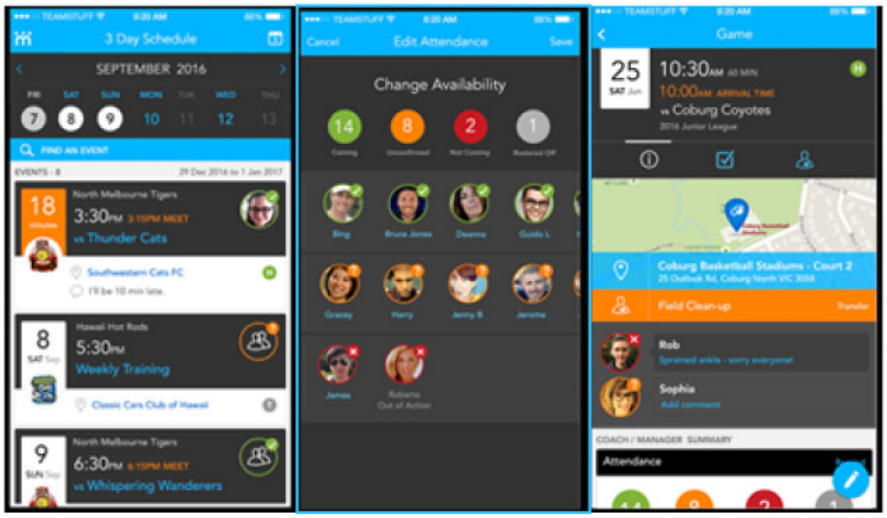
Zásadním problémem vzhledem k využití této služby je její nedostupnost v českém jazyce. Aplikace nicméně řeší mnoho problémů ve vedení sportovního oddílu a splňuje velkou část požadavků popsaných v kapitole 3.2.3, je tedy možné se některými jejími rysy inspirovat při výsledném řešení.

4.3 Teamstuff

Teamstuff[3] je další aplikací určenou pro vedení sportovních týmů (reálně využitelná i pro jiné druhy týmů, než jsou sportovní), která disponuje základními funkcemi jako je komunikace uvnitř týmu, dostupnost a docházka členů, sdílení dokumentů, správa termínů (tréninky, zápasy,...) včetně upozorňování, a další. Služba je dostupná skrze webové stránky a také jako mobilní aplikace.

Kromě těchto základních funkcí dostupných ve většině podobných aplikací obsahuje *Teamstuff* i další zajímavé funkce, jako například správu plateb uvnitř oddílu (včetně možnosti využít platební brány dostupné přímo v aplikaci), dále možnosti zobrazení dostupných akcí pro dané období na mapě a také možnost propojit případné členy rodiny uvnitř oddílu a zobrazovat termíny, kterých se jednotliví členové účastní.

Aplikace *Teamstuff* splňuje velkou část požadavků z kapitoly 3.2.3, přestože pro některé požadované funkce by bylo nutné vhodně využít dostupné možnosti a bylo by nutné dodržování stanovených pravidel — například pro vedení klubového deníku by bylo možné využít možnosti sdílení dokumentů, avšak bylo by nutné určit konkrétní formáty těchto dokumentů pro dané typy záznamů, které by všichni posléze dodržovali. Problémem by bylo plánované napojení na informační systémy *České florbalové unie* pro zobrazování výsledků a statistik a také registrace týmů, vyplňování soupisek a další úkony. Stejně tak spojení s oddílovým webem by při použití této aplikace nebylo možné. Ukázka podoby uživatelského rozhraní je na obrázku 4.2.



Obrázek 4.2: Náhled aplikace *Teamstuff* (převzato z webových stránek)

[3] https://teamstuff.com/

4.4 Další aplikace

K dispozici jsou také další (převážně mobilní) aplikace, jako jsou *Team snap*[4], *Teamer*[5] a další. Tyto aplikace sdílejí mnoho společných rysů — umožňují spravovat členy týmu, události týmu a docházku členů na tyto události, dále pak komunikaci členů, sdílení souborů a fotografií a další. Tyto aplikace však jsou shodně určeny pro použití v rámci konkrétního týmu, nikoliv na úrovni celých oddílů. Každá z těchto aplikací splňuje určitou část požadavků, ale žádná z nich se nepřibližuje výsledné představě natolik, aby bylo možné ji využít pro naplnění stanovených cílů.

[4] https://www.teamsnap.com/
[5] https://teamer.net/

Kapitola 5

Návrh postupu a částí řešení

Na základě studia popsaného v kapitole 2 a požadavků na systém v kapitole 3 je třeba navrhnout vhodné řešení popsaného problému, které bude následně realizováno v dalších částech této práce. Pro vývoj produktu jako celku bude využito prvků spirálového modelu a prvků agilního vývoje, které jsou spolu s dalšími modely životního cyklu softwaru popsané v kapitole 2.1.2. Vývoj bude probíhat iterativně se zapojením zadavatele a cílových uživatelů s cílem získat zpětnou vazbu k již vytvořenému produktu a upravení požadavků k dosažení co nejlepšího výsledku. Rozdělení práce a popis jednotlivých iterací je uveden v kapitole 5.1.

Z důvodů uvedených v kapitole 3.2.3 bude systém vytvořen jako klient-server architektura, kdy pro serverovou část bude využita již existující webová služba oddílu, která bude vhodným způsobem doplněna a rozšířena. Jako klientská část bude primárně sloužit mobilní aplikace, jejíž vývoj je hlavní náplní této práce. Kritickou částí celého systému je uživatelské rozhraní klientské části, jehož návrh je popsán v dalších částech této kapitoly.

5.1 Rozdělení iterací

Přestože se předpokládá zapojení cílových uživatelů ke konzultaci a testování jednotlivých inkrementálních přírůstků implementovaného řešení, bude implementace rozdělena do tří větších iterací, které sdružují postupný vývoj cílové funkcionality. Původní plán rozdělení práce počítá se třemi hlavními iteracemi, přičemž je možné (a pravděpodobné), že na základě budoucích úprav požadavků může být počet a rozsah iterací upraven podle aktuální potřeby.

V rámci první iterace vznikne zejména základní architektura, která bude umožňovat její následné rozšiřování. Verze systému na konci této iterace by měla obsahovat funkce pro použití osobami uvnitř klubu — konkrétně bude rozeznávat role trenéra a administrátora. Tato prvotní verze by poté autorizovaným uživatelům měla umožňovat komunikaci (vzájemnou i ve skupinách), možnost sledovat a filtrovat klubové novinky, přidávat záznamy do klubového deníku a spravovat docházku členů na všechny termíny. Součástí by měla být možnost administrátora spravovat vše, co je v této verzi dostupné — tedy správa uživatelů a jejich rolí, správa konverzací a přístupu k nim, správa deníku a dalších entit vyskytujících se v systému.

Ve druhé fázi vývoje by měla být doplněna role člena (tato role zaštiťuje zároveň role hráče a rodiče), který bude moci skrze aplikaci omlouvat svoji neúčast. Dále by měla být doplněna funkcionalita pro přihlašování členů na organizační záležitosti (pořádání domácích

Aktér	**1. iterace**	2. **iterace**	**3.iterace**
Neautorizovaný	Notifikace, Novinky		Výkonnostní data (výsledky, tabulky,. . .)
Autorizovaný	Komunikace		Profil, Nastavení
Člen	*XXX*	Docházka, Omluvenky, Povinnosti, Přihlašování na organizaci	
Organizátor	*XXX*	*XXX*	Manuál, Přihlašování na organizaci
Trenér	Docházka, Deník	Povinnosti, Termíny	Manuál, Organizace
Sekretář	*XXX*	*XXX*	Správa dat ve FIS, Další definované funkce
Administrátor	Správa uživatelů, deníku a konverzací, Zasílání notifikací	Správa povinností a organizace	Správa manuálu

Tabulka 5.1: Rozdělení případů užití mezi iterace

turnajů, výjezdy na venkovní turnaje) a také přehled a správa povinností jednotlivých uživatelů včetně vhodných notifikací, stejně jako veškeré potřebné administrátorské možnosti.

V rámci poslední iterace poté dojde k rozšíření funkcionality aplikace tak, aby pokryla i ostatní požadavky uvedené v kapitole 3.2.3, a také dodatečné požadavky, které se budou průběžně objevovat během úvodních iterací. Dále zde dojde k vyčlenění rolí organizátora a sekretáře (jejichž konkrétní práva a činnosti v rámci systému bude současně nutné dospecifikovat).

Pro jednotlivé iterace byly vytvořeny diagramy případů užití, které znázorňují rozdělení rolí v systému a zjednodušeně pokrývají předpokládanou funkcionalitu, která by měla být implementována. Tyto diagramy jsou umístěny v příloze A. Případy užití jsou v některých případech zjednodušeny a shrnuty (např. správa *FIS* dat) z důvodu zjednodušení celého diagramu. Rozdělení implementace funkcionality jednotlivých aktérů systému je také znázorněno pomocí tabulky 5.1. Tabulka koresponduje s hierarchií jednotlivých aktérů — veškeré funkce u neautorizovaného aktéra jsou dostupné i u všech ostatních aktérů, autorizovaný aktér sdružuje všechny aktéry kromě neautorizovaného, sekretář je nadřazený aktér trenérovi a administrátor je nadřazený aktér sekretáři. Buňky s obsahem *XXX* značí, že aktér v dané iteraci neexistuje.

5.2 Návrh podoby uživatelského rozhraní

Z důvodu několika plánovaných iterací při vývoji systému je třeba navrhnout uživatelské rozhraní takovým způsobem, aby bylo možné aplikaci dobře využívat v počáteční omezené verzi, a zároveň je třeba udržet možnost jednoduchého doplnění další funkcionality. Při návrhu jsem tedy již od počátku předpokládal budoucí doplnění popsaných předpokládaných funkcí a také možnost vyskytnutí se nových požadavků na další funkce. Pro návrh uživatelského rozhraní byly využity prvky vybraných metod popsaných v kapitole 2.3.

Nejdříve jsem na základě popsaných požadavků pro první verzi aplikace vytvořil úvodní návrh potřebných částí uživatelského rozhraní včetně několika konkurenčních návrhů některých částí (viz kapitola 2.3.2). Následně byly formou konzultace vybrány nejvhodnější z nich a zároveň byly zapracovány i další připomínky k jednotlivým částem rozhraní. Po vzájemné dohodě s vedením oddílu bude uživatelské rozhraní znovu iterativně vyhodnoceno a upraveno po vytvoření úvodní verze aplikace a jejím otestování.

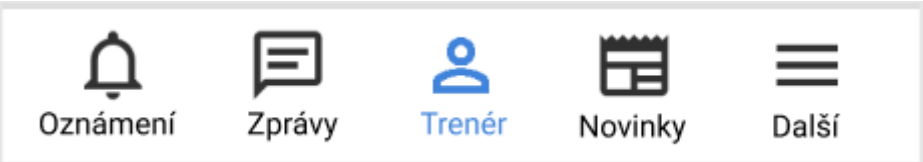
Následující sekce popisují hlavní části vytvořeného návrhu. Vytvořené návrhy se soustředili zejména na rozdělení aplikace na jednotlivé části a také vhodné rozmístění prvků uživatelského rozhraní za účelem dosažení dobré intuitivnosti a jednoduchosti používání. Na vizuální stránku aplikace nebyl v této fázi návrhu brán přílišný ohled a bude následně dořešena v pozdějších etapách vývoje aplikace.

5.2.1 Navigace

Zvoleným způsobem řešení hlavního menu aplikace je lišta záložek umístěná ve spodní části obrazovky (detailnější popis a vhodnost použití tohoto přístupu je popsaná v kapitole 2.5.3). Ideální počet záložek v liště je 3 nebo 5 — počet potřebných obrazovek v aplikaci však bude větší. Tato skutečnost by mohla vést k použití tzv. *Hamburger menu*, ale z hlediska rychlosti navigace a přehlednosti uvnitř aplikace se po konzultaci této možnosti s budoucími uživateli první popsaná varianta ukázala jako lepší.

Z důvodu většího množství obrazovek umístitelných přímo do hlavního menu než 5 je nutné přístup k některým z nich vyřešit jiným způsobem. Z tohoto důvodu bude jedna ze záložek použita jako volba „Další“ a tato záložka bude odkazovat na seznam méně využívaných a podpůrných funkcí, které se nepodaří umístit do ostatních obrazovek.

Zbývající 4 záložky je možné využít pro pohyb aplikací mezi nejdůležitějšími obrazovkami. Na základě analýzy požadavků a další komunikace s vedením klubu byly mezi tyto stránky vybrány stránka zobrazující notifikace, stránka komunikace uživatelů, stránka prezentující novinky z webové stránky oddílu, a nakonec stránka zpřístupňující funkce specifické pro danou roli konkrétního uživatele — tato stránka bude poskytovat různé funkce závislé na tom, zda je aktuální uživatel aplikace v roli člena, trenéra nebo administrátora (v budoucnu i další). Návrh navigace ukazuje obrázek 5.1.



Obrázek 5.1: Návrh navigace aplikace

Některé ze stránek budou potřebovat ještě další dílčí navigaci (například stránka „Další“), tyto dílčí navigace budou řešené pomocí vhodně zvolených dashboardů složených z dlaždic, které reprezentují jednotlivé možnosti na těchto stránkách.

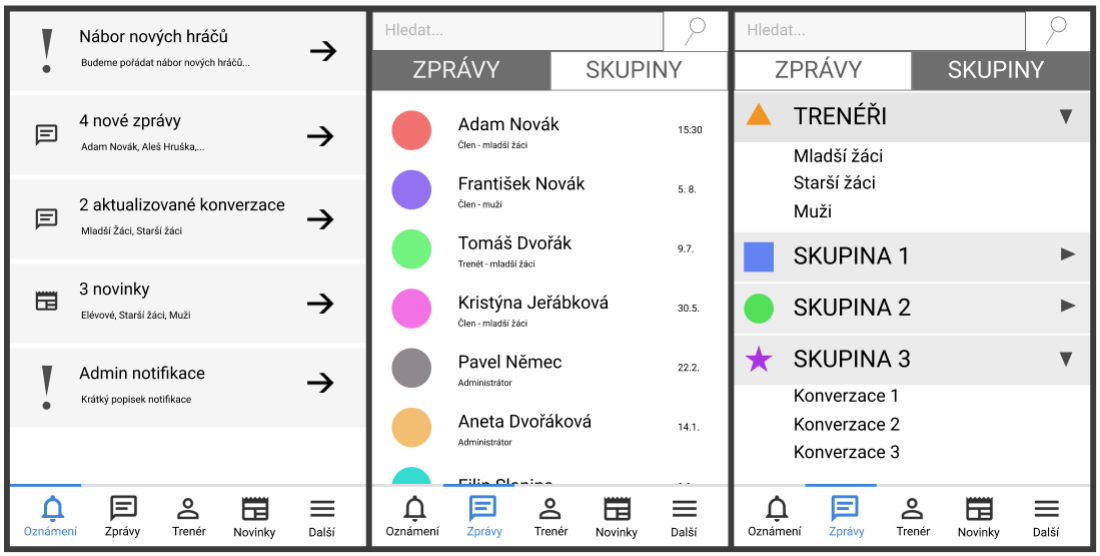
5.2.2 Hlavní stránky aplikace

Jednotlivé hlavní stránky aplikace odpovídají položkám menu zvoleným a popsaným v kapitole 5.2.1. Těchto hlavních stránek je tedy 5 a každá z nich je určitým způsobem unikátní.

Stránka notifikací bude podle požadavků vedení oddílu sdružovat a zobrazovat upozornění pro každého uživatele. Každé upozornění se projeví jednak jako klasická *push notifikace* typická pro mobilní zařízení a následně se objeví v tomto seznamu. Notifikace budou uživatelé dostávat například v těchto případech:

- nová příchozí zpráva od jiného uživatele
- nový příspěvek ve skupinové konverzaci, které se uživatel účastní
- novinka nebo zpráva publikovaná prostřednictvím webových stránek klubu
- blížící se událost nebo termín pro splnění určité povinnosti
- speciální upozornění odesílané administrátorem
- upozornění pro administrátora na nějakou situaci (záznam v deníku, nový uživatel apod.)

V případě více upozornění stejného typu nebude v určitých případech zobrazované každé zvlášť, ale budou sdružené do jedné položky v tomto seznamu, která bude kromě popisu obsahovat také množství daného typu upozornění — například „5 nových zpráv“. Každý typ upozornění povede po stisknutí uživatele do odpovídající části aplikace, případně bude odkazovat do jiné aplikace (typicky webový prohlížeč pro externí odkazy nebo například výchozí aplikace pro navigaci v případě odkazu na mapu atd.).



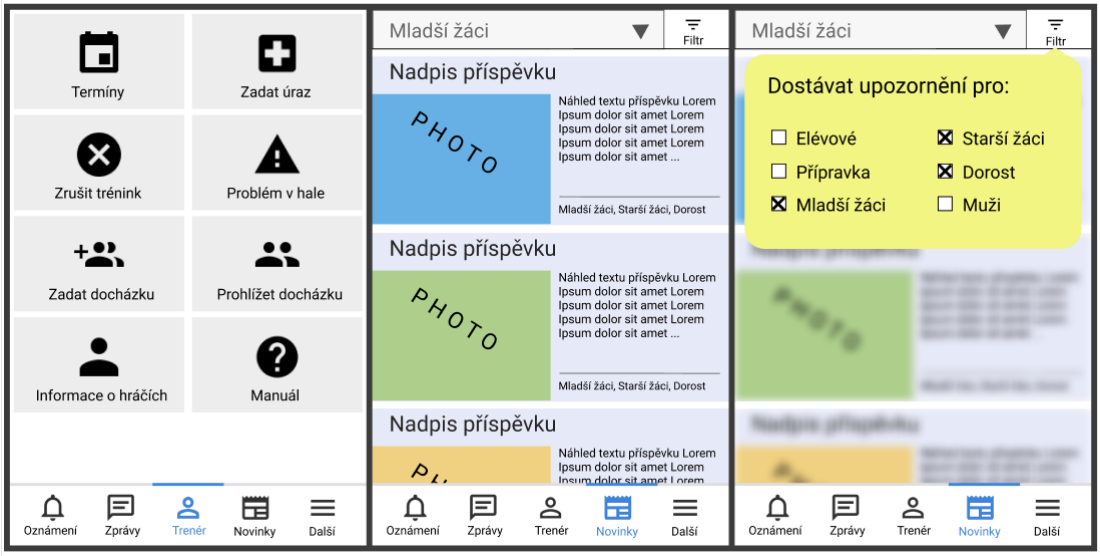
Obrázek 5.2: Návrh stránek upozornění a komunikace

Stránku se zprávami je nutné rozdělit na dvě části — na přímé zprávy a na skupinové konverzace. Přepínání mezi těmito sekcemi bude řešeno pomocí přepínacího tlačítka v horní části obrazovky. Sekce s přímými zprávami bude vizuálně odpovídat podobě jiných komunikačních aplikací. K dispozici bude seznam uživatelů zobrazující jejich jméno a případnou fotografii a po otevření některého z kontaktů potom chronologický seznam zpráv s grafickým odlišením přijatých a odeslaných. Uživatelé budou mít možnost kromě textových zpráv

posílat také soubory a jednoduché emotikony. Očekávaným způsobem bude vyřešená také sekce skupinových konverzací — seznam dostupných konverzací s dvouúrovňovou hierarchií (s možností sbalení a rozbalení na nejvyšší úrovni) a v detailu konverzace opět chronologický seznam zpráv s indikací odesílatele u každé z nich. Stránky v části komunikace jsou spolu se stránkou notifikací zobrazeny na obrázku 5.2.

Stránka s uživatelskými funkcemi bude zobrazovat dlaždice reprezentující jednotlivé možnosti, které uživatel má, pomocí vhodně zvolených ikon a krátkého popisku. Každá dlaždice povede uživatele na další stránky, které poté umožňují vykonání požadované akce. Tento způsob reprezentace konkrétních činností by měl zvýšit přehlednost a intuitivnost používání aplikace, přestože může vést k mírnému prodloužení vykonání některých případů užití. Další výhodou tohoto přístupu je také velmi dobrá rozšiřitelnost, kdy pro přidání nové funkcionality postačí přidat novou dlaždici, která zpřístupní nově požadovanou akci. Druhou možností vytvoření této stránky bylo umístění možností do vertikálního seznamu. Forma dlaždic byla nakonec zvolena jako lepší z obou možností. Konkrétní činnosti prováděné v této sekci aplikace jsou popsány v kapitole 5.2.3. Ukázka stránky s funkcemi trenéra je na obrázku 5.3.

Stránka novinek má za cíl prezentovat uživatelům informace publikované prostřednictvím oddílového webu. Informace na webu jsou rozdělené do kategorií odpovídajících jednotlivým věkovým kategoriím v rámci oddílu. Mezi těmito kategoriemi bude možné přepínat pomocí výběrového seznamu (tzv. *combo-box*) a uživatel bude také mít možnost vybrat si ty kategorie, pro které si přeje dostávat upozornění. Samotné příspěvky budou zobrazené ve formě seznamu a po vybrání konkrétní položky se otevře odpovídající stránka na klubovém webu. V pozdějších iteracích bude tento způsob zobrazení nahrazen zobrazením přímo v aplikaci. Návrh podoby této stránky je spolu s trenérskými funkcemi na obrázku 5.3.



