Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Systém pro správu účastníků a bezčipový záznam výsledků na běžeckých závodech ovladatelný jednou osobou

Daniel Malý

Vedoucí práce: Ing. Petr Špaček, Ph.D.

11. května 2015

Poděkování

Děkuji vedoucímu práce Ing. Petru Špačkovi, Ph.D., který se ujal mého nápadu a svými radami výrazně pomohl jeho realizaci. Dále děkuji své přítelkyni a rodině za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Malý, Daniel. *Systém pro správu účastníků a bezčipový záznam výsledků na běžeckých závodech ovladatelný jednou osobou.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá tvorbou informačního systému, který usnadní organizátorům malých regionálních běžeckých závodů proces vypisování závodu, registrace účastníků a záznamu výsledků. Zvláště je příhlíženo k technologickým, finančním a personálním omezením, kterým tito organizátoři čelí. Problém těchto omezení je řešen využitím mobilní aplikace pro sběr cílových časů a jednoduché desktopové aplikace pro administraci závodu. Dále je vyvinuta webová aplikace umožňující předběžnou registraci běžců na závod. Součástí práce je také analýza stávajících postupů a komerčních i volně dostupných alternativních řešení.

Klíčová slova informační systém, běžecký závod, registrace účastníků, výsledky závodu, časomíra, ruční měření času, mobilní aplikace

Abstract

This work attempts to design and implement a software solution to facilitate the process of advertising, registering participants and recording results in small regional long-distance running events. Special consideration is given to the technological and financial limitations the organizers of such events must face. To satisfy these constraints, a mobile application is used for race timing,

a simple desktop application is developed to handle race administration and a web application is created to advertise races and enable provisional registration. The paper also includes an analysis of established practice as well as a study of commercial and freely available alternatives.

Keywords information system, running race, race registration, race results, finish times, manual timing, mobile application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Po celé České republice se ročně konají stovky drobných závodů ve vytrvalostním běhu na tratě dlouhé od 5 km do 20 km. Tyto závody mívají většinou 50 až 150 účastníků a startovné se pohybuje okolo 80 Kč. Závody organizují místní běžečtí nadšenci a jediná záštita, které se jim většinou dostane, je povolení starosty. Jednoduchým počtem vyplývá, že technologické zázemí na těchto závodech je velmi slabé či neexistující. Systémy pro správu sportovních závodů jsou sice dostupné a často nabízejí sofistikovaný čipový systém pro měření času, takový luxus si však organizátoři těchto malých závodů nemohou dovolit.

Není jak se na závod přihlásit předem, aby organizátoři mohli odhadnout očekávanou účast. Registrace probíhá vyplňováním tabulky v běžném tabulkovém editoru a ručním rozřazováním do kategorií. Záznam výsledků je obtížný a je vždy potřeba minimálně dvou lidí; jeden, který zapisuje dobíhající čísla, a druhý, který zaznamenává časy. Často vznikají chyby a následné zpracování výsledků může trvat hodiny, neboť se opět ručně přepisuje do tabulkového editoru. Tento poslední bod je pro samotné závodníky největším problémem.

Cílem této práce je vytvořit nový systém, který umožní jednoduchou správu registrovaných účastníků a rychlejší manuální přepis naměřených časů. Dále by měl umožnit jediné osobě na cílové čáře pořídit přesný záznam výsledků pomocí obyčejného chytrého telefonu. Budu se snažit zcela eliminovat potřebu manuálního přepisu výsledků přenosem dat od cílové čáry rovnou do hlavní části systému. Dále rozšířím systém o možnost správy závodu jako takového. To bude zahrnovat předběžnou registraci účastníků přes webovou aplikaci, vyhodnocování souhrnných výsledků napříč celým seriálem závodů a přehledné zobrazování výsledků proběhlých závodů na internetu. Navrhnu také řešení, jak můžeme pomocí druhé osoby s chytrým telefonem přinést divákům informace o mezičasech z tratě.

Kapitola 1

Analýza problému

V této části popíšu, jak přesně závody, kterým je náš systém určen, probíhají. To nám umožní lépe pochopit požadavky na systém. Vycházím z vlastní zkušenosti s organizováním jednoho takového závodu [1] a s diváckou účastí na několika dalších.

1.1 Základní pojmy

Vypisování závodu Jedná se o umístění propozic závodu na veřejně dostupné místo, v tomto případě internet, kde se o konání závodu mohou účastníci dozvědět. To může zahrnovat i nějakou jednoduchou formu propagace.

Předběžná a fyzická registrace na závod Běžec, který se chce zúčastnit závodu, se musí dostavit nějakou dobu před začátkem závodu na místo konání, dát organizátorům vědět, že hodlá běžet, a zaplatit startovné. Tento proces označuji v této práci jako „fyzická registrace“, „zápis“ nebo pouze „registrace“. Fyzická registrace se liší od „předběžné registrace“, což je pouze nezávazná informace podaná organizátorům v předstihu, že se běžec hodlá závodu účastnit.

Věková kategorie Na většině běžeckých závodů jsou účastníci rozděleni do kategorií dle svého věku a pohlaví, neboť existují výkonnostní rozdíly mezi staršími a mladšími i mezi ženami a muži. Nejčastěji se na závodech oceňují vítězové nejen celého závodu, ale také jednotlivých kategorií.

Startovní číslo Při fyzické registraci je každému závodníkovi přiděleno unikátní číslo. Štítek s tímto číslem si závodník připne na úbor, aby ho mohli organizátoři snadno identifikovat na cílové čáře. Seznamu zapsaných lidí seřazeném dle startovních čísel říkáme „startovní listina“.

Cílový čas Tento pojem označuje dobu, za kterou závodník uběhl celou trať závodu od startu až po cíl. Cílovému času se říká také „výkon“. Cílový čas se liší od „mezičasu“, který představuje dobu, za kterou závodník uběhl část tratě od startu po nějaký předem stanovený bod na trati.

1.2 Popis procesu pořádání závodu

Pro tento případ vycházím z toho, že se pořadatel pro evidenci závodníků rozhodl použít program Microsoft Excel, jelikož je to nejčastější praxe. Jak uvidíme v kapitole 2, existuje spousta jiných možných scénářů. Zejména u větších a známějších závodů, které mají dostatek sponzorů, bude proces vypadat odlišně. Těmito závody se však v této práci nezabývám.

Závody bývají vypsány na několik měsíců dopředu jednak na stránkách běžeckých klubů, které je pořádají, jednak na webech věnujících se vytrvalostnímu běhu obecně [2, 3]. Téměř výhradně se konají o víkendu a trvají 3 - 5 hodin. Délka trati bývá velmi různá, pohybuje se většinou od 5 km do 15 km. Startovné bývá u menších závodů i méně než 100 Kč, peněžitá cena pro vítěze, pokud nějaká je, bývá v řádu stovek korun. Jsou běžci, kteří jezdí na tyto závody pravidelně po celé republice každý víkend, jiní se účastní pouze závodů v okolí jejich bydliště. Účastníci bývají všech věkových kategorií – na trati se setkávají devítiletí se sedmdesátníky.

Předběžná registrace neprobíhá žádná; kolik účastníků přijede, se ukáže až na místě. Sjíždět se začínají hodinu před startem závodu a od této chvíle začíná registrace účastníků a přidělování startovních čísel. Závodníci buď vyplní krátký registrační formulář se svým jménem, rokem narození a běžeckým oddílem, pokud nějaký mají, nebo tyto údaje nadiktují přímo organizátorům. Zaplatí startovné a na oplátku obdrží startovní číslo, které si připnou k úboru. Pořadatel údaje spolu se startovním číslem zapíše do tabulky. Pomocí roku narození se ručně vypočte věková kategorie účastníka. U mužů bývá obvykle pět věkových kategorií, u žen tři. Po skončení registrace musí pořadatel vytvořit startovní listiny v Excelu dle jednotlivých kategorií. Do nich pak bude vnášet zaznamenané cílové časy.

Jakmile jsou běžci připraveni a registrace dokončena, pořadatel odstartuje závod a sepne časomíru. Tou může být jakýkoli přístroj s funkcí stopek. Pak už jen čeká na doběhnutí nejrychlejších běžců.

Na zaznamenávání cílových časů je potřeba dvou osob. První zapisuje cílové časy, druhá zapisuje startovní čísla v pořadí, v jakém dobíhají. Mezi jednotlivými účastníky bývá většinou několik vteřin rozestup, může se však stát, že přiběhne shluk běžců najednou a v takových případech by osamocený časoměřič situaci nemusel zvládnout. Udržet krok s dobíhajícími účastníky je nesmírně důležité, neboť neprobíhá žádná kontrola, zda sedí časy a čísla. Změřené časy jsou samozřejmě přesné jen na jednotky sekund, jelikož záznam času provádí člověk, nicméně u vytrvalostního běhu to není takový problém.

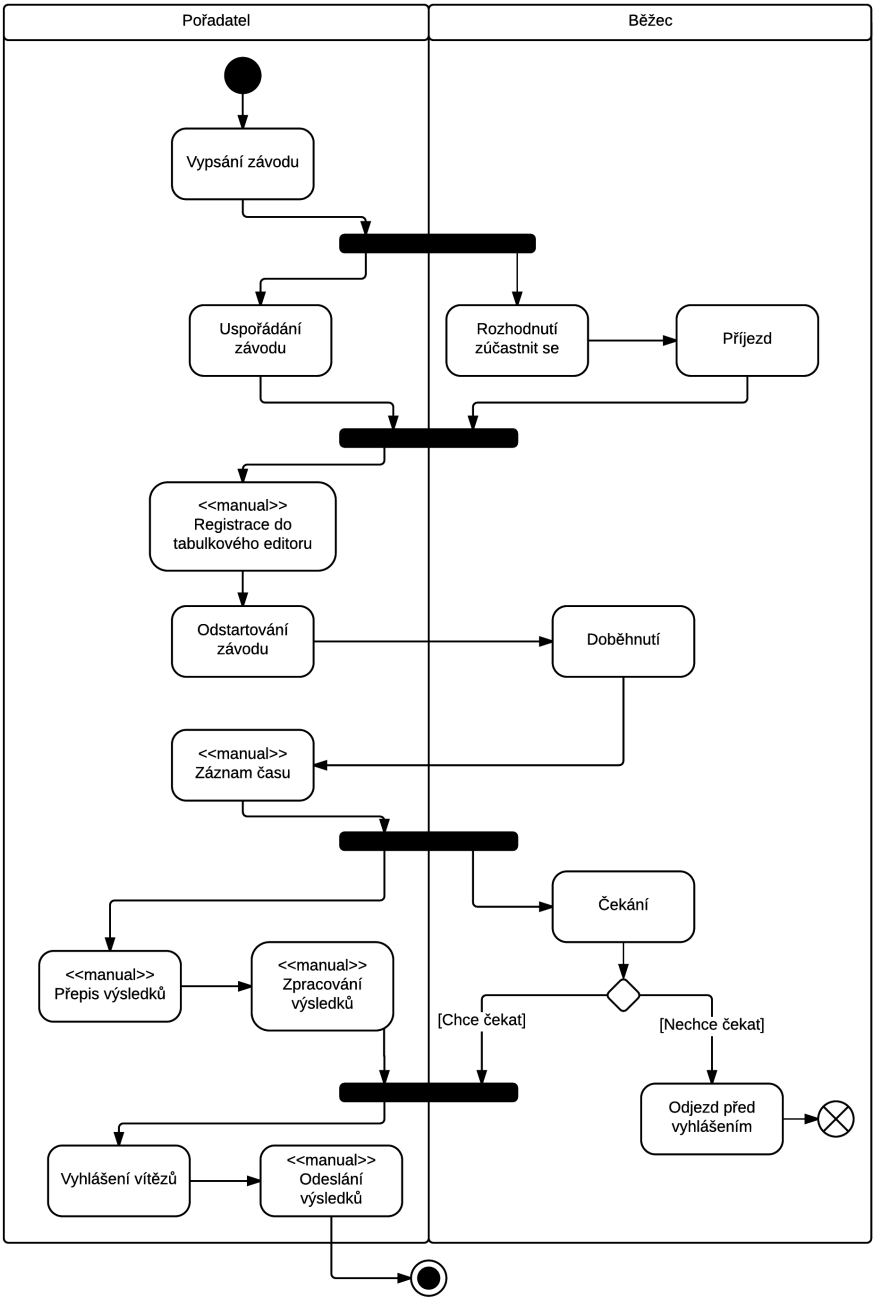
Jakmile jsou časy zaznamenány, musí je pořadatel ručně zanést do Excelových tabulek. Spoléhá při tom na to, že pořadí časů a pořadí čísel odpovídá. Poté musí sestavit výsledkové listiny pro celý závod i pro jednotlivé věkové kategorie. Všechny úkony lze s MS Excel pomocí řazení, kopírování a vkládání vykonat poměrně rychle, ne všichni organizátoři jsou však v používání programu dostatečně zběhlí, a tak se zpracováváním výsledků po závodě dá strávit několik hodin. Přepisování již jednou ručně napsaných výsledků se však nikdo nevyhne.