Obrázek 5.3: Návrh stránky s funkcemi trenéra a stránky klubových novinek

Stránka *Další* bude vytvořena podobným způsobem jako stránka s uživatelskými funkcemi — bude tedy obsahovat dlaždice, které povedou uživatele na stránky reprezentující jednotlivé konkrétní činnosti (nastavení aplikace, správa profilu,...).

5.2.3 Stránky uživatelských funkcí

V rámci první iterace je nutné vytvořit části uživatelského rozhraní pro splnění základních případů užití spojených s činností trenérů a administrátora. Stránky pro případy užití ostatních rolí budou navrženy a vytvořeny v pozdější fázi s využitím zpětné vazby od uživatelů první verze aplikace.

Prvním z případů užití trenérů je přidávání záznamů do klubového deníku. Jelikož první verze aplikace nebude mít v této části aplikace příliš velký počet dlaždic (tedy funkcí), bude pro každý z různých typů záznamů vytvořena dlaždice přímo na stránce uživatelských funkcí. Pokud v budoucnu dojde ke zvýšení počtu dlaždic (zejména pokud velká část z nich bude spojena s deníkem), budou všechny typy záznamů nahrazeny jednou dlaždicí, která bude reprezentovat právě tento případ užití. Výběr konkrétního typu záznamu bude poté vyčleněn na samostatnou obrazovku, která se tak stane jakýmsi mezičlánkem mezi hlavní stránkou a stránkou vkládání konkrétního záznamu. Samotná data budou od uživatele získána za pomocí formulářových prvků typických pro mobilní zařízení s využitím principů popsaných v kapitole 2.5.2 — předvyplnění data, umožnění výběru ze seznamu namísto zadávání textem a podobně.

Dalším důležitým případem užití je zadávání docházky hráčů. V tomto případě je vhodné rozdělení na dva samostatné kroky — specifikace termínu a samotné vyplnění účasti. V první části trenér vybere, zda se jedná o trénink či zápas (případně jinou aktivitu), specifikuje kategorii a zadá datum termínu. V tomto kroku bude ve většině případů možné automaticky vyplnit všechna potřebná data na základě aktuálního času a termínů tréninků a zápasů jednotlivých kategorií. Může ovšem dojít k situaci, že toto nebude možné (trenér zadává docházku do systému dodatečně, tréninku se účastní více než jedna kategorie, a podobně), je tedy nutné umožnit trenérům tyto údaje v případě potřeby upravit. Na základě vybrané kategorie se zobrazí seznam hráčů v této kategorii a u každého z nich bude možné vybrat mezi třemi možnostmi — *zúčastnil se*/*omluven*/*neomluven*. Pro indikaci některé ze tří možností budou využity barevné ikony, které by měly urychlit výběr správné možnosti — zvolené ikony pro možnosti *omluven* a *neomluven* odpovídají způsobu zapisování výše uvedených možností v aktuálně používaných papírových docházkových listech. Ikona pro možnost *zúčastnil se* byla vybrána jako vhodnější oproti dosud používané pomlčce. Výběr barev proběhl s ohledem na asociaci, kterou tyto barvy běžně vyvolávají — zelená barva je běžně užívána pro zvýraznění správné možnosti (v tomto kontextu je ideální možností účast hráče) a červená barva naopak pro zvýraznění možnosti špatné (zde se jedná o situaci, kdy se hráč nezúčastní a neomluví se). Správnou asociací barev při používání uživatelského rozhraní lze dosáhnout přirozenější práce s tímto rozhraním (viz kapitola 2.2.1). Vyplnění docházky by tedy pro trenéry mělo být dostatečně intuitivní. Navržené ikony jsou zobrazené na obrázku 5.4.

Obrázek 5.4: Ikony pro vyplnění docházky

Z hlediska uživatelské přívětivosti je žádoucí, aby u co největšího počtu hráčů byla automaticky vyplněna správná hodnota. V první verzi aplikace bude u všech hráčů předvyplněna možnost *zúčastnil se* — důvodem je fakt, že ve většině případů je skupina účastnících

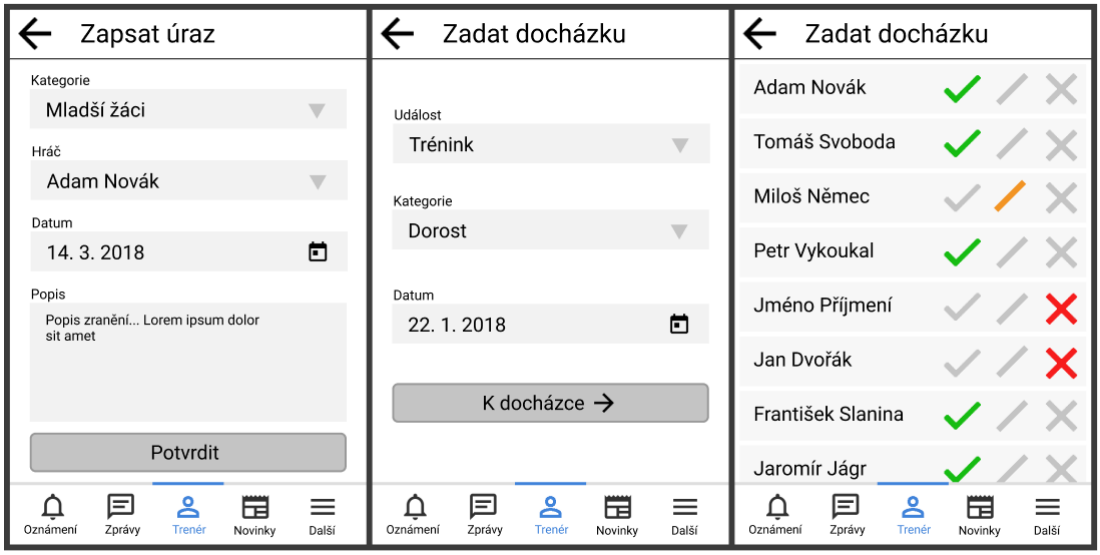
se hráčů největší. U omluvených a neomluvených hráčů bude následně muset trenér tuto skutečnost zadat. V následných verzích aplikace dojde k přidání funkce omlouvání hráčů, počet úprav trenérem tak v budoucnu ještě klesne.

S docházkou je také spojená nutnost jejího zobrazení. Předpokládané způsoby zobrazení jsou:

- zobrazení účasti hráčů na konkrétním termínu formou seznamu hráčů s vyznačením jejich účasti
- zobrazení účasti konkrétního hráče na termínech jeho kategorie formou seznamu termínů s vyznačením účasti hráče na každém z nich
- zobrazení docházky dané kategorie formou tabulky

Tabulka docházky bude na mobilním zařízení poměrně nepřehledná, protože může dosahovat velkých rozměrů (počet hráčů přibližně v rozmezí 10-20, počet termínů přibližně 25-35). Přesto může taková tabulka být pro trenéra v určitých případech hodnotná, bude tedy do aplikace zahrnuta. Tabulka bude skrolovatelná horizontálně i vertikálně a jednotlivá políčka budou vyplněna co nejúspornějším způsobem s důrazem na minimalizaci rozsahu tabulky. V první verzi aplikace bude tabulka obsahovat možnost filtrace pouze těch řádků a sloupců, která obsahují pole vyplněné *neomluven*. Na základě zpětné vazby od uživatelů budou případné další možnosti filtrace tabulky doplněny následně.

V budoucí verzi aplikace přibude možnost omluvení hráče z termínu. Pro tento účel postačí vytvoření jednoduchého formuláře, ve kterém hráč vybere konkrétní termín, ze kterého se chce omluvit, doplní důvod a odešle. Aplikace také umožní každému z hráčů zpětně sledovat jeho docházku podobným způsobem jako trenérovi.



Obrázek 5.5: Návrh vybraného formuláře deníku a návrh zadání docházky

Administrátorské funkce zahrnují v první verzi především zobrazování záznamů v deníku (rozdělené podle druhů a zobrazené podobnou formou, jako jsou vyplňovány ve formuláři) a správu uživatelů a jejich rolí. Administrátor má možnost zobrazit seznam uživatelů a

u každého z nich vidí jejich přiřazené role, které může manuálně měnit. Ukázka některých stránek s uživatelskými funkcemi je na obrázku 5.5.

5.2.4 Uživatelé s více rolemi

Uživatelské rozhraní aplikace musí být schopné reagovat na situaci, kdy má uživatel v rámci systému více rolí. Pokud se jedná o kumulaci rolí v rámci vnitřní struktury oddílu (trenér, sekretář, administrátor), tak není třeba žádných zvláštních úprav — sekretář je rozšířením role trenér a administrátor je rozšířením role sekretář. Takový uživatel tedy uvidí na uživatelské stránce všechny funkce, které mají dostupné všechny jeho role.

Jiná situace nastává, pokud je některý z uživatelů trenérem (případně sekretářem nebo administrátorem) a zároveň také řadovým členem. V takovém případě je nutné oddělit funkcionalitu jednotlivých rolí. Pro tyto účely je možné využít podobného přepínacího tlačítka v horní části obrazovky, jaké je k dispozici na stránce komunikace — uživatel tak bude moci rychle a pohodlně přecházet mezi funkcemi jednotlivých rolí.

5.3 Použité nástroje

Součástí návrhu uživatelského rozhraní je vytvoření referenční vizualizace jeho nejdůležitějších částí. Pro tyto potřeby jsem využil nástroje *Figma*[1], jehož používání je jednoduché a přímočaré, a který umožnil efektivní vytvoření požadovaných zjednodušených návrhů. Vzhledem k důrazu návrhu na rozdělení aplikace do částí a rozvržení prvků uživatelského rozhraní na obrazovce, nikoliv na konkrétní podobu všech součástí, je tento nástroj dobrou volbou právě pro svoji jednoduchost, intuitivnost a přímočarost používání.

Pro samotnou implementaci mobilní aplikace je nutné zvolit mezi použitím nástrojů určených pro vývoj aplikace na zvolené platformě a nástrojů pro vývoj aplikace multiplatformním způsobem. Primárně bude sice aplikace vyvíjena pro platformu *Android*, nelze však vyloučit nutnost rozšíření i na další platformy mobilních zařízení. Z toho důvodu se nabízí využít možnosti vytvoření aplikace multiplatformní cestou — konkrétně s využitím frameworku *Xamarin.Forms*.

Xamarin.Forms umožňuje vývoj aplikací pro platformy *Android*, *iOS* a *Universal Windows platform* s možností sdílení velké části kódu mezi všemi z nich. Největší výhoda tohoto nástroje je tedy zřejmá — vytvoření aplikace pro obě hlavní mobilní platformy (a také právě *UWP*) je díky sdílení kódu mnohem rychlejší než vytváření tří různých aplikací pro každou platformu zvlášť. Vytvořená aplikace může být taktéž provozována na desktopovém počítači se systémem *Windows*, což je skutečnost, která může v případě potřeby velmi efektivně vyřešit možný požadavek na administrátorské rozhraní aplikace.

Samotný framework nabízí několik připravených stránek a komponent, které je možné využít pro vytvoření uživatelského rozhraní, a které mají na každé platformě vzhledové prvky typické pro danou platformu. Všechny komponenty je však možné upravovat dle libosti vývojáře a stejně tak je samozřejmě možné vytvářet nové, vlastní, komponenty. Pro každou platformu je také možné upravit vzhled samostatně, při výrazném aplikování této možnosti však dochází k omezení výhod *Xamarin.Forms*. [6, 17]

[1] https://www.figma.com/

Kapitola 6

Úvodní fáze vývoje

Po získání a zpracování požadavků na nově vznikající systém (kapitola 3.2) a následném navržení základních rysů budoucí realizace tohoto systému (kapitola 5) nadešel čas začít vytvořenou představu realizovat.

Prvním úkolem bylo přetavit návrh uživatelského rozhraní klientské části z původně vytvořené názorné podoby do reálné a spustitelné mobilní aplikace. Jelikož původní návrh se soustředil především na rozdělení aplikace do menších logických celků a dále na rozmístění funkčních prvků v konkrétních částech aplikace, nikoliv na výslednou vizuální podobu uživatelského rozhraní, bylo nutné podrobnosti vyřešit v tuto chvíli. Po vyjasnění požadované podoby aplikace již je možné začít se skutečnou implementací jednotlivých částí určených k realizaci v první iteraci (viz kapitola 5.1).

6.1 Základní prvky uživatelského rozhraní

Hlavní otázkou se přirozeně stala otázka výběru barev. Hlavní barvy oddílu, použité například v logu oddílu nebo existujících materiálech (plakáty, dresy,...), jsou žlutá a černá. Z těchto dvou barev tedy také vychází základní barvy tvořící samotné uživatelské rozhraní — černá barva s tmavším odstínem žluté jako hlavní barvy a světlejší odstín žluté jakožto barva doplňková. Pomocí těchto zvolených odstínů budou vytvořeny všechny části uživatelského rozhraní, kromě případů, kdy bude vhodné nějaký konkrétní prvek významně odlišit z důvodu zvýraznění pro uživatele (dle principů popsaných v kapitole 2.2.3).

Dalším důležitým faktorem byla volba pozadí. Z počátku bylo pozadí ponecháno zcela jako čistě bílé, brzy však ze strany uživatelů vzešel návrh na využití podoby barevných „fleků“, které oddíl používá jako barevné pozadí na svých tištěných materiálech. Po vizuální úpravě a zejména značném zesvětlení tohoto vzoru bylo toto pozadí skutečně zakomponováno a kladně hodnoceno ze strany uživatelů. Brzy se však ukázalo, že na některých specifických stránkách aplikace toto pozadí působí příliš rušivě (např. na stránce konverzace) — výsledkem je kombinované použití, kdy na většině stránek je využito popsané pozadí, pouze na několika vybraných zůstalo původní čistě bílé pozadí. Ukázka využití barevného pozadí je např. na obrázcích 6.2 a 6.5, bílé pozadí je možné vidět např. na obrázcích 6.1 a 6.6.

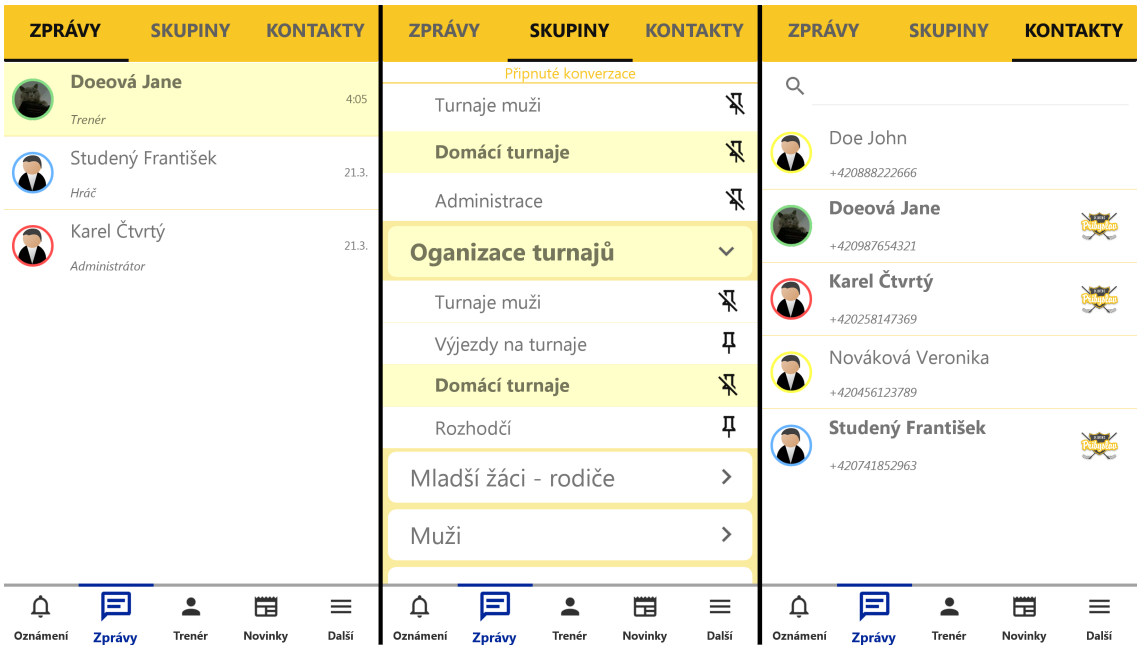
Posledním ze základních kamenů pak bylo hlavní menu aplikace. Každá z položek v hlavním menu je vytvořena kombinací ikony a krátkého popisku (viz kapitola 5.2.1). Menu vzniklo ve dvou rozdílných podobách. Jedna z verzí využívala výše popsané základní barvy — neaktivní položka měla světlý (doplňkový) odstín žluté, zatímco aktivní položka byla

zobrazena pomocí tmavší (hlavní) žluté barvy. Druhá z verzí menu vycházela z původního návrhu v kapitole 5.2.1 a využila barvy mimo vybranou sadu — šedou pro neaktivní a modrou pro aktivní položku. Po konzultaci a vyzkoušení dalších variant (například kombinace žluté a černé) se jako nejlepší ukázala cesta barev z původního návrhu, které výrazně odlišily hlavní menu od zbytku obrazovky, což uživatelé vnímali pozitivně.

6.2 Sekce komunikace uživatelů

Umožnění vzájemné komunikace uživatelů v rámci aplikace byl jedním z hlavních cílů od samého začátku jejího vzniku (viz kapitola 3.2). Návrh této části (popsaný v kapitole 5.2.2) byl inspirovaný jinými běžně používanými aplikacemi s tímto zaměřením, které sdílí velmi podobnou strukturu uživatelského rozhraní. Na tuto strukturu jsou běžní uživatelé zvyklí a během implementace nevyvstal žádný důvod se od ní odklonit.

Během konzultací vytvářeného rozhraní s uživateli docházelo pouze k drobným vizuálním úpravám. Jediným větším zásahem do původního návrhu byl požadavek na doplnění stránky s kontakty. Uživatelé jsou z jiných komunikačních aplikací zvyklí na přítomnost samostatné stránky, která obsahuje seznam všech jejich uložených kontaktů včetně indikace, zda daný kontakt je či není uživatelem stejné aplikace. Původní dvě stránky komunikační sekce tedy byly rozšířeny o navrženou třetí stránku s kontakty, jejíž ukázka je na obrázku 6.1. Součástí obrázku je také výsledná podoba přepínacího panelu, který byl navržen v kapitole 5.2.2, a jehož podoba prošla během konzultací několika změnami, až se ustálila na podobě, kterou popisuje *material design* [5].



Obrázek 6.1: Podoba jednotlivých stránek komunikační sekce včetně přepínacího panelu v horní části.

Kladnou odezvu uživatelů sklidilo také barevné orámování fotek uživatelů v celé komunikační sekci. Každá z uživatelských rolí je vyjádřena odlišnou barvou, což zlepšuje orientaci mezi jednotlivými osobami, se kterými uživatel komunikuje.

Během uživatelského testování v pozdějších fázích vývoje se objevil požadavek na možnost „připíchnutí“ konkrétních konverzací na pevné místo na vrcholu seznamu konverzací. Tato funkce byla implementována s využitím tlačítka doplněného ke konverzacím a následně uživateli velmi kladně přijata. Podobu připínacího tlačítka je také možno vidět na obrázku 6.1.

S komunikací souvisí i část aplikace určená pro šíření novinek publikovaných vedením na webu oddílu směrem k ostatním. Oproti původnímu návrhu v kapitole 5.2.2 bylo ve výsledné podobě upuštěno od zobrazení fotografie. Hlavním důvodem byla skutečnost, že ve většině případů nemá použitá fotografie informační hodnotu, je tedy zbytečné, aby na malé obrazovce mobilního telefonu zaujímala značnou část prostoru. Vynechání fotografie má také pozitivní vliv na dobu načítání novinek. Ostatní navržené prvky v této části zůstali během konzultací při implementaci zachovány a došlo pouze k jejich vizuální úpravě.

6.3 Formulářové elementy a deník

Vedení klubového deníku je v kapitole 3.2 uvedeno jako jedna ze základních funkcionalit budoucí aplikace, chybí zde však přesnější popis konkrétních informací, které mají být v deníku obsaženy. Po dodatečné komunikaci se zástupci oddílu a analýze současného deníku byly definovány tři hlavní typy záznamů — `hala` (problémy spojené s aktivitou ve sportovní hale, např. poškození vybavení), `úraz` (úrazy během všech akcí oddílu) a `organizace` (všechny ostatní organizační problémy, které nespadají do předchozích kategorií). Každý ze záznamů by měl obsahovat několik různých povinných či nepovinných položek, které zde nebudou přesně definovány.

6.3.1 Realizace deníku

V kapitole 5.2.3 byl problém zadávání záznamů do deníku vyřešen navržením formuláře pro vyplnění potřebných informací. Jenom pro potřeby deníku bude nutné vytvořit tři částečně odlišné formuláře. Navíc je předem jisté, že formulářových stránek bude v aplikaci více (příkladem může být vytváření omluvenky). Je proto výhodné nalézt typické prvky společné všem formulářovým stránkám a pokusit se je abstrahovat, aby následná implementace konkrétních formulářových stránek probíhala co nejpřímočařeji.