Jakmile jsou výsledky zpracovány, což je většinou hodinu až dvě po doběhnutí posledního účastníka, nastává vyhlášení vítězů, předání cen a odjezd účastníků. Po skončení závodu musí ještě pořadatel prostřednictvím e-mailu odeslat výsledkové listiny pořadatelům seriálu nebo rodičovskému oddílu, který je pak zveřejní na svých stránkách.

Celý proces je znázorněn na UML diagramu aktivit v obrázku 1.1. Na něm je vidět, že po doběhnutí účastníků čeká organizátora úctyhodné množství úkonů, což je hlavní příčinou dlouhých prostojů před vyhlášením.

1.3 Nevýhody stávajícího postupu

- Existují sice centralizované weby pro vypisování závodů, těch je však několik a pořadatel musí na každém z nich vypsat závod zvlášť. Výsledky pak jsou většinou uveřejněny jen na jednom místě, nikoli všude tam, kde byl závod vypsán.
- Pořadatel nemůže předem ani přibližně vědět, kolik mu na závod přijede účastníků. Pokud chce zavést předběžnou registraci, omezit kapacitu závodu či zřídit čekací listinu, musí si na to implementovat vlastní mechanismus.
- Není možné z tratě přinášet divákům mezičasy.
- Záznam výsledků je nespolehlivý a náchylný k chybám. Navíc je zapotřebí dvou osob.
- Vyhodnocování výsledků trvá kvůli ručnímu přepisování příliš dlouho a účastníkům často dojde trpělivost a odjedou ještě před vyhlášením.
- Pořadatel závodu musí výsledky uveřejnit na internetu ručně. Pro spočítání souhrnných výsledků seriálů závodů se musí zpracovat výsledky z různorodých výsledkových listin, což je také časově náročné.



Obrázek 1.1: Diagram aktivit procesu pořádání závodu

Kapitola 2

Rešerše existujících řešení

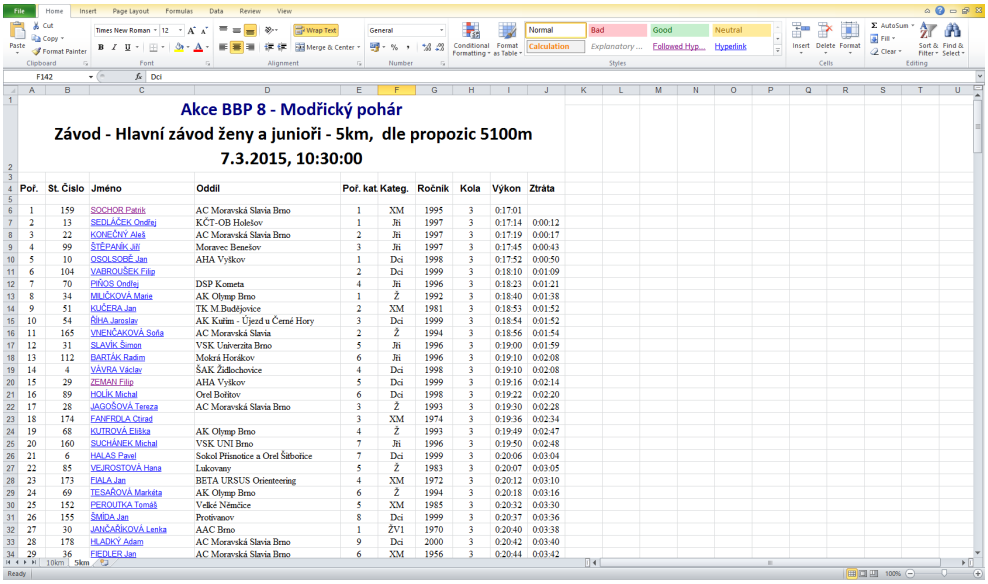
Technologické zázemí je na menších běžeckých závodech naprosto minimální. Především je to dáno nedostatkem financí, ale také neinformovaností pořadatelů o existenci jiných možností. V této sekci se pokusím některá volně dostupná řešení pro správu závodů analyzovat. Podívám se i na některá komplexnější komerční řešení. Ta budou zcela jistě pokročilejší než systém vyvíjený v této práci, jelikož naše cílová skupina je odlišná. Bude však užitečné zjistit, co všechnno můžou pořadatelé od svého softwaru na správu sportovních událostí chtít.

2.1 Microsoft Excel a jiné tabulkové editory

Microsoft Excel není specializovaný systém na správu závodů. Dalo by se argumentovat, že jakýkoli informační systém lze nasimulovat v tabulkovém editoru, má-li člověk dostatek času a zručnosti. Je třeba ho však do tohoto srovnání zařadit, neboť je na nesponzorovaných závodech zdaleka nejpoužívanějším řešením. Obrázek 2.1 ukazuje typickou výsledkovou listinu sestavenou v Excelu.

Hlavní nevýhoda tohoto řešení je pracnost zadávání dat, které není nijak optimalizováno pro tento případ užití. Optimalizovat by se sice dalo pomocí programovatelných funkcí editoru, použití takových funkcí je však mimo schopnosti většiny uživatelů, navíc je časově náročné. Zpracování dat musí také proběhnout ručně.

Naopak jednou výhodou použití obecného tabulkového editoru je prakticky neomezená možnost přizpůsobení ukládaných informací. Potřebujeme-li místo podle cílového času řadit účastníky dle uběhnuté vzdálenosti nebo k nim navíc ukládat údaje o národnosti, tabulkový editor nám umožňuje tyto požadavky splnit. Pro specializovaný systém, který tuto funkcionalitu nepodporuje, to může znamenat rozsudek nepoužitelnosti.



Akce BBP 8 - Modřický pohár

Závod - Hlavní závod ženy a junioři - 5km, dle propozic 5100m

7.3.2015, 10:30:00

Poř.	St. Číslo	Jméno	Oddíl	Poř. kat.	Kateg.	Ročník	Kola	Výkon	Ztráta
1	159	SOCHOR Patrik	AC Moravská Slavia Brno	1	XM	1995	3	0:17:01	
2	13	SEDLÁČEK Ondřej	KČT-OB Holešov	1	Jn	1997	3	0:17:14	0:00:12
3	22	KONEČNÝ Aleš	AC Moravská Slavia Brno	2	Jn	1997	3	0:17:19	0:00:17
4	99	ŠTĚPANÍK Jiří	Moravec Benešov	3	Jn	1997	3	0:17:45	0:00:43
5	10	OSOLSOBĚ Jan	AHA Vyškov	1	Dci	1998	3	0:17:52	0:00:50
6	104	VABROUŠEK Filip		2	Dci	1999	3	0:18:10	0:01:09
7	70	PIŇOS Ondřej	DSP Kometa	4	Jn	1996	3	0:18:23	0:01:21
8	34	MLČKOVÁ Marie	AK Olymp Brno	1	Ž	1992	3	0:18:40	0:01:38
9	51	KUČERA Jan	TK M.Budějovice	2	XM	1981	3	0:18:53	0:01:52
10	54	ŘÍHA Jaroslav	AK Kuřim - Újezd u Černé Hory	3	Dci	1999	3	0:18:54	0:01:52
11	165	VNENČÁKOVÁ Soňa	AC Moravská Slavia	2	Ž	1994	3	0:18:56	0:01:54
12	31	SLAVÍK Šimon	VSK Univerzita Brno	5	Jn	1996	3	0:19:00	0:01:59
13	112	BARTÁK Radim	Mokrá Horákov	6	Jn	1996	3	0:19:10	0:02:08
14	4	VÁVRA Václav	ŠAK Židlochovice	4	Dci	1998	3	0:19:10	0:02:08
15	29	ZEMAN Filip	AHA Vyškov	5	Dci	1999	3	0:19:16	0:02:14
16	89	HOLÍK Michal	Orel Bořitov	6	Dci	1998	3	0:19:22	0:02:20
17	28	JAGOŠOVÁ Tereza	AC Moravská Slavia Brno	3	Ž	1993	3	0:19:30	0:02:28
18	174	FANFRDLA Ctirad		3	XM	1974	3	0:19:36	0:02:34
19	68	KUTROVÁ Eliška	AK Olymp Brno	4	Ž	1993	3	0:19:49	0:02:47
20	160	SUCHÁNEK Michal	VSK UNI Brno	7	Jn	1996	3	0:19:50	0:02:48
21	6	HALAS Pavel	Sokol Přísnotice a Orel Šitbořice	7	Dci	1999	3	0:20:06	0:03:04
22	85	VEJROSTOVÁ Hana	Lukovany	5	Ž	1983	3	0:20:07	0:03:05
23	173	FIALA Jan	BETA URSUS Orienteering	4	XM	1972	3	0:20:12	0:03:10
24	69	TESAŘOVÁ Markéta	AK Olymp Brno	6	Ž	1994	3	0:20:18	0:03:16
25	152	PEROUTKA Tomáš	Velké Němčice	5	XM	1985	3	0:20:32	0:03:30
26	155	ŠMÍDA Jan	Protivanov	8	Dci	1999	3	0:20:37	0:03:36
27	30	JANČAŘÍKOVÁ Lenka	AAC Brno	1	ŽV1	1970	3	0:20:40	0:03:38
28	178	HLADKÝ Adam	AC Moravská Slavia Brno	9	Dci	2000	3	0:20:42	0:03:40
29	36	FIEDLER Jan	AC Moravská Slavia Brno	6	XM	1956	3	0:20:44	0:03:42

Obrázek 2.1: Výsledková listina v programu Microsoft Excel

2.2 Placené systémy

Větší závody, jejichž konání nepřehlédne ani člověk, který se o tento druh sportovních událostí aktivně nezajímá, využívají pro své mediální, marketingové a technologické zázemí prostředky sponzorů [4]. Z těchto prostředků pak mohou financovat specializované systémy vyvinuté na míru pro organizaci sportovních událostí. V této sekci analyzuji dva takovéto systémy, jeden tuzemský a jeden zahraniční a shrnu, co organizátorům závodů nabízí.

2.2.1 Atletika UNI

Atletika UNI je brněnská firma, která se zabývá „časomírou, výsledkovým servisem na různých malých i velkých sportovních akcích a také přímo pořádáním sportovních akcí“ [5]. Nejedná se tedy čistě o softwarové řešení, ale o celý servis technického zázemí závodu. Středobodem tohoto zázemí je automatizované měření času buď čipovou technologií, nebo cílovou kamerou. Ke zpracování, zobrazování a odesílání výsledků používají vlastní softwarový produkt nebo program Atletická kancelář. Hlavními výhodami využití služeb této nebo podobné firmy je přesnost měření, možnost okamžitě informovat komentátora, diváky a média o naměřených výsledcích a v neposlední řadě také fakt, že se organizátor závodu může věnovat jiným činnostem s vědomím, že všechny technologické úkony za něj dělá někdo jiný. Tato možnost je samozřejmě preferována jak účastníky, tak organizátory závodu. Při dostatečných finančních prostředcích neexistuje důvod nevyužít služeb specializované firmy.

Kromě měření času se Atletika UNI stará o online registraci účastníků a registraci na místě. Účastníkům seriálů závodů jsou vydávány plastové karty s čárovým kódem, kterými se prokazují na dalších závodech seriálu. To umožňuje urychlit registraci a zároveň identifikuje závodníky, což usnadňuje párování jejich výsledků napříč celým seriálem.

Na webových stránkách firmy jsou uveřejněny plánované i proběhlé závody s výsledky ke stažení. Z těchto informací je patrné, jak jsou závody v systému organizovány. Jednotlivé „akce“ pod sebou sdružují jeden či více závodů. Pokud jich závodů v jedné akci více, jsou rozděleny podle věkových kategorií, které obsahují, a také podle délky tratě [6]. Jeden závod však může umožňovat běh více kategorií najednou. Jednotlivé závody mají starty v různý čas a výsledkové listiny jsou uvedeny u každého zvlášť. V rámci jednoho závodu lze filtrovat výsledky podle věkové kategorie.