Druhou součástí deníku (opět nejenom deníku, ale i různých dalších částí aplikace) bude možnost procházení vložených záznamů a jejich případná úprava. Při procházení záznamů je nutné vyřešit dvě situace — nejprve zobrazení seznamu všech záznamů (který by měl umožnit jejich filtrování) a následné otevření konkrétního zvoleného záznamu, který bude možné v některých případech také upravovat. I v tomto případě je vhodné vytvořit takové řešení, které bude možné později univerzálně použít i v jiných případech, než je samotný deník.

Realizace deníku se tak stala vhodnou příležitostí pro vytvoření sady vizuálních elementů (anglicky zvané *controls*, jedná se o znovupoužitelné vizuální prvky) pro zadávání (kapitola 6.3.2) i zobrazování (kapitola 6.3.4) dat s využitím principů popsaných v kapitole 2.4.4. Tyto elementy budou uplatněny i v mnoha jiných částech aplikace — přehled a

správa kategorií, manuálů, checklistů[1], omluvenek a dalších. Bylo by zbytečné opakovaně popisovat a zdůvodňovat jejich využití v podobných situacích, přestože v mnoha případech tvoří jádro řešení. V následujících kapitolách již tedy nebude explicitně popisováno řešení všech částí aplikace, které jsou tvořeny právě níže popsanými univerzálními prvky, kromě případů, kdy došlo k jejich dalšímu uzpůsobení konkrétní situaci.

6.3.2 Formuláře v aplikaci

Každý formulář bude potřebovat zejména prvky pro zadání dat — nejčastěji text, datum a výběr z více možností. Je nutné však počítat také s ojedinělou a specifickou položkou v některém z formulářů. Pro tyto prvky je možné vytvořit znovupoužitelné elementy. Každé pole ve formuláři bude mít svůj popisek a také indikaci, zda je pole povinné (viz kapitola 2.4.4).

Základ tvoří generický element (`FormGenericControl.xaml`), který řeší zobrazení všech popsaných náležitostí kromě samotné hodnoty. S využitím tohoto generického základu následně vznikly další elementy pro nejobvyklejší situace — např. zadání krátkého textu (`FormEntryControl.xaml`) či data (`FormDateControl.xaml`). Pro specifické případy je možné tento element využít pouhou implementací samotného obsahu, zatímco ostatní části již jsou vyřešené generickou implementací. Ukázky vzhledu jednotlivých elementů je možné vidět například na obrázcích 6.2 či 6.5.

Pro docílení konzistentního vzhledu všech formulářů byly využity ještě dva prvky. Jedním z nich je potvrzovací tlačítko (`FormButtonControl.xaml`) a druhým obálka celého formuláře, pro univerzální odsazení (`FormWrapperControl.xaml`). Využitím všech těchto univerzálních formulářových elementů tedy docílíme kromě přímočarosti vytváření stránek také skutečnosti, že všechny formuláře v aplikaci budou mít stejnou vizuální podobu. Tento efekt by měl přispět k pozitivnímu přijetí uživatelského rozhraní uživateli, jak udává studium z kapitoly 2.2.

Problematika formulářů však nekončí pouhým zaručením konzistentního vzhledu. Kapitola 2.5.2 nastiňuje i další principy, které je nutné vzít v úvahu při jejich vytváření. Jedním z nejdůležitějších je předvyplnění dat do položek. U všech formulářů v aplikaci je na vhodných místech předvyplnění využito. Typickým případem je vyplnění data aktuálního dne (např. u deníku) nebo předvýběr nejpravděpodobnější z více možností (např. `Důvod` u omluvenky nebo `Událost` u haly v deníku apod.). Za zmínku stojí například také předvyplnění uživatelů přiřazených k checklistu — na základě analýzy lidí typicky zodpovědných za události, pro které jsou checklisty určeny, vyšlo najevo, že často se jedná o trenéry kategorie, která se události účastní. Při vytváření nových checklistů jsou tedy tito automaticky předvybráni.

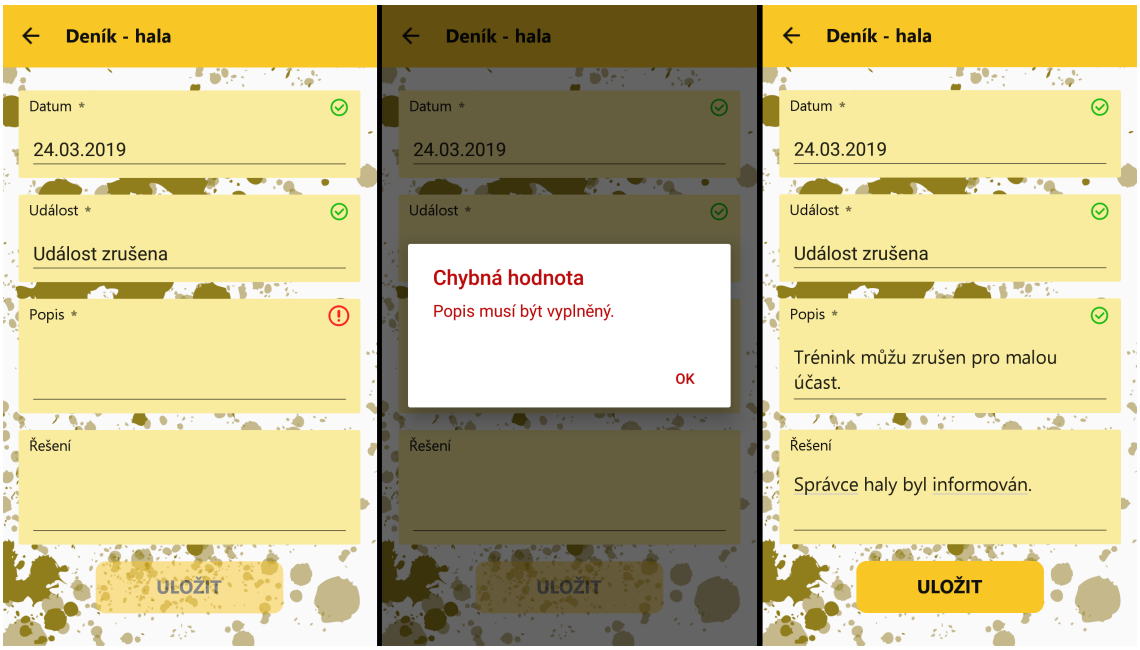
V určitých situacích se však tato praktika ukázalo jako nevhodná. V případě výběru z více možností, kdy jsou však data načítána dynamicky (typicky výběr kategorie, hráče nebo události), se ukázalo, že předvýběrem některé z možností (většinou první možné) se zvýšila pravděpodobnost chyby ze strany uživatele. Tento typ předvýběru byl tedy následně odebrán, s výjimkou situací, kdy je možná pouze jedna volba — v tom případě je automaticky zvolena.

Pro výrazné zlepšení použitelnosti formulářových částí aplikace je možné využít různých typů vestavěných klávesnic v mobilních zařízeních. Operační systémy zpravidla nabízí různé klávesnice pro zapisování textu, čísel, telefonních čísel, emailových adres apod. *Xa-*

[1] přestože slovo *checklist* — anglický výraz označující soubor „odškrtávacích“ položek — není součástí spisovné češtiny, z důvodu absence adekvátního českého výrazu bude v této práci používáno se skloňováním dle vzoru *hrad* mužského rodu

marin forms nabízí u textových polí možnost výběru z těchto různých klávesnic. Jedná se o drobnou změnu při použití stejného elementu v různých případech, tato změna však může velice zrychlit a zpříjemnit vyplňování formuláře. V implementované aplikaci je tato možnost plně využívána na všech místech vkládání textových dat — tedy ve výše popsaných formulářových elementech i v dále zmíněných samostatných úpravných dialozích).

Další důležitou oblastí je validace vložených informací. V případě povinných polí, či polí se specifickou strukturou požadovaných dat, bude nutné jednoznačně zobrazovat případné chyby při vyplňování (viz kapitola 2.2.3). V případě nevyplnění všech potřebných dat uživatel nemůže potvrdit formulář — tlačítko pro potvrzení vizuálně napovídá, že není možné pokračovat.



Obrázek 6.2: Ukázka rozdílu mezi nevalidním (levý) formulářem a validním (pravý) formulářem spolu s ukázkou detailního chybového hlášení (uprostřed), které je možné vyvolat klepnutím na indikátor chyby — červený vykřičník v levém obrázku.

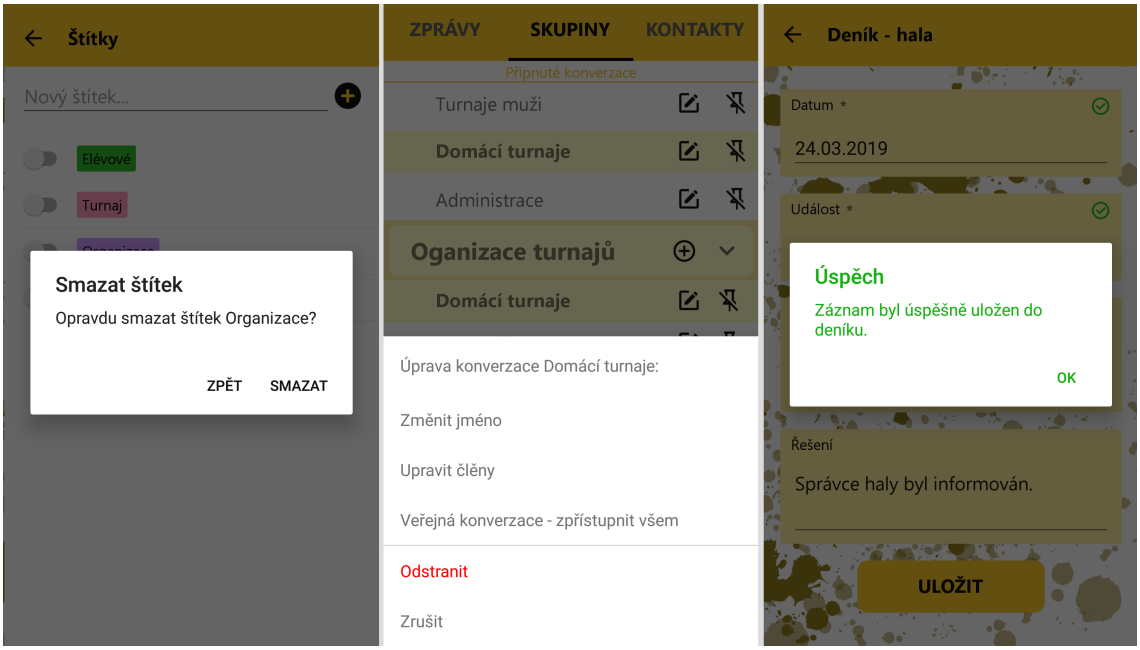
V případě vyplněných, avšak nevalidních dat, může být uživatel upozorněn dvojím způsobem — buď chybovým hlášením s popisem problému nebo grafickým zvýrazněním nevalidního pole. Testováním obou možností byla nakonec zvolena varianta grafického zvýraznění nevalidního pole. Hlášení s konkrétním popisem chyby může uživatel zobrazit stisknutím ikony indikující chybu. Během testování se také ukázala užitečnost vizuálního odlišení chybových hlášení od ostatních dialogových oken, jejich barva byla proto upravena na červenou. Ukázka validního i nevalidního formuláře (včetně ručně vyvolaného hlášení o detailu chyby) je na obrázku 6.2.

6.3.3 Dialogová okna

Výše zmíněné zobrazení chybového hlášení je jedním z příkladů využití různých dialogových oken (implementovaných s využitím knihovny *ACR User Dialogs*[2]) v rámci celé aplikace.

[2] https://github.com/aritchie/userdialogs

Stejný princip využívají i okna, která informují uživatele o dokončení nějaké operace (viz kapitola 2.2.3) — např. při úspěšném vložení záznamu do deníku (ukázka na obrázku 6.3). Dialogové okno je také prostředkem pro potvrzení úmyslu provést požadovanou akci, kdy má uživatel možnost akci potvrdit nebo naopak odvrátit. Toto potvrzování je výhodné pro operace se závažnými nebo nevratnými důsledky, jako například přidělení rolí uživatelům nebo smazání určitého objektu (ukázka taktéž na obrázku 6.3).



Obrázek 6.3: Ukázka využití dialogových oken v aplikaci — požadavek na potvrzení záměru provést akci (vlevo), výběr dostupných akcí (uprostřed) a oznámení o úspěšné akci (vpravo).

Dalším příkladem použitých dialogů je okno určené k získání textového vstupu. Tento dialog může sloužit jako zajímavá alternativa k základní podobě textových polí popsaných výše. Jeho využití je vhodné zejména v případech, kdy je od uživatele vyžadováno zadání pouze jediné hodnoty. Tato situace nastává ve dvou případech — při vytváření objektu, kde pouze jedna hodnota vyžaduje uživatelský vstup (např. nová skupinová konverzace), a také při úpravě jedné konkrétní existující hodnoty (např. změna telefonní čísla uživatele v detailu uživatelů aplikace). Zobrazení dialogového okna a zadání hodnoty přímo do něj se ukázalo v těchto situacích rychlejší a příjemnější pro uživatele než otevírání samostatné stránky s formulářem o jediném poli.

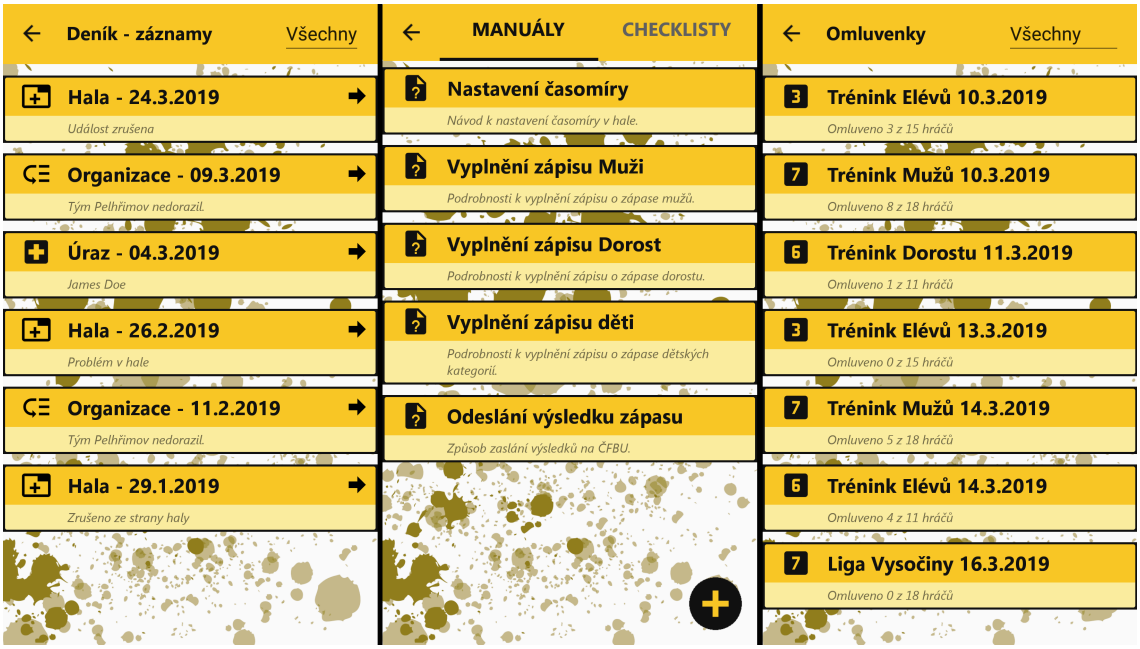
Získat od uživatele vstup tímto způsobem by samozřejmě bylo možné i v případě komplexnějších vstupů, a to například zřetězením několika po sobě jdoucích dialogů. Tento přístup však má dva zjevné problémy — neumožňuje vyplnit data v libovolném pořadí, přestože není důvod takovému přístupu bránit, a neumožňuje uživateli vizuálně udržovat kontext mezi všemi vloženými informacemi. Experimentálně se oba tyto problémy ukázaly být reálnou překážkou v pohodlnosti používání aplikace a využití klasické formulářové stránky tedy zůstalo primárním způsobem získání vícehodnotových vstupů. Jedinou výjimkou je přidávání nových kroků během vytváření checklistu — požadované hodnoty jsou dvě (text kroku a počet dní před termínem, kdy má být krok splněn) a vzájemně se přímo

neovlivňují. Opět experimentálně se ukázalo, že na tomto konkrétním místě uživatelům více vyhovoval dvoukrokový dialog než samostatná stránka s dvěma poli.

Posledním z využitých dialogů je dialog pro zvolení uživatelské akce. Na některých místech v aplikaci má uživatel možnost nad jedním objektem provést několik operací. Jelikož obrazovka mobilního telefonu je omezená prostorem, může být výhodné tyto operace schovat za jediný aktivní prvek, který po stisknutí nabídne všechny možnosti, které zastupuje. Nevýhodou takového řešení je nutnost provedení jedné uživatelské akce navíc, využití tohoto přístupu tedy závisí na konkrétní situaci. Při uživatelském testování vyvíjené aplikace byl tento způsob schování více možností uživateli upřednostňován před přímým umístěním více tlačítek pro všechny akce. Dialog pro výběr operace je tedy v aplikaci využíván na několika místech. Dialog výběru z dostupných akcí je možno vidět v pravé části obrázku 6.3.

6.3.4 Zobrazení a úprava záznamů

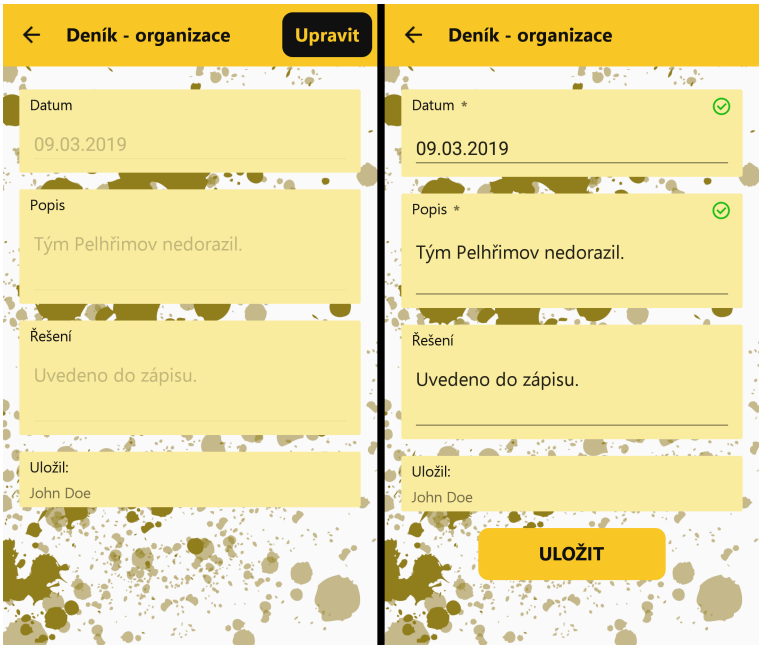
Pro zobrazení seznamu položek byl využit stejný vizuální element, který byl původně vytvořený během návrhu pro seznam upozornění na první stránce aplikace, a který je možné vidět na obrázku 5.2. Element prošel vizuální úpravou, avšak jeho obsah se od původního návrhu nezměnil. Každá položka sestává ze čtyř částí — dvou textových řádků pro hlavní titulek a doplňkový popis ve středu a dvou ikon po obou stranách textu, kdy levá z nich je určená pro lepší vizuální odlišení různých typů zobrazovaných položek a pravá ikona může sloužit pro dokreslení zobrazení, či pro jiné specifické účely (v mnoha případech však není vůbec přítomna). Obrázek 6.4 znázorňuje příklady různých seznamů položek, další příklady jsou pak na obrázcích 8.1, 8.2 a 8.3.



Obrázek 6.4: Ukázka některých seznamů položek vytvořených pomocí popsané šablony. Na levé straně je seznam záznamů v deníku, uprostřed seznam manuálů (zde je také ukázka využití plovoucího akčního tlačítka) a v pravé části je seznam termínů s počtem omluvených účastníků.

Potřeba filtrování záznamů byla vyřešena umístěním filtru do horní lišty stránky. V ní bylo původně umístěno také tlačítko pro přidání záznamu, při testování však uživatelé preferovali jeho přemístění do pravého spodního rohu ve formě plovoucího akčního tlačítka, které je typické pro *material design* [5]. Ukázka použití takového tlačítka je na obrázku 6.4 v části se sekcí „Manuály“.

Pro přehledné prohlížení konkrétního záznamu se ukázalo jako nejvhodnější využít stejného formuláře jako pro jeho vytvoření. Pro úpravu záznamu přirozeně také. Výše popsané formulářové prvky byly tedy jednoduše doplněny pro režim prohlížení, kdy není možné jednotlivé prvky upravovat. Pokud má daný uživatel možnost zobrazený záznam také upravovat, je možné se do režimu úprav přepnout pomocí tlačítka v horním panelu. Rozdíl režimů zobrazení a úpravy je patrný na obrázku 6.5.



Obrázek 6.5: Rozdíl vizuální podoby formuláře v režimu zobrazení záznamu (vlevo) a úpravy záznamu (vpravo).