K závodům jsou dále uváděny propozice ve formátu PDF spolu s informacemi o místě konání, pořadateli závodu, rozhodčích, zpracovatelích výsledků a výší startovného.

2.2.2 ActiveWorks

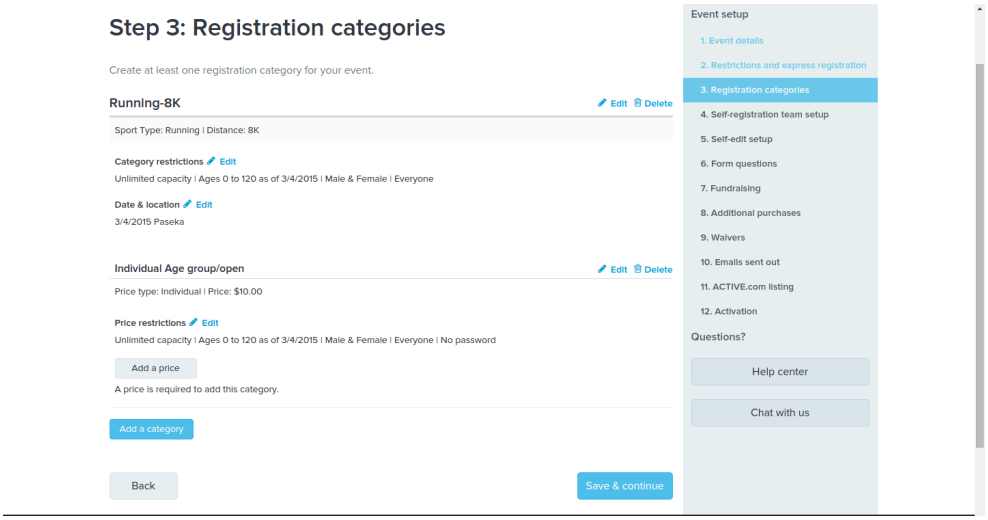
Americká společnost Active Network nabízí ActiveWorks, software pro správu účastníků na vytrvalostních závodech všech druhů a velikostí. Na rozdíl od služeb společností jako Atletika UNI je toto řešení zaměřeno čistě na vypisování závodů a registraci účastníků. Součástí systému tedy není žádné řešení časomíry a samotného záznamu výkonů [7].

I přes toto užší zaměření nabízí ActiveWorks obdivuhodně mnoho nástrojů pro usnadnění práce organizátora. Software se snaží být univerzálně použitelný napříč mnoha sporty, kromě běhu například i pro cyklistiku a triatlon. Jeho hlavním zaměřením je pomoci organizátorovi přilákat na závod co nejvíce účastníků a těmto účastníkům umožnit zaplatit startovné přes internet.

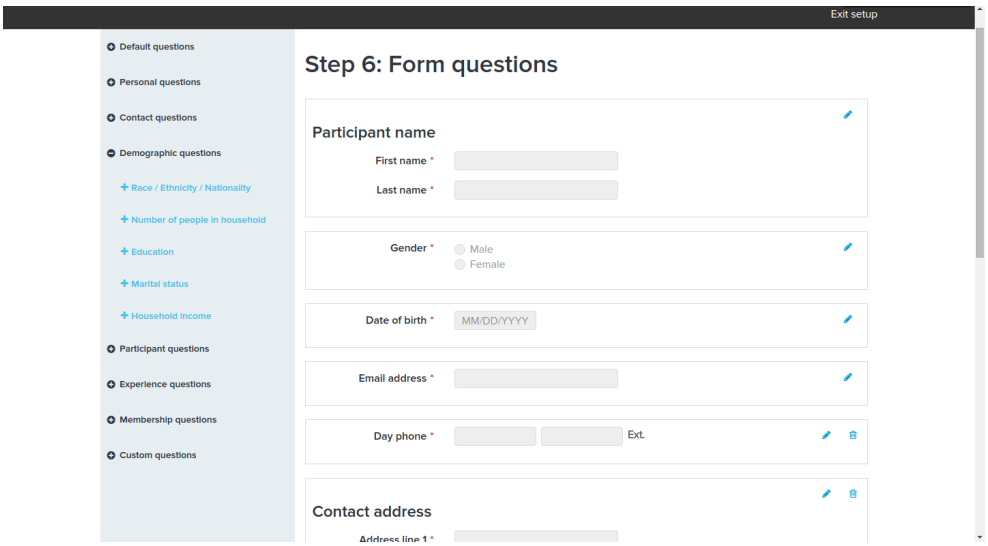
Použití ActiveWorks je zpoplatněno procentuelním poplatkem z každého vybraného startovného. Nejnižší sazba je ovšem $3,25 za jednu registraci, což je na mnoha závodech v ČR více než celé startovné.

ActiveWorks je webová aplikace a začít používat ji může kdokoli, kdo se zaregistruje jako organizátor. To mi umožnilo blíže prostudovat funkce, které systém nabízí. Kompletní seznam funkcí je na stránkách výrobce. Zde popíšu pouze ty nejrelevantnější pro závody, kterými se v této práci zabývám.

Definice závodu Ihned po webové registraci organizátora je zpřístupněna funkce založení závodu. Zakládání závodu má dvanáct kroků, prvním z nichž je zadávání základních informací jako jméno akce, sport, místo a datum konání. Dále musí pořadatel specifikovat rozřazení do kategorií podle věku a pohlaví, což znázorňuje obrázek 2.2. Následuje definování možnosti registrace týmů a změny registračních údajů.



Obrázek 2.2: Specifikace kategorií a startovného v aplikaci ActiveWorks



Obrázek 2.3: Specifikace polí registračního formuláře v aplikaci ActiveWorks

Zajímavá funkce, kterou systém nabízí, je možnost pořadatele si nadefinovat, které údaje budou při registraci po účastnících vyžadovány, viz. obrázek 2.3.

Následující kroky se zabývají fundraisingem a prodejem dárkových předmětu jako například triček. Ty si může účastník koupit předem na webu a dostat je pak při fyzické registraci na závodě. Pořadatel zde může nadefinovat, jaké předměty jsou na prodej a kolik stojí.

V dalším kroku může pořadatel přiložit podmínky, se kterými musí účast-

ník při registraci souhlasit. Poté si může nastavit, jaké e-maily bude registrovaným účastníkům posílat. Na výběr má pobízecí zprávu uživatelům, kteří nedokončili proces registrace, a zprávu o potvrzení registrace. Ta může obsahovat QR kód, kterým se účastník na akci prokáže, a urychlí tím tak finální zapsání do závodu.

Registrace a platba startovného Hlavní funkce celého systému je online registrace účastníků na akci. Tato registrace má závazný charakter, jelikož účastník musí už při této registraci zaplatit startovné. Pro tento účel je k dispozici platební brána, z níž jdou peníze společnosti Active Network. Teprve po skončení závodu je startovné bez manipulačních poplatků předáno organizátorovi.

Kromě startovného lze skrz systém například prodávat propagační materiály a vybírat poplatky za změnu registračních údajů, které si účastníci mohou měnit sami až do nějakého předem definovaného data.

Týmy V systému může pořadatel umožnit nebo dokonce vyžadovat registraci v týmech, ať už pro štafetové závody nebo nějaký jiný druh kooperace. Všechna data o týmech, jako například registrovaní členové a kapitáni týmu, může pořadatel měnit.

Online propagace Active Network nabízí pořadatelům propagaci jejich závodů skrz portál active.com, který má dle stránek provozovatele 2,5 milionů unikátních návštěvníků měsíčně. Tato propagace je v ceně používání systému, pořadatele tedy nestojí nic navíc. Registrace do závodů je navíc napojená na sociální sítě, aby se mohli zápisem do závodu účastníci pochlubit ostatním. Tím získá závod potenciálně další marketing zdarma.

2.2.3 Závěr analýzy placených systémů

V navrhovaném řešení pochopitelně nebude možné napodobit všechny funkce a výhody placených systémů. Chci se však inspirovat webovými rozhraními obou analyzovaných řešení a některé myšlenky přenést i do své implementace. Jsou jimi například zveřejňování propozic vypsaných závodů a následné zpřístupnění všech výsledků dle věkových kategorií. Zajímavá je také možnost organizovat více jednotlivých závodů v rámci jedné „akce“ se sdílenými propozicemi a možnost libovolně definovat věkové kategorie.

Naopak implementací placení startovného přes internet, prodejem propagačních materiálů ani podporou týmů se zabývat nebudu.

2.3 Program fsTimer

Program fsTimer je desktopová aplikace pro správu registrací a záznam časů na běžeckých závodech. Jelikož je volně dostupná ke stažení na internetu [8] a zabývá se jak registrací, tak záznamem výsledků, mnohem více se přibližuje přímé alternativě systému vyvíjeného v rámci této práce. Autoři vyvinuli aplikaci pro použití na závodech, které sami organizují, dá se tedy říct, že přímo řeší jejich požadavky. Z těchto důvodů se fsTimeru budu v této sekci zabývat hlouběji než ostatním systémům.

Aplikace fsTimer je napsána čistě pro desktopové operační systémy (Windows, OS X, Linux). Jedná se o soběstačnou aplikaci, která se nepřipojuje k žádné síti a kromě jiných instancí téže aplikace na různých počítačích nekomunikuje s žádným dalším systémem. Napsána je na platformě Python 2.7 a využívá knihovny PyGTK pro vykreslování uživatelského rozhraní.

Funkce fsTimeru se dají rozdělit do tří oblastí: tvorba nového závodu, registrace a záznam výsledků.

2.3.1 Tvorba závodu v fsTimeru

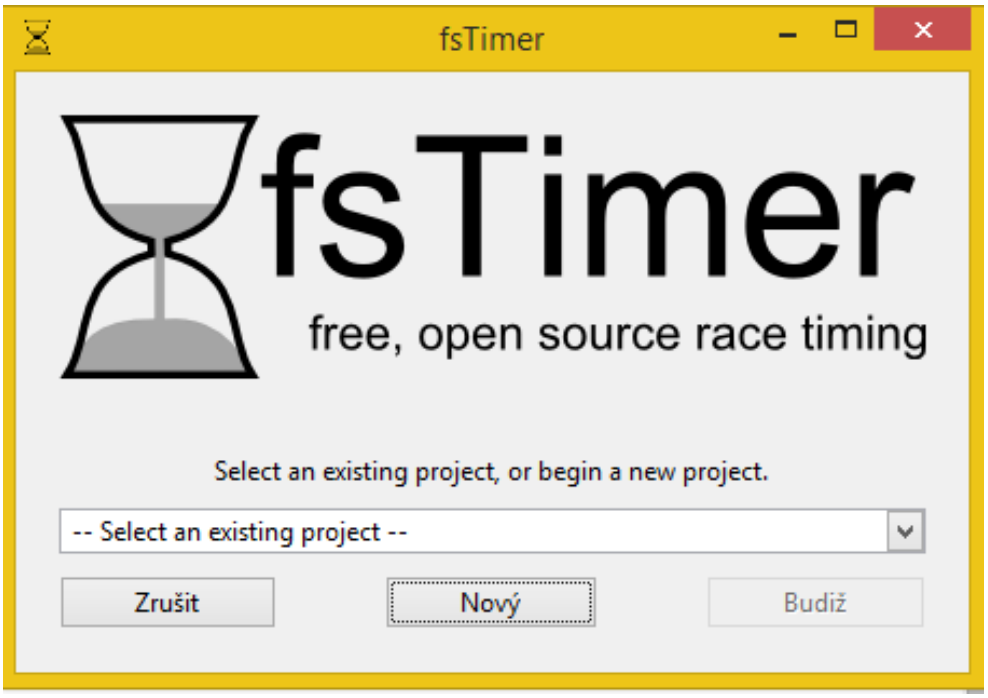
Tvorba závodu je v aplikaci přístupná hned z úvodního dialogu, viz. obrázek 2.4. Probíhá sérií několika formulářů, ve kterých musí pořadatel specifikovat parametry závodu.

Jako první se zobrazí dialog s výzvou pro zadání názvu závodu. V názvu nemohou být žádné mezery ani speciální znaky, s českými názvy bychom tedy měli problém. Následuje výběr formátu závodu: máme na výběr standardní závod a závod s handicapem, kdy jsou někteří běžci od začátku znevýhodněni nějakým časovým postihem. Dále můžeme zaškrtnout, že se závod běží na více kol a můžeme zadat, kolik jich bude. To má vliv na měření času – až do posledního kola jsou všechny naměřené časy pouhými mezičasy.