6.4 Docházka

Docházková část je jednou z klíčových součástí celého implementovaného systému. Během specifikace (viz kapitola 3.2) byly definovány dvě hlavní části tvořící docházku — omluvenky účastníků události (tedy hráčů, případně jejich rodičů) před jejím konáním a samotné zaznamenání účasti na této události zodpovědným trenérem. Do první iterace implementace (viz kapitola 5.1) byla původně zahrnuta pouze druhá část. Během konzultací detailů však vyšla najevo skutečnost, že možnost omluvit hráče by měla být přístupná také pro trenéry jejich kategorií. Implementace omluvenek proto byla přidána již do této verze — prozatím však pouze pro trenéry.

6.4.1 Omluvenky

Obsahem každé omluvenky musí být několik nutných údajů, zejména kdo a kým je omlouván, jakých událostí se omluvení týká a jaký je důvod omluvení. Autora omluvenky je samozřejmě možné určit automaticky na základě uživatele, který akci provádí. Ostatní informace vyplní uživatel pomocí formuláře. Důvod omluvení vybere uživatel z předpřipravené nabídky, případně doplní textovým komentářem.

Výběr události je možný vybráním konkrétní události z nabídky všech událostí hráče. Alternativou je možnost zadat časový interval platnosti omluvenky a systém automaticky omluví hráče ze všech událostí v tomto období — tento způsob je pro uživatele příjemnější než ruční výběr většího množství konkrétních událostí.

Součástí požadavků na omluvenky je i možnost jejich zobrazení trenérem ještě před začátkem události. Důvodem je především možnost kontroly počtu účastníků a včasného zrušení události při nízkém počtu. Zobrazení bylo proto zvoleno jako seznam blížících se událostí, kdy u každé z nich je dispozici údaj o počtu omluvených a celkovém počtu účastníků. Omluvenky se všemi detaily je poté v případě potřeby možno zobrazit pro každou konkrétní událost.

6.4.2 Zadání docházky

Zadání docházky účastníků na události doznalo během vývoje několika změn. Nejvýraznější z nich je kompletní změna prvního kroku (viz kapitola 5.2.3). Původní záměr, kdy trenéři měli mít možnost libovolně zapisovat docházku na základě určení data, kategorie a typu události, byl nahrazen umožněním doplnit docházku pouze k událostem nacházejícím se v klubovém kalendáři. Původní formulář tohoto kroku byl tedy výrazně zjednodušen na pouhý výběr konkrétní události. Lze přitom předpokládat, že docházku budou trenéři (stejně jako doposud papírově) vyplňovat přímo během události, případně těsně po jejím skončení. Aplikace tedy může jednoduše předem vyplnit požadované pole a původní záměr přímočarého prvního kroku zmíněný v kapitole 5.2.3 zůstal zachován.

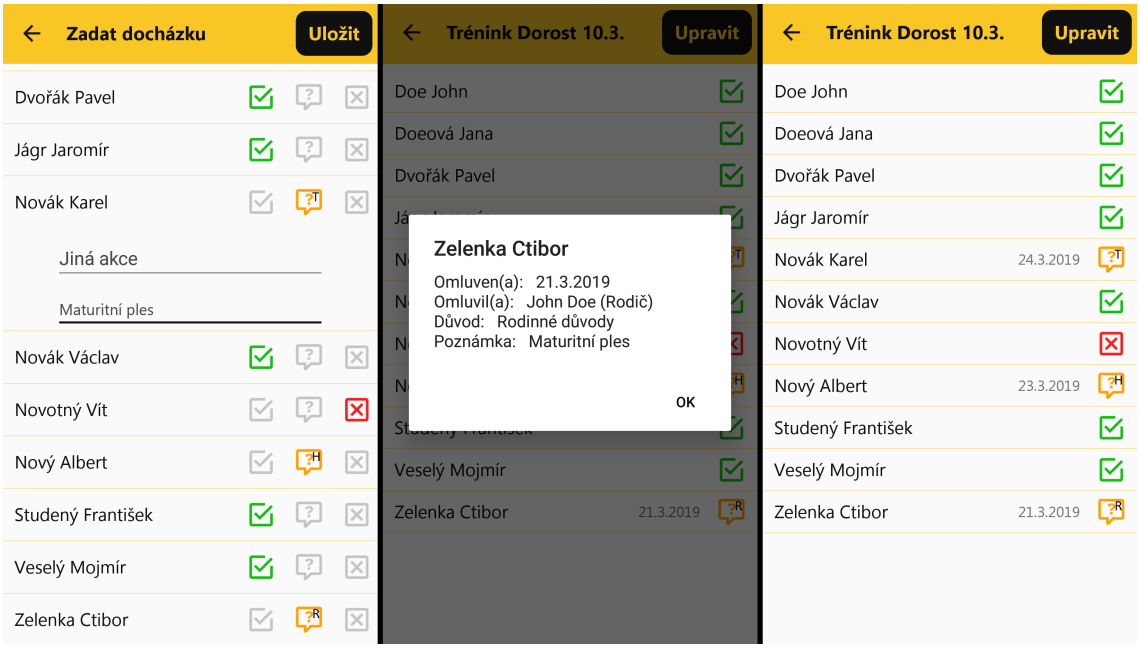
Úpravy druhého kroku (tedy samotného vyplnění účasti hráčů) jsou mnohem mírnější. Oproti původnímu návrhu došlo k upravení podoby jednotlivých ikon určených pro vyplnění účasti či neúčasti. Zároveň je znak zastupující volbu `omluven` doplněn v pravém horním rohu o malé písmeno — H (hráč), R (rodič) nebo T (trenér) — které značí, kým byl hráč z události omluven.

Druhá změna vyplývá ze specifikace omluvenek popsané výše (viz kapitola 6.4.1). Jelikož při každém omluvení je nutné doplnit důvod nepřítomnosti hráče, tak i při nastavení účasti na `omluven` při vyplňování docházky trenérem by důvod nepřítomnosti měl být vyplněn. Ukázka vyplnění docházky události je na obrázku 6.6.

6.4.3 Zobrazení docházky

Z původních tří způsobů zobrazení docházky trenérům v kapitole 5.2.3 byly v úvodní iteraci implementovány pouze dva — detail události a detail hráče. Oba tyto případy byly vyřešené podobným způsobem s využitím stejných ikon, jako v případě zadávání docházky a příklad je znázorněn na obrázku 6.6. Obě možnosti zobrazení hodnotily trenéři během testování jako užitečné.

Naproti tomu třetí z navrhovaných způsobů, tedy tabulka obsahující přehled o docházce celé kategorie (všech hráčů na všechny události), byl z pohledu trenérů nakonec vyhodnocen jako zbytečný. V kombinaci s nepřehledností takové tabulky na malé obrazovce byl tento



Obrázek 6.6: Ukázka zadávání docházky (vlevo) a zobrazení vyplněné docházky pro konkrétní událost (vpravo) spolu s detailem omluvenky hráče (uprostřed).

bod z mobilní verze aplikace vypuštěn. Popsaná tabulka je však hodnotným přehledem chodu celé kategorie pro administrátora oddílu, její realizace by tedy mohla být součástí případného administrátorského rozhraní v budoucnu.

Kapitola 7

Serverová část systému

Nedílnou součástí celého systému je kromě klientské aplikace také jeho serverová část. Dle původního návrhu v kapitole 5 měla vzniknout jako součást již fungující webové stránky klubu. K aktuální prezentační části webu měla být připojena služba určená pro potřeby vznikající aplikace, důvody pro tento způsob řešení byly dva. Jedním z důvodů je finanční stránka — využití již registrované služby by snížilo náklady na zavedení nového systému. Druhým důvodem bylo možné budoucí propojení obou částí a využití oddílového webu například pro administrátorské rozhraní.

Po analýze tohoto řešení se však ukázala jeho realizace jako příliš komplikovaná a serverová část proto byla oddělena jako samostatná služba. Služba vznikne s využitím frameworku `ASP.NET Core`[1], který je multiplatformní a umožňuje skloubit dva důležité aspekty — vývoj pomocí jazyka `C#` (preference autora systému) a možnost běhu na linuxovém serveru (finančně výhodnější oproti alternativě s operačním systémem `Windows`).

7.1 Architektura serverové aplikace

Implementovanou serverovou aplikaci je možné rozdělit do tří hlavních částí typických pro aplikace podobného účelu — `REST API`, aplikační logiku a databázi pro perzistenci všech aplikačních dat.

Databáze byla vytvořena jako relační [21] a obsahuje všechna data potřebná pro běh systému, kromě několika výjimek ve službách třetích stran (události v kalendáři, novinky na webu,...).

Část označená jako `REST API` je vrstva, která definuje způsob, kterým můžou klienti se serverem komunikovat. Vrstva byla vytvořena co nejužší — jedinými zodpovědnostmi této vrstvy jsou jednak logování přicházejících požadavků pro řešení případných problémů a především autorizace těchto požadavků a vyvolání odpovídající obsluhy ve vrstvě aplikační logiky. Tento přístup je dodržen s ohledem na potenciální rozšíření systému v budoucnu, kdy by mohlo dojít k vytvoření webového rozhraní. V takovém případě bude možné tuto webovou aplikaci napojit na samotnou logickou vrstvu, bez nutnosti zbytečného využití vrstvy `REST API`.

Logická vrstva je nejrozsáhlejší součástí serverové aplikace. Pro udržení přehlednosti a budoucí snadnou úpravu je implementována s využitím několika vhodných návrhových vzorů. Kostru celé vrstvy tvoří knihovna `Riganti.Utils.Infrastructure`[2], která je po-

[1] https://docs.microsoft.com/en-us/aspnet/?view=aspnetcore-2.2
[2] https://github.com/riganti/infrastructure

stavena na těchto návrhových vzorech (zejména *Repozitory*, *Query* a *Unit of Work*) [4] a vytváří generický základ pro implementaci většiny součástí logické vrstvy, zejména části přistupující k databázi.

Samotnou vrstvu aplikační logiky je možné dále rozdělit do několika dílčích součástí. Vstupními body pro požadavky z `API` (případně např. z výše zmíněné webové aplikace) jsou takzvané *fasády* (dle návrhového vzoru *Facade* [4]). Jednotlivé fasády jsou typicky vytvořené pro každou samostatnou entitu zvlášť — existuje samostatná fasáda pro docházku, deník, konverzace, uživatele apod. Administrátorské operace jsou taktéž odděleny od ostatních. Příklady fasád jsou třídy `AdminNotificationFacade`, `EventsFacade` nebo `ChecklistFacade`.

Další součástí jsou objekty zapouzdřující databázové dotazy — takzvané `query` objekty (dle návrhového vzoru *Query*, viz výše). Jednotlivé dotazy jsou psané pomocí `LINQ`[3] výrazů v jazyce `C#`. Tyto výrazy jsou při volání využity ORM nástrojem *Entity Framework*[4], který se postará o samotné získání dat z databáze. Pro transformaci samotných uložených relačních dat (ve zdrojových textech třídy se sufixem `Entity`) na datové objekty určené pro manipulaci a přenos ke klientům (třídy se sufixem `DTO` — zkratka pro `Data Transfer Object`, česky `objekt pro přenos dat`) je využita knihovna `AutoMapper`[5].

Poslední částí logické vrstvy stojící za zmínku jsou implementace tříd využívajících služeb třetích stran, kterým se věnuje kapitola 7.2.

Diagram zjednodušené struktury aplikační logiky serveru se nachází v příloze B na obrázku B.2.

7.2 Využití služeb třetích stran

Aplikace ke svému běhu využívá několik externích služeb. Důvody jejich využití jsou dva — získání dat oddílu uložených v těchto externích službách (*Google kalendář*, klubový web) a zaslání dat uživatelům alternativní cestou než přímo skrze klientskou aplikaci (SMS, notifikace). Napojení jednotlivých částí systému na služby třetích stran je znázorněno v diagramech struktury systému v příloze B. Následující řádky podrobněji popisují využití obou typů služeb.

7.2.1 Externě uložená data

Aplikace využívá tři externí zdroje dat. Jedním z nich je klubový web, na kterém jsou publikovány různé zprávy, které jsou požadovány uvnitř aplikace. Jelikož je zbytečné tyto novinky přeposílat klientům skrze server, server slouží pouze pro notifikování uživatelů o nové zprávě. Klientská aplikace obsah této zprávy poté získá přímo z klubového webu.

Druhým zdrojem dat je klubový kalendář vedený pomocí služby *Google kalendář*[6] (viz kapitola 3.2), který obsahuje všechny události oddílu. I zde se nabízí implementace stejnou cestou jako u novinek — tedy přímý přístup klientské aplikace ke kalendáři bez prostředníka v podobě serveru. Problémem je ovšem skutečnost, že kalendář je veden v privátním režimu. Pro přístup k němu by tedy uživatelé museli aplikaci zpřístupnit své přihlašovací údaje k účtu u *Google* (který také teoreticky někteří nemusí mít) a dále by jejich účet musel být administrátorem manuálně povolen. Jednodušší cestou je tedy vytvoření tzv. *servis-*

[3] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/
[4] https://docs.microsoft.com/en-us/ef/
[5] https://automapper.org/
[6] https://developers.google.com/calendar/

ního účtu[7], který je využit pro přístup ke kalendáři ze serverové strany systému. Události jsou tedy klientským aplikacím zprostředkovány skrze server. Pro minimalizaci prodlevy způsobené tímto zdvojeným voláním ukládá server načtené události v paměti, kde jsou uchovávány pro rychlé použití při dalším požadavku — uživatelé tedy při používání aplikace zdržení z důvodu komunikace s kalendářem nepocítí (kromě případů, kdy dochází k periodickému obnovování dat z kalendáře z důvodů reflektování případných změn v nich).

Posledním případem využití externích dat je import hráčů a uživatelů, který je blíže popsán v kapitole 7.3.

7.2.2 Alternativní datové kanály

Pro některé případy je potřeba využít externích služeb pro zaslání dat k uživatelům. Prvním tímto případem je zaslání registrační SMS sloužící pro autorizaci uživatele při instalaci aplikace. Pro tuto funkci je využita služba `SMS brána`[8]. Samotné použití služby je triviální a není tedy třeba ho blíže specifikovat.

Další důležitou součástí systému je využití tzv. *push notifikací*. Běžná komunikace mezi klienty a serverem probíhá na základě dotazu klienta a odpovědi serveru. V určitých situacích (nová zpráva, záznam v deníku, blížící se událost apod.) je však třeba, aby server informoval klienta přímo. Právě pro tyto situace slouží notifikace, běžně používané mnoha mobilními aplikacemi. Detaily záměrů pro využití notifikací v implementované aplikaci jsou popsané v kapitole 5.2.2.

Pro implementaci notifikací byla využita služba *Firebase Cloud Messaging*[9], zkráceně FCM. Každá klientská aplikace se při instalaci automaticky zaregistruje pro přijímání notifikací a získá svůj unikátní klíč. Tento klíč je uložen na serveru a v případě potřeby zaslání notifikace konkrétnímu uživateli, je tento klíč spolu s daty předán službě FCM, která provede doručení dat skrze notifikaci na zařízení uživatele. Obsluha takové přijaté notifikace je už v moci samotné klientské aplikace a je popsaná v kapitole 8.2.

7.3 Automatizovaná správa

Server je centrální částí celého systému, a přestože většina zodpovědnosti za správu leží v rukou administrátora, určité úkony je možné automatizovat v rámci serveru. Následující odstavce popisují právě takové případy.

Po většinu času server pracuje tím způsobem, že reaguje na akce vyvolané uživateli. Pro určité úkony, zejména při nutnosti detekovat určitou změnu nebo jinou událost, je však vhodnější samostatná činnost serveru, který následně může uživatele samostatně upozornit na určitou záležitost (typicky pomocí *push notifikací* popsaných výše). Hlavními činnostmi, které vyžadují tento přístup jsou:

- informování o novinkách na webu oddílu,
- upozornění uživatele na blížící se událost, brigádu apod.,
- údržba samotného serveru — například mazání starých souborů, upozornění administrátora na nutnou údržbu apod.,

[7] https://cloud.google.com/compute/docs/access/service-accounts
[8] https://www.smsbrana.cz/
[9] https://firebase.google.com/docs/cloud-messaging/

- odstranění rodičovských vazeb u hráče, který dovršil 18 let,
- uzavření sezony — například uzamčení docházky.

Samotná implementace těchto samostatných činností je možná pomocí tzv. *hostovaných služeb*[10], které jsou k dispozici přímo v platformě `ASP.NET Core`. Každá služba implementuje jednak samotnou akci, kterou je třeba vykonat, a zároveň specifikuje interval, v jakém se má akce opakovaně provádět. Příkladem implementace takové služby je například třída `ClubNewsTask`.

Jedním z dodatečných požadavků, které vyvstaly během vývoje, byl záměr vedení oddílu udržovat detailní databázi hráčů (včetně kontaktů na rodiče) odděleně od samotného systému. Systém však bez těchto dat nemůže fungovat. Jako řešení byla definována nutná podmnožina zmíněné databáze, která musí být systému k dispozici, a kterou bude administrátor pravidelně poskytovat pro synchronizaci. Výsledkem byla definice formátu `.csv` souboru, který slouží právě pro účely synchronizace databáze účastníků uvnitř a vně systému.

Během synchronizace se server stará o několik věcí. Základem je aktualizace dat u samotných hráčů (kategorie, telefonní kontakty,...), ze které následně vyvstávají další úkony. Zejména se jedná o vytvoření a udržování uživatelů patřících k jednotlivým hráčům (tedy samotní hráči a jejich rodiče) a jejich rolí pro konkrétní kategorie. Důležitou součástí této operace je odstranění již nepotřebných vazeb.

7.4 Nasazení serverové aplikace

Součástí implementace serverové části systému je zajištění jejího hladkého a bezpečného běhu na přístroji dostupném z internetu. Jak již bylo zmíněno na začátku této kapitoly, cílovým operačním systémem, na kterém server poběží, má být některá z linuxových distribucí.

Serverová aplikace vytvořená pomocí `ASP.NET Core` však není po svém publikování vhodná pro samostatné fungování. Především z bezpečnostních důvodů je nutné mezi samotný proces spuštěného serveru a vnější svět (tedy internet) vložit mezičlánek — tzv. *proxy server*[11]. Tento mezičlánek slouží jako prostředník pro příchozí požadavky, které filtruje a předává dál (ve větších aplikacích slouží také pro rozdělení zátěže mezi více instancí serveru, to však není případ této aplikace).

Zvoleným *proxy* serverem pro naší aplikaci je řešení zvané `nginx`[12]. Tato služba běží na samostatném síťovém portu (port samotné aplikace je blokován pro vnější komunikaci) a příchozí požadavky přeposílá na implementovaný server pomocí definované konfigurace.

Součástí zabezpečení komunikace mezi serverem a klientskými aplikacemi je šifrování přenášených dat (které je v rámci této aplikace vynucené a server na požadavky pomocí nešifrovaného protokolu http neodpovídá). Zajištění šifrování je poměrně přímočaré — po získání potřebného certifikátu (v mém případě pomocí služby `Let's encrypt`[13]) stačí upravit konfiguraci `nginx` a serverová aplikace je od této chvíle pro klienty dostupná pomocí šifrované komunikace přes protokol https.

[10] https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services
[11] https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-2.2
[12] https://www.nginx.com/
[13] https://letsencrypt.org/

Kapitola 8

Dokončení první verze

Po vytvoření základního funkčního prototypu mobilní aplikace (kapitola 6) a také serverové části, která s aplikací komunikuje (kapitola 7), je dalším přirozeným krokem napojení mobilní aplikace na zmíněný server a dokončení první použitelné verze.

Během konzultací při vývoji uživatelského rozhraní byly sesbírané připomínky uživatelů přímo zahrnuty do dalšího postupu a uživatelské rozhraní se iterativně měnilo do podoby popsané v kapitole 6. Zároveň došlo také k ujasnění některých detailů, které byly v původní specifikaci (viz kapitola 3.2) vynechány.

Následující odstavce popisují některé problémy, které bylo nutné vyřešit pro úspěšné dokončení první verze aplikace.

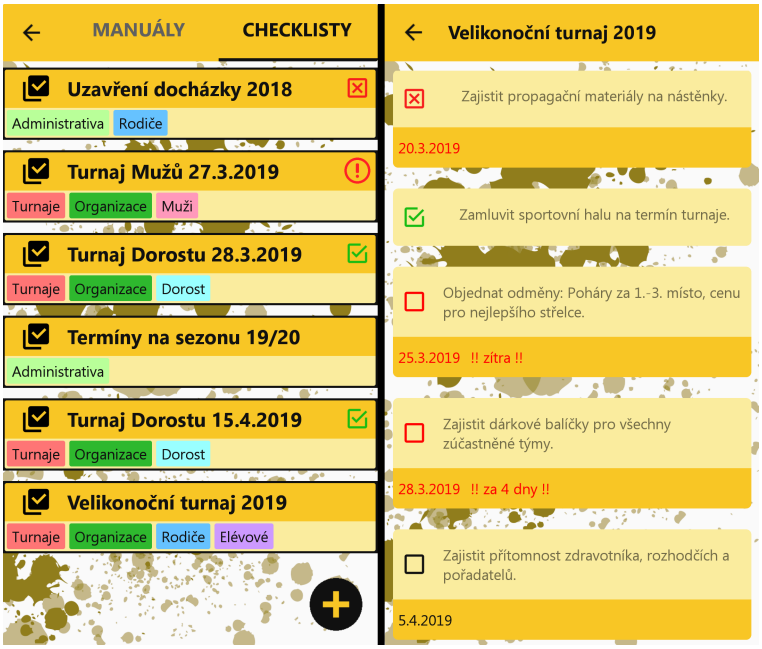
8.1 Funkcionalita organizačních záležitostí

Dle průběžné komunikace s uživateli postupně vyvstaly dodatečné funkce, které je nutné implementovat již do první verze aplikace, aby její používání mělo potřebný smysl. Zejména se jednalo o doplnění funkcí svázaných s pořádáním turnajů a dalších událostí — tedy přehledy manuálů, checklistů (vysvětlení počeštění tohoto anglického výrazu viz kapitola 6.3.1) a brigády.