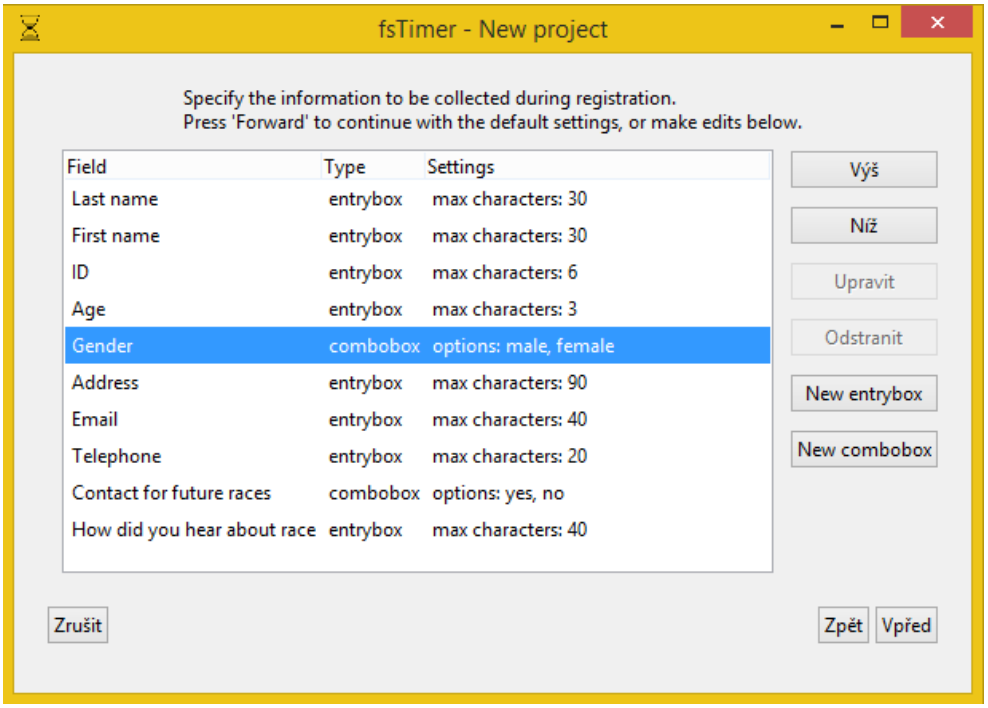
Jakmile je formát závodu specifikován, zobrazí se nám okno, kde si můžeme vybrat či přímo nadefinovat údaje, které chceme o účastnících závodu v rámci registrace sbírat, viz. obrázek 2.5. Máme zde na výběr několik předpřipravených polí, jako je jméno, příjmení, e-mailová adresa, ale také například, zda chce být účastník kontaktován v souvislosti s dalšími pořádanými závody. Pokud nám však předdefinovaná pole nevyhovují, můžeme si zavést vlastní v podobě buď standardního formulářového pole, nebo výběru z několika předem určených hodnot. Vybrané druhy údajů můžeme libovolně přeskupovat do pořadí, v jakém se mají objevovat na formuláři při registraci.

Je třeba podotknout, že registračním formulářem je zde myšleno pouze okno, které se zobrazí na obrazovce registrátora. Tisk papírových formulářů aplikace nepodporuje.

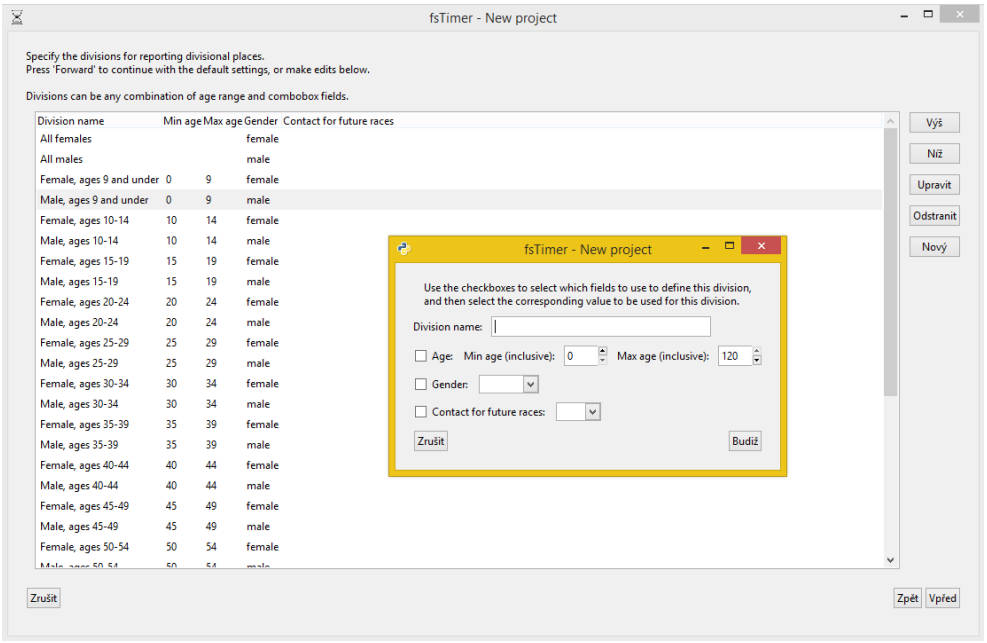
Následuje výběr registračních polí, která mají být sdílená mezi příslušníky jedné rodiny. U registrace nabízí program možnost zapsat člena stejné rodiny jako byl předchozí účastník. To umožňuje pořadatelům neopisovat stejné údaje