8.1.1 Implementace checklistů

Realizace checklistů je jednou ze součástí, která má za cíl zjednodušit plnění uživatelových povinností popsaných během analýzy požadavků. Princip má být takový, že administrátor vytvoří konkrétní checklist platný k nějakému datu (typicky vázaný na nějakou oddílovou událost) a přiřadí checklistu odpovědného uživatele. Ten pak musí zajistit včasné splnění všech kroků. Jako klíčový aspekt byla ze strany uživatelů definována nutnost rychlé a přehledné orientace mezi checklisty, a to ve dvojím smyslu — jednak mezi typy checklistů (pořádání turnajů, interní povinnosti,...) a také na základě stavu konkrétního checklistu (vzdálený, blížící se, splněný,...).

Z důvodu konzistence uživatelského rozhraní bylo vhodné udržet seznam checklistů v podobě blízké jiným seznamům v aplikaci (viz kapitola 6.3.4) a tuto podobu pouze doplnit o určité specifické prvky. Pro rychlý přehled o stavu konkrétního checklistu byla využita přítomnost volitelné ikony v pravé části zobrazeného prvku. Výsledek je možné vidět na obrázku 8.1, kde ikony v pravé části označují checklisty ve stavu: *nesplněno* (červený kříž), *blížící se* (červený vykřičník) a *splněno* (zelená ikona). Checklisty bez ikony jsou prozatím nesplněné a se vzdáleným termínem splnění.



Obrázek 8.1: Ukázka seznamu checklistů včetně použití štítků (vlevo) a stránka detailu checklistu s jednotlivými kroky v různých stavech (vpravo).

Pro odlišení typů checklistů byl navržen systém tzv. štítků, částečně inspirovaný štítky v aplikaci *Google keep*[1]. Každý štítek obsahuje krátký název a také barvu, která ho především odlišuje od ostatních. Pomocí různých barev pro různé štítky mohou uživatelé rychle rozlišovat zaměření checklistu. Ukázku štítků je možné vidět na obrázku 8.1. Přestože štítky byly v této části vývoje vytvořeny specificky kvůli checklistům, byla jejich implementace oddělena zvlášť — v případě potřeby využít v budoucnu štítky i u jiných entit bude takové doplnění jednoduché.

Na stránce detailu konkrétního checklistu je taktéž nutné zvýraznit stav jednotlivých kroků (s využitím stejných ikon jako na stránce se seznamem). Specifickým prvkem na této stránce je zobrazení data splnění jednotlivých kroků. Uživatelům byly předloženy dvě varianty — jedna s klasickým zobrazením data, druhá místo toho ukazovala počet dní, který zbývá do konce. Po debatě s uživateli byla nakonec použita kombinace obou přístupů, kdy datum je vždy přítomné a zároveň v intervalu pěti dní před tímto datem je informace doplněna i o ukazatel počtu zbývajících dní. Současně byla kladně hodnocena i barevná odlišnost textu (černý text při vzdálenějším termínu, červený při blížícím se a zelený u splněného). Detail ukázkového checklistu je k vidění na obrázku 8.1.

Součástí implementace checklistů je samozřejmě také jejich správa administrátorem. Pro vytváření nových checklistů slouží formulář, vytvořený shodným způsobem jako ostatní formuláře v aplikaci popsané v kapitole 6.3.2. Při konzultaci případů užití spojených s checklisty vyšla najevo skutečnost, že mnohdy spolu více checklistů souvisí a v čase se opakují. Pro lepší uživatelskou zkušenost se správou checklistů je tedy vhodné využít této skutečnosti k ulehčení práce uživatelů. Vznikly proto tzv. šablony checklistů. Šablona umožňuje administrátorovi uložit si opakující se data v checklistech (generický název nebo jeho část,

[1] https://keep.google.com/

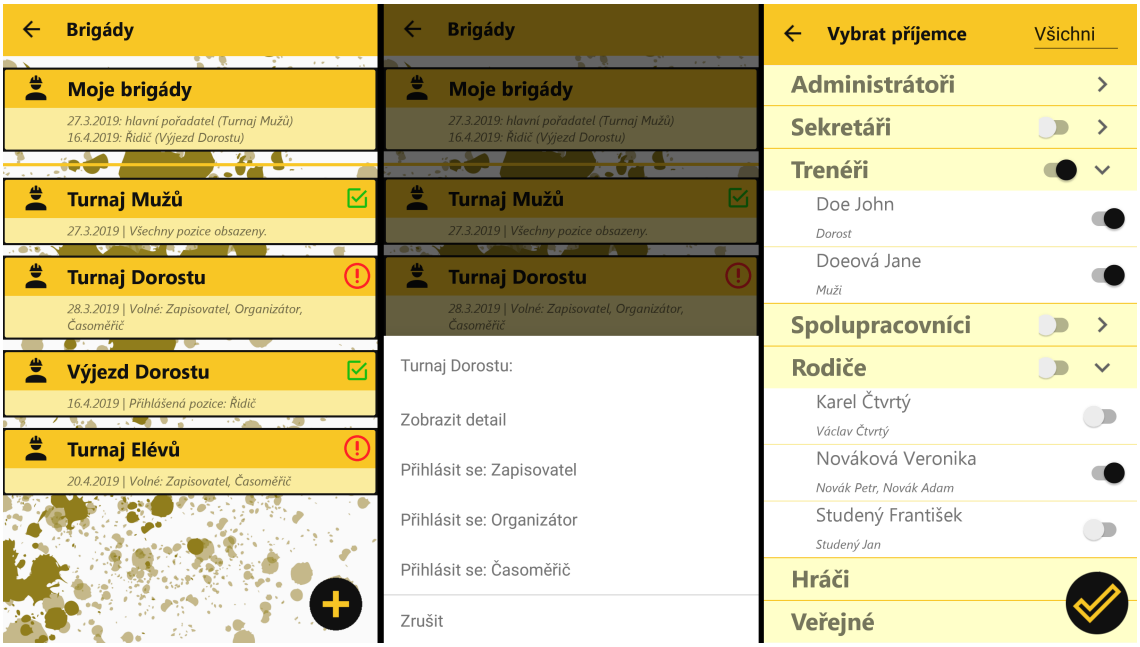
úkony v checklistu, štítky) a při vytváření nového checklistu do této šablony pouze doplnit specifické informace (datum, návaznost na konkrétní událost apod.).

8.1.2 Implementace členských brigád

Druhou důležitou funkcí spojenou s pořádáním událostí je přihlašování uživatelů na brigády, které vytváří administrátor. Každá brigáda má definovaných několik pozic, které je nutné obsadit. I zde je pro udržení konzistence uživatelského rozhraní využito stejného generického seznamu položek. Rozhodující informací je pro uživatele kromě názvu brigády a data především informace o obsazenosti, respektive neobsazenosti pozic. Pro zobrazení této informace přímo v seznamu je možné využít část položky určenou pro doplňující text, kde uživatel vidí právě seznam volných míst, po přihlášení pak naopak název jím přihlášené pozice. Pro přihlášení stačí uživateli vybrat jednu z položek a následně z otevřeného seznamu volných pozic vybrat jím požadovanou.

Součástí formuláře pro vytváření brigád (stejně tak i jiných částí aplikace, například výběr příjemců speciální administrátorské notifikace nebo výběr členů skupinové konverzace) je výběr podmnožiny ze seznamu uživatelů. Pro tento krok byla vytvořena samostatná stránka se seznamem uživatelů, seskupených dle jejich přidělených rolí, ve kterém je možné označit požadované uživatele pro následné použití. Seznam je možné filtrovat dle přiřazené kategorie a taktéž je možné označovat (a rušit označení) hromadně celé skupiny uživatelů. Obě tyto funkce byly velmi kladně hodnoceny administrátorem (který je tím, kdo popsaný seznam využívá při práci s aplikací). Příklad popsaného výběrového seznamu je na obrázku 8.2.

Po otestování uživatelé zmiňovali absenci přehledu jejich přihlášených brigád. Po konzultaci možností řešení tohoto problému byla do horní části seznamu umístěna oddělená položka, která plní právě tento účel. Příklad stránky s brigádami je na obrázku 8.2.



Obrázek 8.2: Přehled stránky s brigádami (vlevo) s dialogovým oknem pro přihlášení (uprostřed) a stránka pro výběr uživatelů (vpravo).

8.2 Notifikace na straně klientské aplikace

Jak již bylo popsáno v kapitole 5.2.2, součástí řešení systému s mobilními klienty je využití tzv. *push notifikací*, které slouží pro komunikaci iniciovanou ze strany serveru. Řešení na straně serveru probírá kapitola 7.2.2, implementaci obsluhy notifikací na straně mobilních klientů popisují následující odstavce.

Během konzultací s vedením oddílu byly probrány určité nedostatky klasických notifikací typických pro mobilní aplikace — tedy zejména skutečnost, že pokud uživatel ručně smaže notifikaci ze seznamu ve svém zařízení, tak je pro něj navždy ztracena. Druhým problémem je teoreticky i skutečnost, že samotné doručování vygenerovaných notifikací není pro platformu *Android* garantované. Po diskuzi nad těmito problémy bylo učiněno rozhodnutí, že kromě zmíněných *push notifikací* budou notifikace implementovány také uvnitř samotné aplikace a obě varianty se budou vzájemně doplňovat (viz zmínka o upozorněních v analýze požadavků v kapitole 3.2).

Samotná implementace přijímání a zpracování notifikací se během vývoje ukázala jako poměrně komplexní problém. Základní rozdíl při přijímání notifikací tvoří stav, v jakém se mobilní aplikace právě nachází (zda je v popředí nebo ne). Pokud se aplikace nenachází v popředí, pak je situace jednodušší — po otevření notifikace je uživatel přesměrován na odpovídající stránku aplikace. Pokud je však aplikace aktuálně otevřená (tedy v popředí), je nutné věnovat zpracování notifikace zvláštní pozornost. Důvodem jsou případy, kdy příchozí notifikace nějakým způsobem ovlivňuje právě otevřenou stránku. Takovou notifikaci je nutné pro zajištění dobré uživatelské zkušenosti okamžitě zpracovat a na její obsah reagovat na otevřené stránce.

Přestože primární účel notifikací je upozornění uživatele na nějakou skutečnost (jako příchozí zpráva, blížící se termín,...), je možné jich využít i jiným způsobem. Jedním ze způsobů využití notifikací, které byly při vývoji využity, je reakce na změnu role uživatele. Změnou role dochází také k získání či ztrátě přístupu uživatele k určitým funkcím v systému. Velkým problémem by mohla být zejména situace, kdy uživatel provede akci, ke které již nemá oprávnění, přestože aplikace instalovaná v jeho telefonu mu tuto operaci umožňuje provést. Řešením takové situace je samozřejmě skutečnost, že autorizace všech požadavků probíhá na serveru a případný neoprávněný požadavek z neaktuálně nastavené aplikace bude odmítnut a snaha uživatele tak skončí chybou.

Toto řešení je však jen částečné — zároveň je nutné provést přenastavení uživatelovy aplikace do takového stavu, který bude reflektovat jeho přiřazené pravomoci. A právě k tomuto účelu mohou dobře posloužit notifikace. V systému byly definovány speciální notifikace navržené pro tyto účely, které jsou odesílány jako tiché notifikace (uživateli se nezobrazí) a na které může aplikace zareagovat patřičným způsobem, například změnou nastavení (i když právě v tomto případě je vhodné uživatele upozornit na probíhající změnu).

Jednou z častých reakcí uživatelů při prvotním testování byla absence indikace nepřečtených notifikací — pokud si prvotní *push notifikaci* ručně vymazali a aplikaci si později otevřeli, nebyl přítomný žádný indikátor toho, že v seznamu upozornění se nějaké nepřečtené nachází. Tato připomínka byla vyřešena umístěním indikátoru nové aktivity do položky v hlavním menu. Jako vhodný indikátor byla navržena a otestována červená tečka, která funguje ve dvou režimech. Po přijetí nového upozornění se rozbliká, čímž silně působí pro získání pozornosti uživatele. Po přechodu uživatele na stránku s upozorněními blikat přestane (uživatel prokazatelně vidí, že má nějaké nepřečtené upozornění a není tedy třeba odvádět pozornost blikáním) a následně zůstává ve stálém tvaru až do chvíle, kdy ze se-

znamu všechny notifikace zmizí. Podoba seznamu notifikací uvnitř aplikace je k vidění na obrázku 8.3 spolu s výše popsaným červeným indikátorem nepřečtených notifikací.

Implementace obsluhy notifikací je založena na knihovně *Firebase Push Notification Plugin*[2]. Z důvodu postupného nárustu různých druhů notifikací potřebných v aplikaci vznikla potřeba abstrahovat jejich obsluhu ještě nad rámec samotné knihovny. Jako generický základ byla implementována základní obsluha v abstraktní třídě `PushNotificationHandlerBase`. Pro každý typ příchozí notifikace (definovaný výčtem `NotificationType`) je poté z této abstraktní třídy odvozena samostatná obslužná třída, která implementuje specifické úkony nutné pro zpracování dané notifikace (úprava zobrazení v seznamu, přesměrování po otevření, obsluha v popředí apod.). Konkrétními příklady implementace jsou například třídy `DirectMessagePushNotificationHandler` nebo `JournalPushNotificationHandler`.

Tyto třídy však nejsou volané přímo, ale jednotlivé úkony jsou pro volání z ostatních částí kódu přístupné v třídě `NotificationFacade`, která obsluhuje všechny požadavky na zpracování notifikace bez ohledu na její konkrétní typ, konkrétně na základě typu získá instanci správné obslužné třídy a tu následně pověří provedením požadované akce. Posledním chybějícím článkem do této hierarchie je třída `PushNotificationHandlerResolver`, tedy právě třída zodpovědná za poskytnutí správné instance pro obsluhu daného druhu notifikace.

Již při první specifikaci požadovaných funkcí byla avizována potřeba administrátora zasílat uživatelům ručně vytvořená upozornění (viz kapitola 3.2). Pro tento účel vznikl krátký formulář, kde administrátor může vyplnit jak textové informace, tak také odkaz na webovou adresu, kam následně notifikace uživatele přesměruje. Součástí formuláře je možnost vybrat pouze některé konkrétní uživatele, kteří upozornění obdrží.

8.3 Doplnění uživatelských rolí

Prozatímní vývoj se soustředil především na funkcionalitu spojenou s rolemi trenéra a administrátora, nikoliv na ostatní role popsané v kapitole 5.1. Velká část potřebné funkcionality pro tyto role je již implementována, protože se shoduje s verzí pro role trenérů a administrátora. Přesto je třeba aplikovat některé úpravy a doplnění, které popisují následující odstavce.

8.3.1 Sekretář a organizátor

Sekretář a organizátor (také označovaný jako externí pracovník) jsou dvě doplňkové role k hlavním rolím vyskytujícím se uvnitř oddílu. Jejich pravomoci v rámci systému jsou podmnožinou pravomocí, které mají právě dvě hlavní role, tedy administrátor a trenér (viz příloha A.3).

K doplnění těchto dvou rolí tedy postačí implementovat základní zobrazení (podobu ikony v menu a konkrétní dlaždice na hlavní stránce) a následně také správně ošetřit přístup k požadovaným funkcím. Z toho důvodu bylo zahrnutí rolí sekretáře a organizátora poměrně přímočaré a není důvod ho detailněji popisovat.

8.3.2 Hráč a rodič

Pro uživatele spadající do rolí hráčů a rodičů (pro zjednodušení budou obě role shrnuty do role *člen*, podobně jako v kapitole 5.1) je nutné, kromě společných funkcí jako jsou

[2] https://github.com/CrossGeeks/FirebasePushNotificationPlugin

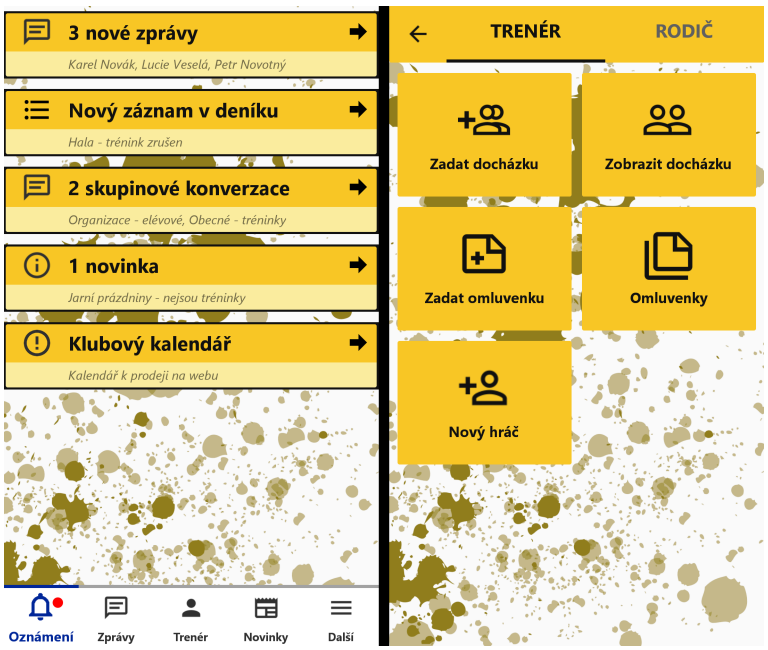
např. zprávy nebo novinky, zpřístupnit i některé specifické části systému — konkrétně části související s organizováním událostí (manuály, checklisty, brigády) a samozřejmě také docházku.

U funkcí spjatých s organizací není nutné žádného zvláštního zásahu. Použití pro tyto role se shoduje se způsobem, jakým zmíněné části aplikace používají již trenéři. Postačí tedy zpřístupnit již implementovanou funkcionalitu i pro další role uživatelů.

Jiná situace je u části aplikace pro správu docházky. V této části je pro trenéry implementovaná možnost správy docházky pro celé kategorie, bude tedy nutné zahrnout určité změny. Členové samozřejmě nemohou vyplňovat docházku a také doplňovat do docházky nové hráče, tyto dvě funkce je možné tedy rovnou odstranit.

U ostatních funkcí (zadání omluvenky, procházení omluvenek a procházení docházky) postačí drobné změny. U omluvenek mohou členové zadávat či prohlížet pouze své omluvenky (v případě rodičů svých dětí). Taktéž prohlížení docházky je nutné omezit pouze na docházku týkající se konkrétního uživatele.

Vzhledem k rozdílné funkcionalitě docházkové části aplikace pro různé role uživatelů je nutné vyřešit potenciální situaci, kdy jeden uživatel bude mít jak roli uvnitř oddílu, tak také roli člena (typicky například trenér některé z mládežnických kategorií může být zároveň hráčem v kategorii mužů, viz kapitola 5.2.4). Tuto kolizi je možné vyřešit dvěma způsoby. Jednou možností je rozdělení docházky do dvou samostatných částí (z výše zmíněného příkladu na část pro trenéra a část pro hráče), druhou pak ponechání jedné docházkové sekce a začlenění akcí obou rolí do stejné stránky. Uživatel by pak například v části s omluvenkami vybral mezi omluvenkou pro sebe či některého ze svěřených hráčů až přímo v konkrétním formuláři.



Obrázek 8.3: Ukázka výsledné podoby seznamu notifikací (vlevo) a podoby sekce *Docházka* pro uživatele s více rolemi.

Z důvodu přehlednosti a jednoznačnosti rozdělení akcí obou rolí nakonec zvítězil první z uvedených způsobů. Přepnutí mezi funkcemi jednotlivých rolí je implementováno pomocí

přepínacího panelu v horní části obrazovky, stejným způsobem jako je tomu v sekci *Organizace* — viz obrázek 8.1, kde je možno vidět přepínací panel stránek *Manuály* a *Checklisty*. Obrázek 8.3 pak ukazuje způsob popsaného řešení v sekci *Docházka*.

8.4 Optimalizace výkonu aplikace

Již v kapitole 2.2.3 bylo zmíněno, že jedním z kritérií ovlivňujících uživatelskou zkušenost s aplikací je rychlost odezvy uživatelského rozhraní. I v klientské aplikaci implementované během této práce je samozřejmě možné najít několik způsobů, jak odezvu při práci s aplikací urychlit. Použitá vylepšení je možné rozdělit na dva druhy — optimalizace samotné aplikace a minimalizace prodlevy způsobené komunikací se serverem. Oba druhy použitých vylepšení popisují následující řádky.

8.4.1 Optimalizace odezvy klientské aplikace

Xamarin Forms obsahuje vestavěné mechanismy, které je možné využít pro zlepšení rychlosti vykreslování uživatelského rozhraní [18]. Dvě z těchto technik jsou tzv. komprese rozložení (`layout compression`) a rychlé vykreslování (`fast renderers`).

Rychlé vykreslování je technika sloužící k omezení počtu vykreslovaných objektů při vytváření stránky za běhu aplikace. Jejím použitím je možné u některých prvků uživatelského rozhraní (především tlačítek a textových elementů) přimět operační systém *Android* (na ostatní platformy tato technika nemá vliv) k vytváření menšího počtu objektů ve vizuálním stromu, což má za následek rychlejší vykreslení stránky.

Na podobném principu pracuje také technika komprese rozložení. Jejím cílem je zamezit zahrnutí vybraných kontejnerových prvků (prvky, které slouží pro rozmístění vnořených prvků — např. *Grid*, *StackLayout* nebo *AbsoluteLayout*) do vizuálního stromu. Využít této možnosti lze pouze u takových kontejnerů, které přímo neupravují vzhled stránky (například barvu pozadí nebo rotaci) a zároveň nereagují na uživatelské akce. I přes tato omezení je možné dosáhnout redukce velikosti vizuálního stromu o desítky procent a tím dosáhnout značného urychlení vykreslení stránky zejména na starších zařízeních.

8.4.2 Minimalizace síťové komunikace

Síťová komunikace se serverem je jednoznačně největším zdrojem zdržení při uživatelské práci s aplikací. Přestože samotnou rychlost komunikace se podaří jen stěží ovlivnit (s výjimkou optimalizace zpracování požadavků na serverové straně systému), je možné snížit její dopad minimalizací objemu komunikace. Pro tento účel bylo uplatněno několik různých postupů.

Základním prostředkem pro snížení objemu přenášených dat je zamezení opakovanému přenášení stejných dat. Pro tento účel si mobilní aplikace určitá data drží v paměti během svého běhu a pouze ve zvolených pravidelných intervalech tato data znovu stahuje ze serveru (pro zajištění reflektování případných změn jiným uživatelem). Jako příklad dat, se kterými je v aplikaci nakládáno tímto způsobem, mohou být kategorie, události nebo manuály.

Ještě o krok dále je možné zajít s použitím lokální perzistence. Té je vhodné využít zejména u takových dat, která se po svém vytvoření již dále nemění — typickým příkladem mohou být zprávy, a to jak přímé zprávy s jinými uživateli, tak také zprávy v hromadných konverzacích. Tyto zprávy (odchozí i příchozí) jsou ukládány v lokální databázi přímo na

mobilním zařízení a není tedy nutné je již při dalším otevření konverzace opakovaně stahovat ze serveru.

Poslední technikou snižující zdržení uživatele při práci je přednačtení dat ze serveru ještě před tím, než je uživatel potřebuje. Taková data jsou v případě potřeby již připravena a uživateli se mohou zobrazit bez dalšího zpoždění. Rizikem tohoto přístupu je možnost zbytečného čerpání zdrojů, kdy jak výpočetní výkon zařízení, tak přenosové pásmo aktuální sítě mohou být předběžným stahováním dat na pozadí využity na úkor vyřizování aktuálních akcí uživatele. Tento přístup byl tedy využit omezeně — hlavním použitím je okamžité stažení všech nepřečtených zpráv při startu aplikace a také načtení některých dat při obdržení notifikace.

8.5 Finalizace a nasazení aplikace

Po dokončení veškeré potřebné implementace nastává poslední krok, kterým je nasazení celého systému do cílového prostředí. Nejdříve je nutné uvést do provozu serverovou část systému, o čemž pojednává kapitola 7.4. Následně již nic nebrání finalizaci klientské aplikace pro mobilní telefony a její publikace mezi uživatele. V této fázi bude aplikace nasazena do obchodu s aplikacemi pro operační systém *Android*. Pro další platformy bude v případě potřeby publikována později.

Jelikož je pravděpodobné, že během používání aplikace narazí uživatelé na některé chyby a další problémy, je vhodné mít k dispozici dobrý mechanismus pro diagnostiku právě takových situací. Pro aplikace vytvořené pomocí nástroje *Xamarin Forms* je možné využít služby *Visual Studio App Center*[3], která (mimo jiné) obsahuje právě diagnostiku publikovaných aplikací. Po začlenění do aplikace je možné ve vhodných situacích odesílat diagnostická data, zároveň jsou automaticky zaznamenána veškerá neočekávaná selhání. Veškerá odeslaná data je poté možné analyzovat pomocí k tomu určeného webového nástroje.

Posledním krokem k publikaci aplikace je doplnění veškerého nutného obsahu pro zobrazení v obchodu s aplikacemi (například obrázky z aplikace, ikona aplikace nebo textový popis) a následné nahrání samotného balíčku s aplikací. Po schválení aplikace ze strany obchodu je aplikace publikována a k dispozici cílovým uživatelům.

[3] https://docs.microsoft.com/cs-cz/appcenter/

Kapitola 9

Vyhodnocení a plán vývoje

Smyslem této kapitoly je rekapitulovat a vyhodnotit dosavadní stav implementované aplikace. Následující odstavce se postupně ohlíží nad zapojením uživatelů do průběhu vývoje a jejich názor na implementovaný systém, diskutuje splnění vytyčených požadavků a cílů a nastiňuje další pravděpodobný vývoj systému v budoucnu.

Vývoj aplikace byl od počátku uspořádán jako uživatelsky orientovaný (viz kapitola 2.4) a cíloví uživatelé byli jeho součástí po celou dobu od úvodního popisu požadavků (viz kapitola 3.2) až do vydání v oficiálním obchodu (viz kapitola 8.5). Uživatelské testování je možné rozdělit do dvou samostatných kategorií — pravidelné uzavřené testování během vývoje a uživatelské testy po vydání aplikace.

9.1 Testování během vývoje

Právě z důvodu snahy o vývoj aplikace s pomocí uživatelsky orientovaného přístupu se základem celého testování staly průběžné testy jednotlivých inkrementálních přírůstků výsledného řešení, které byly postupně implementovány a zpětně upravovány a opravovány. Těchto testů se po celou dobu vývoje účastnilo několik málo referenčních uživatelů, kteří měli k dispozici aktuální rozpracovanou verzi, a průběžně poskytovali zpětnou vazbu k implementované funkcionalitě, ke vzhledu aplikace a k jejímu ovládání. Výstupy těchto drobných uživatelských experimentů jsou průběžně popisovány v předchozích kapitolách věnujících se implementaci konkrétních částí.

Pro ilustraci přínosu těchto testů je možno uvést několik příkladů. Velkou část zpětné vazby v této fázi testování tvořily požadavky na vizuální úpravy zlepšující přehlednost a použitelnost aplikace, konkrétně například odstranění zaoblení v rozích dlaždic na uživatelské stránce, úpravy textů a ikon (zejména ve zmíněných dlaždicích a také některých tlačítkách), změna ikon použitých v docházce (popsané v kapitole 6.4.2) a podobně. Z funkčních požadavků je možné zmínit zejména doplnění stránky s kontakty v komunikační sekci (viz kapitola 6.2) a potřeba implementace omluvenek již v první části vývoje.

Druhou fází uživatelských testů bylo několik komplexnějších experimentů, které následovaly po dokončení ucelených částí aplikace přibližně odpovídajících předpokládaným výstupům prvotně navržených iterací (viz kapitola 5.1). Tato testování probíhala formou osobních schůzek s částí uživatelů, kteří testovali jednotlivé funkce a hodnotili zejména jejich použitelnost a validní chování. Výstupem těchto testů byly neformální poznámky o kladných a záporných názorech uživatelů na konkrétní prvky aplikace.

Z těchto experimentů vzešlo několik námětů k důležitým úpravám a vylepšením, které byly po implementování velice kladně hodnoceny a velmi zlepšily uživatelskou zkušenost s aplikací. Jedním z příkladů těchto vylepšení je možnost „připíchnutí“ důležitých skupinových konverzací na vrchol seznamu popsaný v kapitole 6.2.

Dalším příkladem je požadavek na možnost změny rozmístění jednotlivých dlaždic na stránce s uživatelskými funkcemi dle individuálních preferencí každého uživatele. Při testování sekcí aplikace spojených s organizačními záležitostmi se ukázalo, že někteří uživatelé by preferovali v horní části umístění jiných dlaždic, než které zde v té době byly pevně umístěné. Po diskuzi nad konkrétním rozmístěním všech dlaždic vyšlo najevo, že uživatelé mají na toto rozmístění různý názor. Výsledkem této diskuze byl právě požadavek na možnost úpravy rozmístění za běhu aplikace. Tato možnost byla následně implementována a kladně přijata.

Velmi důležitým výstupem popsaných experimentů byla také skutečnost, že aplikace v některých situacích reaguje pomalu a zdržuje uživatele při práci s ní. Uživatelská zkušenost s používám aplikace se v takových případech velmi zhoršovala. Řešení tohoto problému adresuje kapitola 8.4.

9.2 Finální uživatelské testování

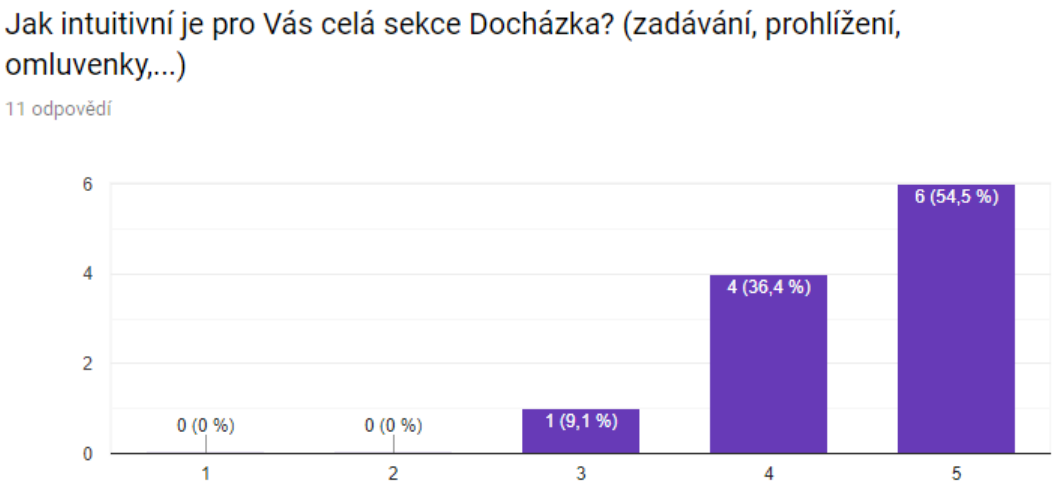
Po zprovoznění systému a rozšíření klientské aplikace mezi uživatele bylo nutné uskutečnit uživatelské testování reálně nasazené aplikace, které by poskytlo zpětnou vazbu o splnění původních cílů vytyčených před začátkem vývoje. Předmětem testování se tedy staly především ty části systému, které tvoří nejdůležitější případy užití aplikace jejími uživateli. Pro účely testování byl vytvořen formulář pokrývající právě tyto nejdůležitější sekce a uživatelé byli instruováni k jeho vyplnění přímo během vykonávání dotčených případů užití.

Otázky byly vytvořeny tak, aby bylo možné většinu z nich vyhodnotit přímočarým kvantitativním způsobem — převážně výběr z předem připravených odpovědí a hodnocení na číselné stupnici. K doplnění těchto odpovědí o podrobnější údaje obsahuje každá sekce formuláře také prostor pro případné poznámky uživatelů. Konkrétní podoba všech kvantitativních otázek je spolu s přehledem odpovědí obsažena v příloze C. Příklady některých otázek i s přehledem odpovědí jsou na obrázcích 9.1, 9.2 a 9.3. Odpovědi byly sesbírány od celkem jedenácti uživatelů, z nichž téměř všichni mají v oddílu roli trenéra (jedinou výjimkou je uživatel v roli sekretáře) a současně většina z nich plní také roli hráče nebo rodiče.

První testovanou částí byla docházka. Z odpovědí vyplynulo, že se podařilo dosáhnout dobré úspěšnosti při předvyplnění požadovaného termínu pro vyplnění docházky (viz kapitola 6.4.2). Při hodnocení intuitivnosti a přehlednosti zadávání docházky (a prohlížení celé docházkové sekce — tedy včetně omluvenek) uživatelé volili odpovědi v spíše v kladné části spektra. Dle doplňujících poznámek je prostor ke zlepšení zejména v lepším naznačení těch položek docházky, které po kliknutí uživatele zobrazují více informací.

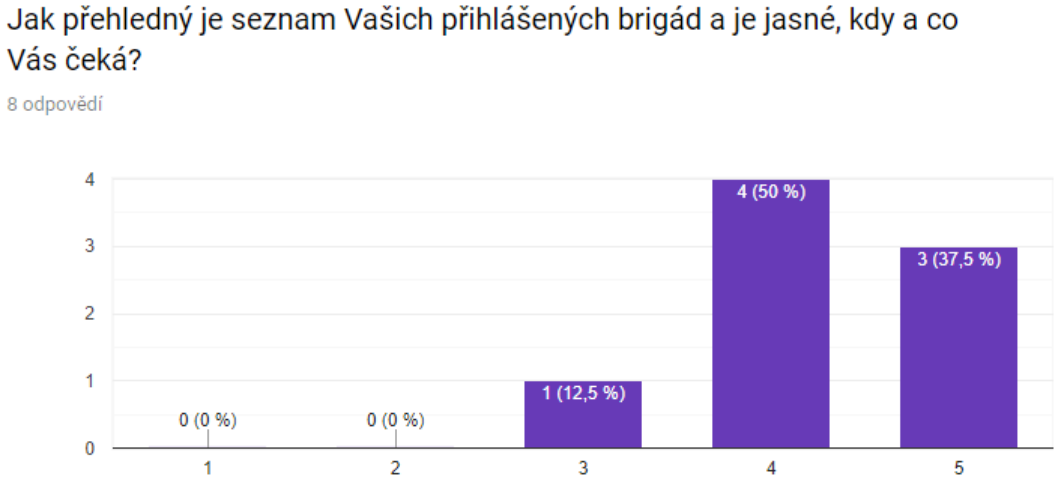
Druhá část formuláře cílila na otestování intuitivnosti a snadnosti použití formulářů v aplikaci. Jako testovací příklad byl vybrán formulář pro záznam o zranění hráče v klubovém deníku, avšak uživatelé byli instruováni k využití těchto otázek pro hodnocení všech formulářů aplikace. Odpovědi v této části se opět převážně blíží ke kladné části stupnice a ani žádné slovní výhrady uživatelé neprojevili.

Třetí sekce testovala stránku aplikace obsahující přihlašování uživatelů na termíny organizací událostí. V této části uživatelé projevili nespokojenost se způsobem zobrazení informace o jimi přihlášených brigádách — zejména absence možnosti přímého zobrazení



Obrázek 9.1: Ukázka jedné z otázek uživatelského testování zaměřená na docházkovou sekci včetně znázorněných odpovědí uživatelů.

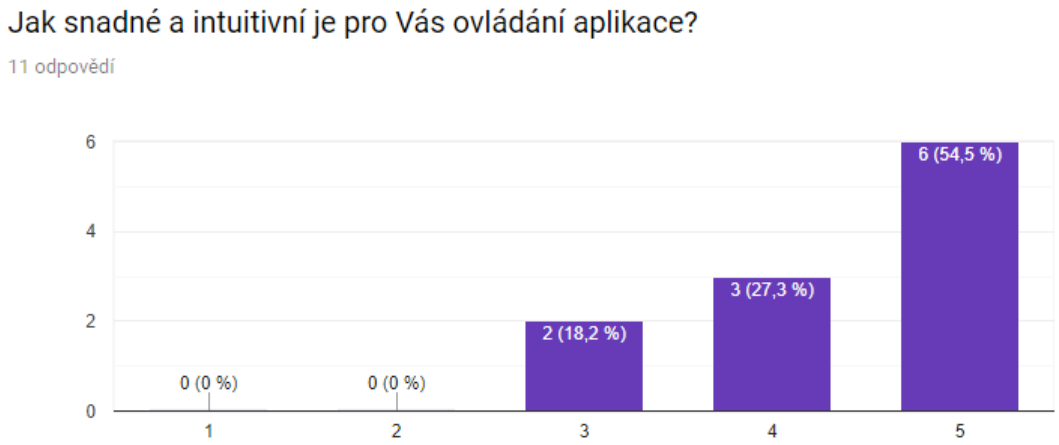
detailních informací v samotné horní části obrazovky určené pro přehled přihlášených událostí a také absence výrazného odlišení přihlášených událostí v seznamu všech dostupných brigád.



Obrázek 9.2: Ukázka jedné z otázek uživatelského testování zaměřená na sekci přihlašování brigád včetně znázorněných odpovědí uživatelů.

Poslední specifickou částí aplikace, která byla formulářem pokryta, je část komunikace. I zde se uživatelé při hodnocení drželi spíše v kladné části stupnice, nicméně vyjádřili také některé drobné nedostatky jako je absence přesného počtu nových zpráv od konkrétního uživatele (či v konkrétní skupině) a také potřebu lepšího zvýraznění konkrétních dosud nepřečtených komentářů uvnitř skupinových konverzací.

Na závěr formulář cílil na zjištění celkové zkušenosti uživatelů při používání aplikace. Stejně jako u specifických částí uživatelé z větší části reagovali spíše kladným hodnocením u otázek na přehlednost navigace a intuitivnost ovládání aplikace. Důležitým zjištěním je také skutečnost, že většině uživatelů se s aplikací dobře pracuje hned od samého začátku používání, případně po krátké době od začátku používání. Také vizuální podoba aplikace je hodnocena dobře s některými drobnými výhradami, které však dle uživatelů nemají vliv na použitelnost aplikace.



Obrázek 9.3: Ukázka jedné z otázek uživatelského testování zaměřená na celkovou zkušenost uživatelů s aplikací včetně znázorněných odpovědí uživatelů.

Z hodnotících uživatelů jeden vyjádřil výraznější problémy s ovládáním aplikace, intuitivností a přehledností jednotlivých částí a celkové navigace. Dle vlastních slov zatím při práci tápe, ale ještě potřebuje více času na seznámení s pro něj novou mobilní aplikací a získání zkušeností s jejím používáním a nechce zatím dělat další závěry. Ostatní uživatelé projevili v těchto otázkách spíše spokojenost. Z těchto důvodů prozatím nebudou podnikány výrazné změny v celkové koncepci aplikace a jejím ovládání a vyhodnocení spokojenosti a zkušenosti uživatelů bude provedeno opět po delším čase jejich sžití s aplikací.

Během prvních týdnů používání aplikace uživatelé objevili a hlásili jeden důležitý problém, který negativně ovlivňuje jejich zkušenost s používáním aplikace, a tím jsou notifikace. Konkrétně se jedná o problém se spolehlivostí doručování *push notifikací* a také chování aplikace při jejich otevření uživatelem. Problémy spojené s reakcí aplikace na otevření notifikací se následně podařilo úspěšně vyřešit, nicméně problém se spolehlivostí doručování notifikací u některých uživatelů přetrvává i v době psaní tohoto textu — postižení uživatelé musí sami aktivně kontrolovat přítomnost notifikace otevřením aplikace — to samozřejmě značně zhoršuje jejich celkový dojem z používání aplikace a je nutné tento problém prioritně adresovat při další práci na implementovaném systému.

9.3 Rekapitulace a vyhodnocení vytyčených cílů

Kapitola 3.2.2 popisuje hlavní cíle, kterých měla implementace systému popsaná předešlými kapitolami dosáhnout. Základní předpoklad — tedy vytvoření a nasazení samotné aplikace v prostředí sportovního oddílu — se podařilo naplnit. Nyní je třeba vyhodnotit splnění jednotlivých cílů a navrhnout další postup ve vývoji systému.

Z pohledu implementované funkčnosti se podařilo splnit všechny klíčové požadavky, jako je komunikace uživatelů, šíření a upozornění uživatelů na nové informace od vedení oddílu, vytvoření docházkového systému, digitalizace klubového deníku, správa povinností uživatelů, koordinace uživatelů směrem k organizaci událostí a další.

Některé z původně plánovaných doplňkových požadavků popsaných v kapitole 3.2.3 byly během vývoje vypuštěny — jedná se zejména o napojení aplikace na data ze systému *České florbalové unie*. Důvodem je publikace samostatné mobilní aplikace *Český florbal*[1], která původně předpokládané případy užití pokrývá, a není tedy důvod vkládat je také do implementované aplikace.

Požadavky na uživatelskou zkušenost s aplikací definované v kapitole 3.2.3 kladly důraz především na intuitivnost uživatelského rozhraní, snadné používání celé aplikace a snadnou orientaci v ní. Dle zpětné vazby od aktuálních uživatelů systému popsané v kapitole 9.2 se tyto požadavky podařilo splnit uspokojivým způsobem jak v hlavních případech užití, tak také na úrovni celé aplikace. Uživatelé hodnotili popsané znaky kladně, a i když byly objeveny některé nedostatky hodné budoucí pozornosti a úpravy, vytvořená aplikace běží v prostředí reálného sportovního oddílu a slouží pro naplnění popsaných potřeb při jeho každodenním běhu ke spokojenosti uživatelů.

9.4 Směr dalšího vývoje

Vývoj systému vznikajícího během této diplomové práce bude pokračovat i nad její rámec. Prvním úkolem bude zlepšení spolehlivosti notifikací, která se během prvních týdnů po nasazení systému ukázala nedokonalá. Následovat bude opravení dalších drobných problémů, které se během testování ukázaly, a které se ještě mohou objevit v týdnech následujících. Všechny tyto změny by měly mít za cíl dostat uživatelskou zkušenost s aplikací do co nejlepší podoby.

Pro zlepšení uživatelské zkušenosti bude také implementováno více možností individuálního nastavení aplikace. Zejména by se mělo jednat o možnost přenastavení barev použitých pro odlišeních jednotlivých uživatelských rolí v komunikační sekci a v případě zájmu uživatelů také o možnost nastavení různých zvuků pro různé druhy přicházejících notifikací.

Následně bude nutné s vedením oddílu identifikovat případné další funkční požadavky na rozšíření aplikace, které prozatím nebyly implementovány, a postupně tyto požadavky realizovat. Součástí tohoto procesu by mělo být také rozhodnutí o potřebě či nepotřebě administrátorského rozhraní zmiňovaného během úvodní analýzy požadavků.

[1] https://play.google.com/store/apps/details?id=cz.tmobile.florbal

Kapitola 10

Závěr

Hlavním tématem této práce je proces vývoje systému určeného k vedení sportovního oddílu. Implementačním výstupem práce je především mobilní aplikace SK JUVENIS PŘIBYSLAV[1] určená pro platformu *Android*, která slouží jako klientská část implementovaného systému. Uživatelskému rozhraní této aplikace se věnuje podstatná část tohoto textu.

Na začátku se práce věnuje studiu principů a metod používaných při vývoji softwarových produktů, zejména pak způsobům návrhu uživatelských rozhraní a tvorbě dobré uživatelské zkušenosti s těmito rozhraními. Následuje přiblížení prostředí sportovního oddílu, důvody k vytvoření softwarového systému pro podporu jeho vedení a požadavky na tento systém získané za pomoci osobních konzultací a pozorování uvnitř samotného oddílu. Součástí studia požadavků na výsledný systém je také přehled několika již existujících aplikací sloužících podobným účelům.

Další části práce se již věnují samotné realizaci požadovaného řešení. Iterativním způsobem se zapojením budoucích uživatelů systému nejprve vznikl návrh dílčích součástí uživatelského rozhraní. S pomocí nástroje *Xamarin Forms* byla následně implementována mobilní aplikace s navrženým uživatelským rozhraním, která byla po celou dobu vývoje konzultována s uživateli, jejichž zpětná vazba se promítla do výsledné podoby aplikace. Kapitoly zaměřené na tuto implementaci popisují zejména postupný vývoj klíčových částí aplikace, které umožňují vykonání hlavních definovaných případů užití jako je komunikace uživatelů, správa docházky, vedení klubového deníku a další.

Nedílnou součástí realizace byla implementace serverové části systému pomocí jazyka `C#` a frameworku `ASP.NET Core`, které se tento text také okrajově věnuje. Následuje popis dokončení první verze určené pro publikaci, v rámci které bylo nutné zejména doplnit funkcionalitu pro více rolí uživatelů, implementovat obsluhu příchozích notifikací a optimalizovat rychlost odezvy aplikace s cílem zajistit lepší uživatelskou zkušenost při používání aplikace.

Po dokončení popsaných částí byla aplikace publikována a nasazena v prostředí sportovního oddílu SK JUVENIS PŘIBYSLAV, kde byla následně přibližně patnácti uživateli využívána při každodenním chodu oddílu. Závěr práce popisuje zpětnou vazbu těchto uživatelů, která potvrzuje použitelnost a užitečnost aplikace pro požadované účely, a tím splnění základního cíle celé práce. Pokračující vývoj systému nad rámec této práce se bude věnovat zejména dalšímu zlepšení uživatelské zkušenosti s jeho používáním a na základě dalších konzultací s vedením oddílu případným rozšířením poskytované funkcionality.

[1] https://play.google.com/store/apps/details?id=juvenispribyslav.droid

Literatura

[1] Babich, Nick: *Basic Patterns for Mobile Navigation*. [Online; navštíveno 9.11.2017].
URL http://babich.biz/basic-patterns-for-mobile-navigation/

[2] Babich, Nick: *UX Design for Mobile: Bottom Navigation*. [Online; navštíveno 9.11.2017].
URL https://uxplanet.org/perfect-bottom-navigation-for-mobile-app-effabbb98c0f/

[3] Cockburn, A.: *Agile Software Development*. Addison-Wesley, 2002, ISBN 0-201-69969-9.

[4] Fowler, M.: *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003, ISBN 0-321-12742-0.

[5] Google: *Material Design*. [Online; navštíveno 9.11.2017].
URL https://material.io/guidelines/

[6] Hermes, D.: *Xamarin Mobile Application Development*. Apress Media LLC, 2015, ISBN 978-1-4842-0215-9.

[7] Jacobson, Ivar: *Use case 2.0, The Guide to Succeeding with Use Cases*. [Online; navštíveno 14.12.2017].
URL https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf

[8] Křena, Bohuslav and Kočí, Radek: *Úvod do softwarového inženýrství IUS, Studijní opora*. [Online; navštíveno 8.1.2018].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/texts/IUS_opora.pdf

[9] Michálek, Martin: *Webové formuláře na mobilech*. [Video; Online; navštíveno 8.11.2017].
URL https://slideslive.com/38903186/webove-formulare-na-mobilech

[10] Nielsen, Jakob: *Introduction to Usability*. [Online; navštíveno 26.10.2017].
URL https://www.nngroup.com/articles/usability-101-introduction-to-usability

[11] Nielsen, Jakob: *Iterative User Interface Design*. [Online; navštíveno 26.10.2017].
URL https://www.nngroup.com/articles/iterative-design

[12] Nielsen, Jakob: *Parallel & Iterative Design + Competitive Testing = High Usability.* [Online; navštíveno 26.10.2017].
URL https://www.nngroup.com/articles/parallel-and-iterative-design

[13] Preece, J. a.: *Human-Computer Interaction.* Addison-Wesley, 1994, ISBN 0-201-62769-8.

[14] Sandu, Bogdan: *What Is UX Design And Why It's Important.* [Online; navštíveno 5.11.2017].
URL http://www.designyourway.net/blog/web-and-mobile-design/what-is-ux-design-and-why-its-important/

[15] Shneiderman, B.; Plaisant, C.: *Designing the user interface: strategies for effective human-computer interaction.* Addison-Wesley, 2010, ISBN 978-0-321-53735-5.

[16] Site: UX Planet: *Horizontal scrolling in mobile, UX of scroll interface.* [Online; navštíveno 9.11.2017].
URL https://uxplanet.org/horizontal-scrolling-in-mobile-643c81901af3/

[17] Site: Xamarin: *Xamarin.Forms.* [Online; navštíveno 8.1.2018].
URL https://www.xamarin.com/forms

[18] Site: Xamarin documentation: *Layout compression.* [Online; navštíveno 24.11.2018].
URL https://docs.microsoft.com/en-us/xamarin/xamarin-forms/deploy-test/performance

[19] Wiegers, K.: *Požadavky na software.* Computer Press a.s., 2008, ISBN 978-80-251-1877-1.

[20] Zendulka, Jaroslav and Bartík, Vladimír and Květoňová, Šárka: *Analýza a návrh informačních systémů AIS, Studijní opora.* [Online; navštíveno 2.11.2017].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/AIS-IT/texts/AIS.pdf

[21] Zendulka, Jaroslav and Rudolfová, Ivana: *Databázové systémy IDS, Studijní opora.* [Online; navštíveno 16.2.2018].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php/course/IDS-IT/texts/in_Czech/Databazove_systemy.pdf

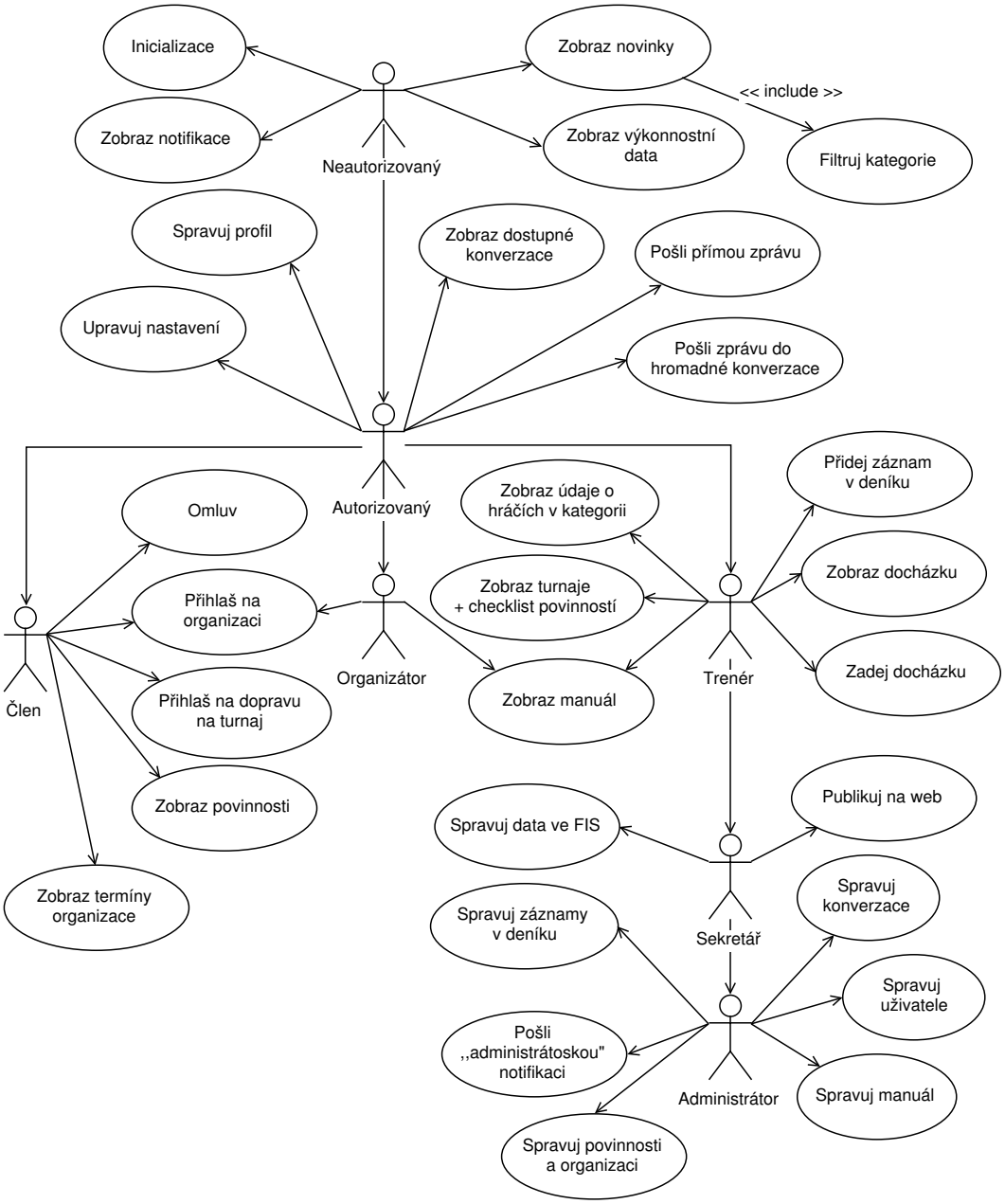
Příloha A

Diagramy případů užití

Obrázek A.1: Diagram případů užití v první iteraci

Obrázek A.2: Diagram případů užití v druhé iteraci

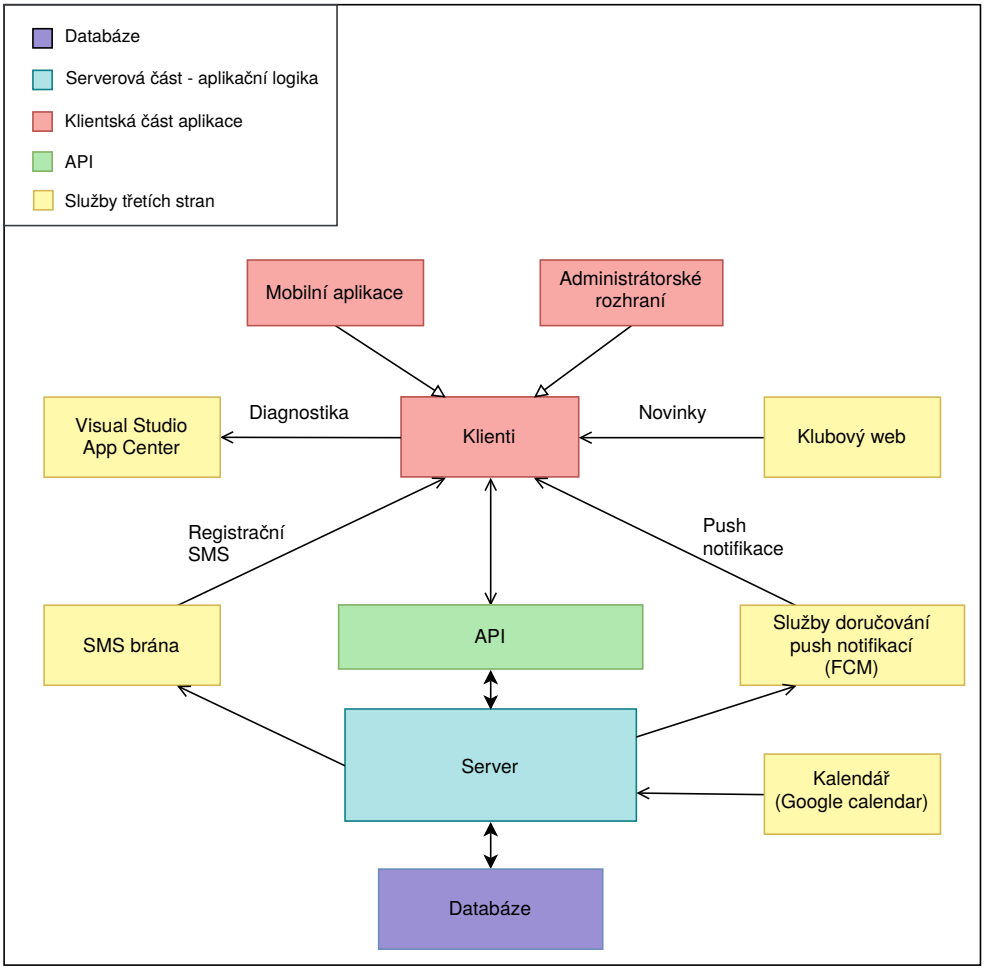
3. Iterace - doplnění funkcí



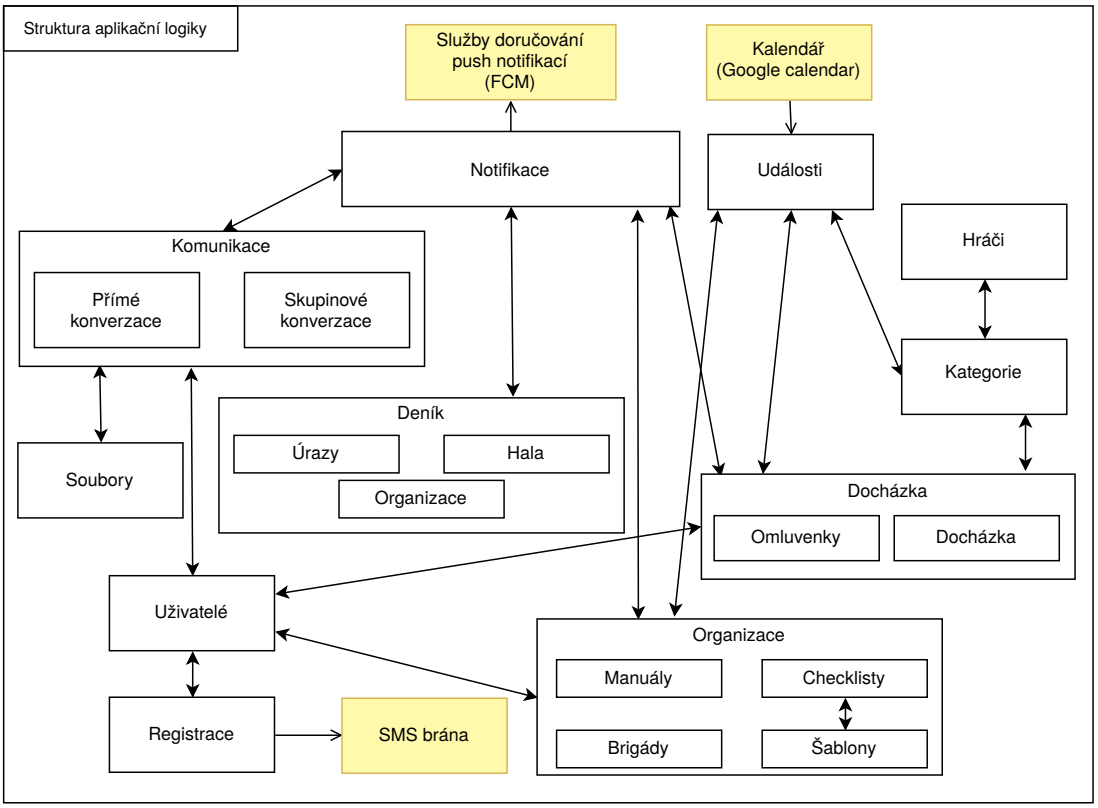
Obrázek A.3: Diagram případů užití v třetí iteraci

Příloha B

Diagram struktury systému



Obrázek B.1: Diagram struktury systému včetně externích služeb (administrátorské rozhraní v tomto diagramu vyjadřuje pouze možné budoucí rozšíření)



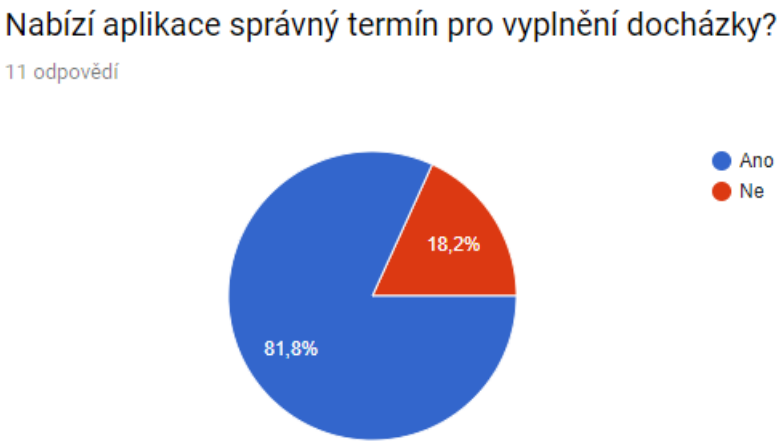
Obrázek B.2: Zjednodušený diagram aplikační logiky serveru

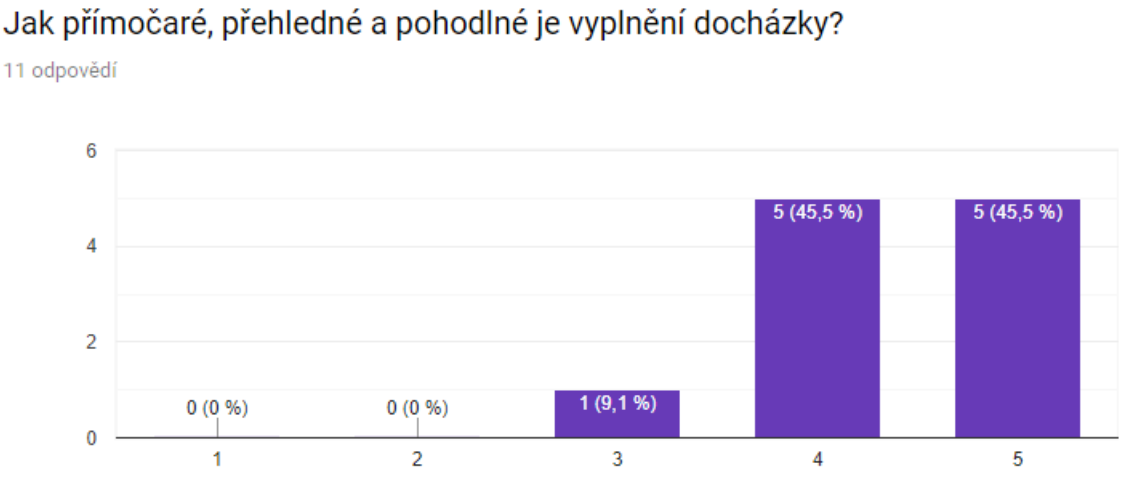
Příloha C

Formulář uživatelských testů

Níže následuje přehled kvantitativních otázek použitých při uživatelském testování aplikace včetně znázornění sesbíraných odpovědí na každou z nich.

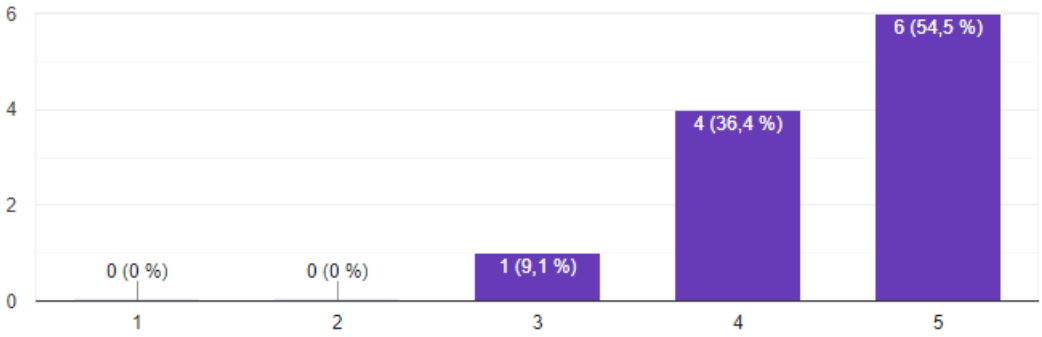
C.1 Testování sekce Docházka





Jak intuitivní je pro Vás celá sekce Docházka? (zadávání, prohlížení, omluvenky,...)

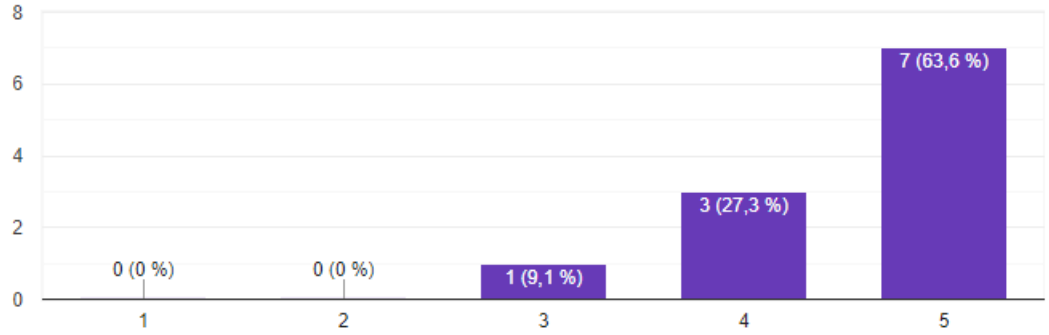
11 odpovědí



C.2 Testování formulářů (zaměřeno na Deník)

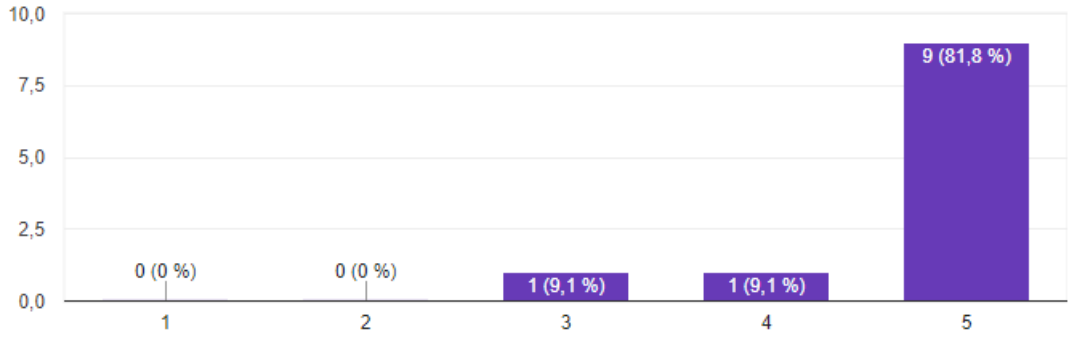
Jak hodnotíte přehlednost a intuitivnost formuláře? Je jasné, kam patří jaká informace?

11 odpovědí



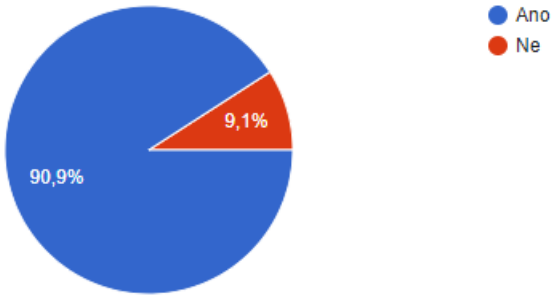
Jak snadné a pohodlné je vyplnění formuláře?

11 odpovědí



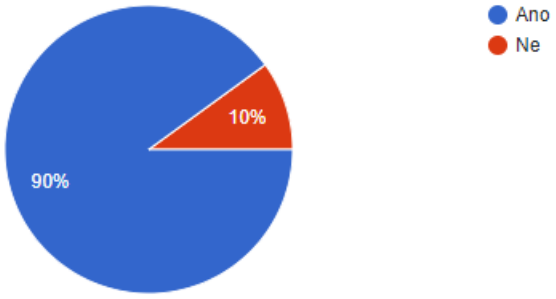
Je jednoznačné, která pole jsou povinná?

11 odpovědí



Je dobře patrné, která pole jsou špatně vyplněná a proč tomu tak je?

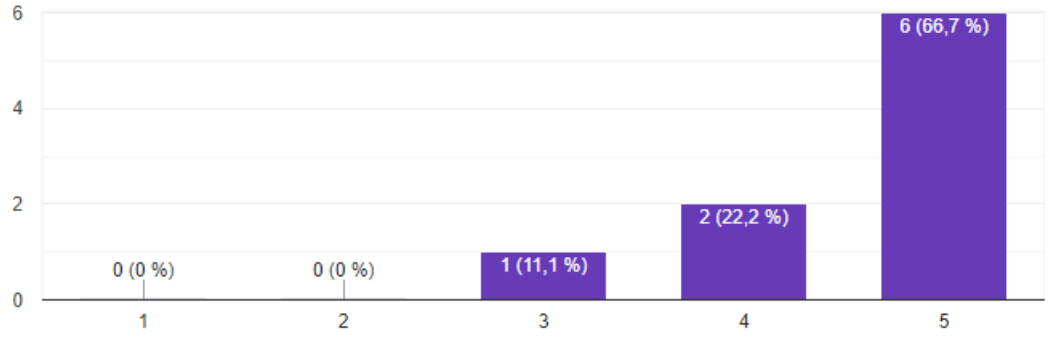
10 odpovědí



C.3 Testování sekce Brigády

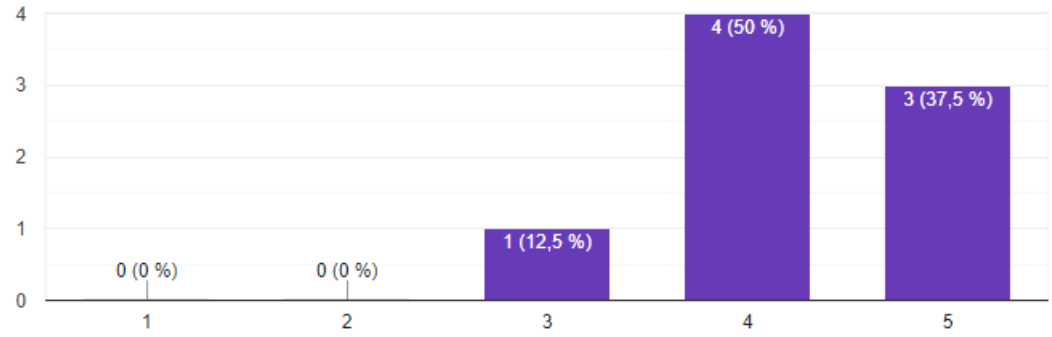
Jak přehledný je seznam brigád a je jasné, na jaký den a jakou pozici se hlásíte?

9 odpovědí



Jak přehledný je seznam Vašich přihlášených brigád a je jasné, kdy a co Vás čeká?

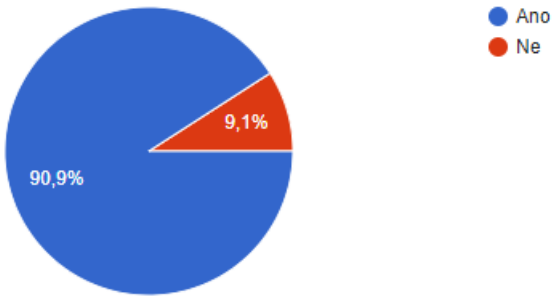
8 odpovědí



C.4 Testování sekce Komunikace

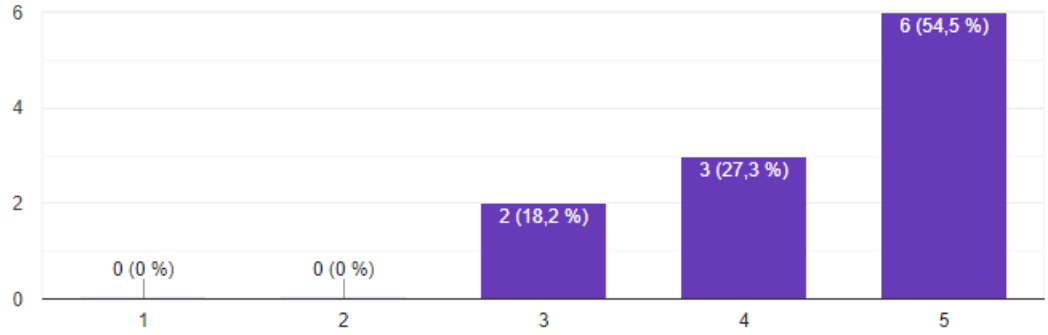
Jsou novinky přehledné a zobrazují se ty, které jsou důležité pro Vaši kategorii (či více kategorií)?

11 odpovědí



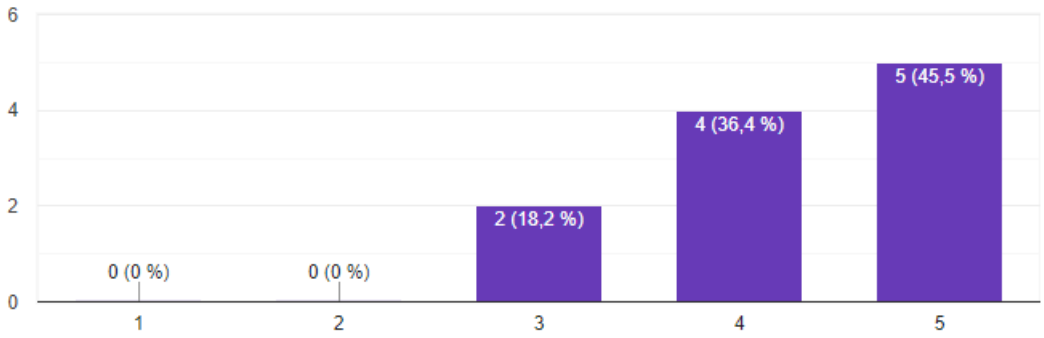
Jak přehledný je seznam kontaktů (včetně odlišení uživatelů aplikace od ostatních) a vyhledávání v něm?

11 odpovědí



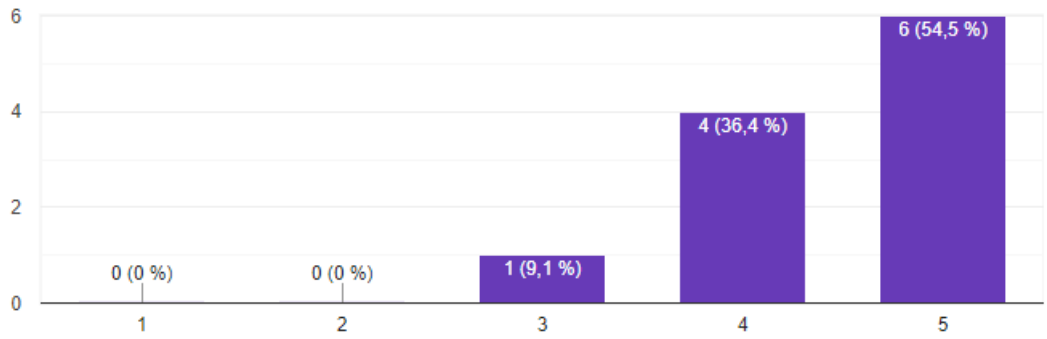
Jak přehledná a snadno použitelná je pro Vás sekce přímých zpráv?

11 odpovědí



Jak přehledná a snadno použitelná je pro Vás sekce skupinových konverzací?

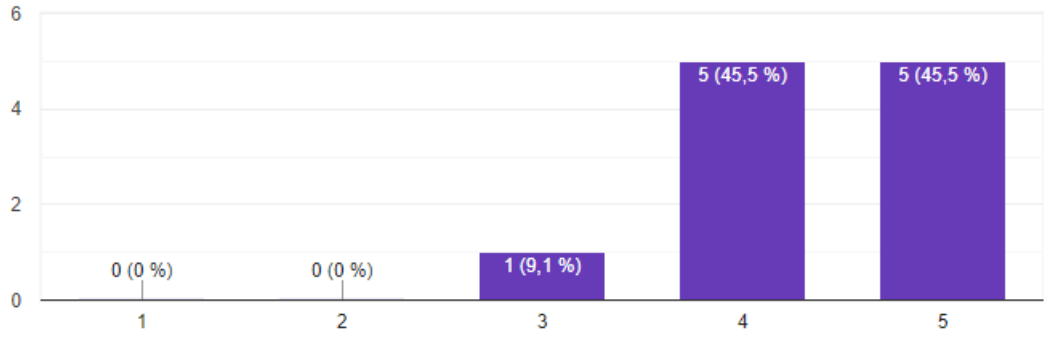
11 odpovědí



C.5 Zjištění celkové zkušenosti s používáním

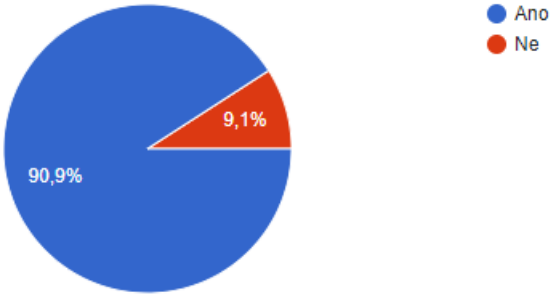
Jak přehledná a intuitivní je pro Vás navigace v aplikaci? (hlavní menu v dolní části, záložky v horní části, dlaždice na stránce s funkcemi apod.)

11 odpovědí



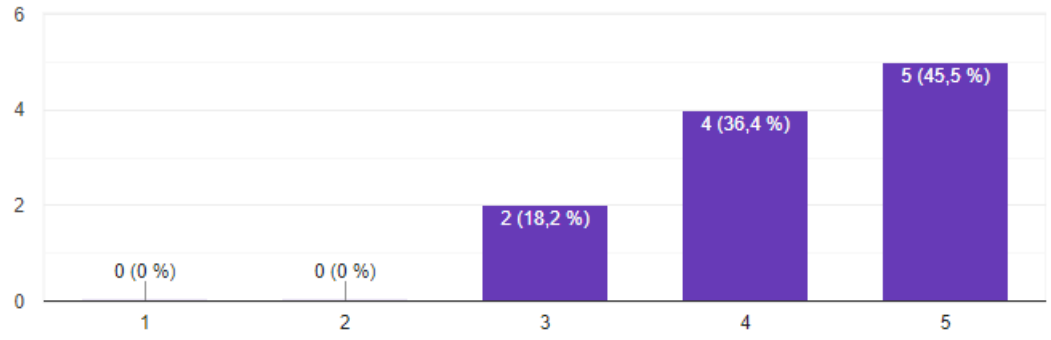
Je vždy jasné, v jaké části aplikace se nacházíte a jakou funkci právě vykonáváte?

11 odpovědí



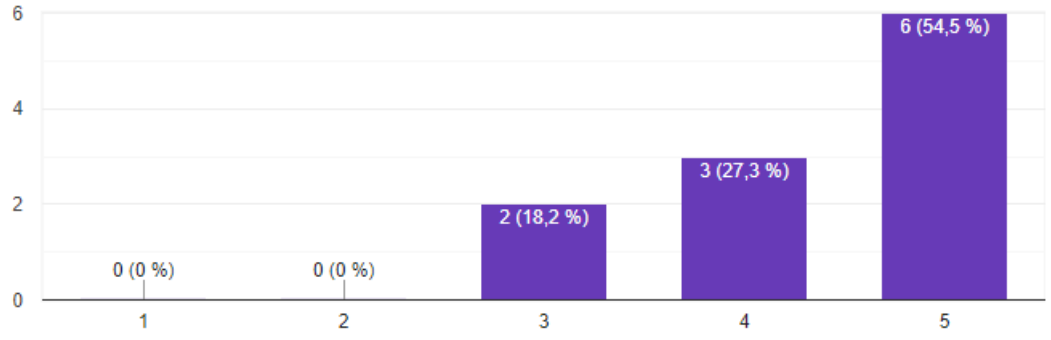
Jak dobře použitelná je pro Vás aplikace při běžném chodu oddílu?

11 odpovědí



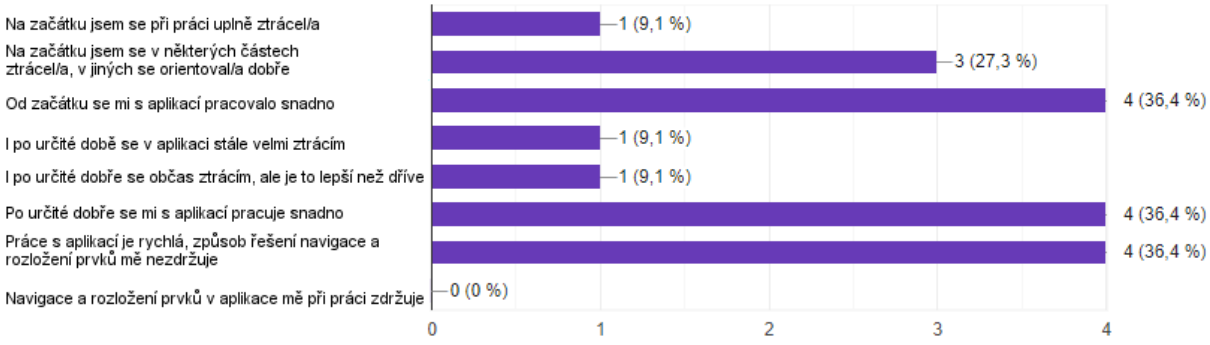
Jak snadné a intuitivní je pro Vás ovládání aplikace?

11 odpovědí



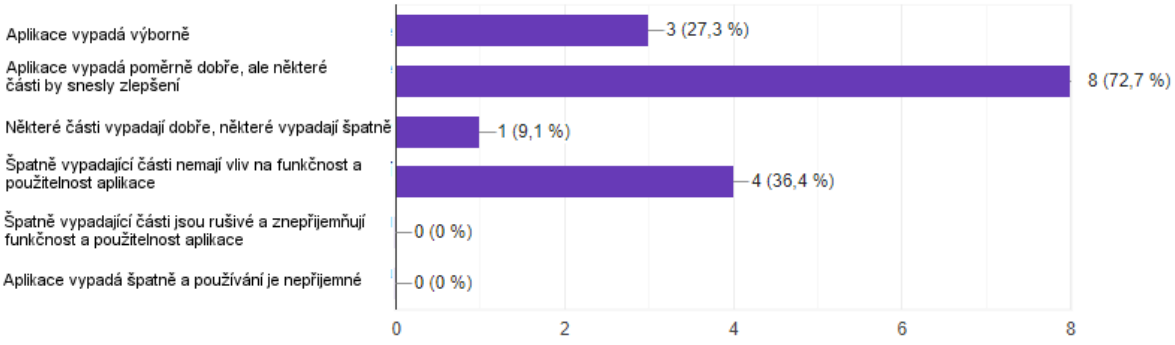
Které z následujících možností se hodí pro popsání Vaší práce s aplikací? (možno více odpovědí)

11 odpovědí



Jak hodnotíte vizuální vzhled aplikace? (možno více odpovědí)

11 odpovědí



Příloha D

Obsah přiloženého DVD

- `/bin/` — spustitelné soubory
  - `app/*` — klientská aplikace pro *Android*
  - `server/*` — serverová aplikace
- `/config/` — konfigurace serverové aplikace
  - `additionalsettings.json` — konfigurační soubor serveru s údaji pro testování
  - `additionalsettings_empty.json` — konfigurační soubor serveru bez údajů
  - `JuvenisApp.db` — *SQLite* databáze s testovacími daty
- `/src/` — zdrojové kódy
  - `app/*` — klientská část
  - `server/*` — serverová část
- `/text/` — technická zpráva diplomové práce
  - `src/*` — zdrojové soubory technické zprávy
  - `xholcm04-aplikace-vedeni-sportovniho-oddilu-tisk.pdf` — verze pro tisk
  - `xholcm04-aplikace-vedeni-sportovniho-oddilu-wis.pdf` — elektronická verze
- `/poster.pdf` — plakát s výsledky práce
- `/README.txt` — manuál k překladu a spuštění obou částí aplikace