Obrázek 2.4: Úvodní dialog aplikace fsTimer



Obrázek 2.5: Definice registračních polí v fsTimeru



Obrázek 2.6: Definice věkových kategorií v fsTimeru

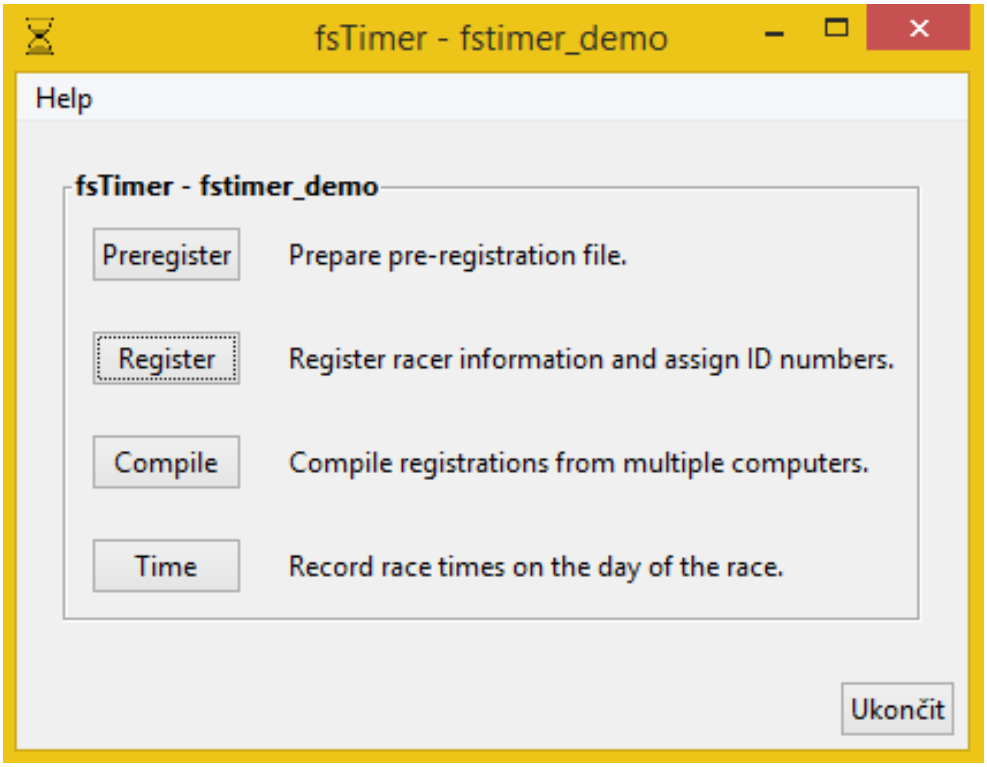
několikrát dokola a zrychlit tak registraci. Zde si tedy uživatel navolí, které údaje jsou mezi rodinnými příslušníky sdílené.

Posledním krokem při tvorbě závodu je definice věkových kategorií. „Kategorie“ v této aplikaci znamená pouze filtr dle věku a pohlaví na výsledkových listinách. Kategorií si může uživatel definovat, kolik chce, a program mu nijak nezabrání v jejich překrývání. Na vlastních závodech je definice kategorie užší – každý účastník má přiřazen právě jednu a z tohoto důvodu není možné, aby se jakýmkoli způsobem překrývaly. Pro účely fsTimeru však volnější pravidla pro kategorie dávají smysl, neboť neexistuje žádná další část systému, která by vyžadovala jednoznačné rozdělení. Navíc díky tomu lze v fsTimeru generovat výsledkové listiny různými způsoby.

2.3.2 Registrace v fsTimeru

Program fsTimer umožňuje importovat údaje nasbírané nějakým druhem předběžné registrace. Sám žádný mechanismus pro online registraci samozřejmě neobsahuje, je však schopný údaje o účastnících načíst ze souboru ve formátu `csv`. Načítání funguje tak, že uživatel vybere csv soubor na disku, program načte první řádek jako hlavičky sloupců a uživatel je pak vyzván k tomu, aby sloupce ze souboru přiřadil registračním údajům, které si zadefinoval během tvorby závodu.

Při importu nemusí být pokryty všechny registrační údaje. Je například vhodné, aby přítomnost startovního čísla indikovala fyzickou registraci v den



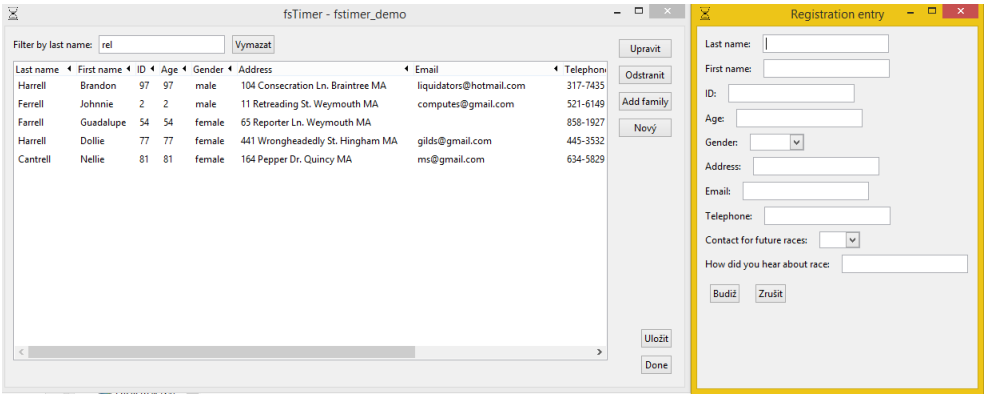
Obrázek 2.7: Hlavní nabídka aplikace fsTimer

závodu, kdy už má účastník zaplacené startovné. Jakmile jsou totiž překlopeny údaje z předběžné registrace, není možné jinak poznat, zda jde o registraci předběžnou nebo fyzickou.

V dalším kroku vytvoří program soubor ve formátu JSON a uloží ho do složky se stejným jménem jako závod. Můžeme se tak přesunout k samotné registraci přes hlavní menu aplikace, viz. obrázek 2.7. Napřed je nutné uvést pořadové číslo daného počítače, to protože fsTimer podporuje registraci na více počítačích a potřebuje identifikovat data z jednotlivých instancí pro účely slučování do jednoho souboru. Zde si lze také vybrat soubor vytvořený importem předběžné registrace. Data v něm obsažená se automaticky stanou prvními registrovanými běžci.

Obrazovka registrace obsahuje tabulku zapsaných účastníků a tlačítka pro registraci dalšího nebo jen rodinného příslušníka posledního registrovaného, viz. obrázek 2.8. Zápis proběhne vyplněním formuláře s poli, která si organizátor zadefinoval při tvorbě závodu.

Kliknutím na tlačítko „uložit“ vytvoří program další soubor ve formátu JSON a uloží ho do složky závodu. Ještě se nejedná o finální seznam závodníků – je napřed potřeba sloučit všechny zápisy ze všech počítačů, na kterých byla registrace prováděna. To lze provést pomocí volby „Compile“ v hlavní



Obrázek 2.8: Registrace účastníků v fsTimeru

nabídce aplikace. Otevře se dialog, kde lze vybrat dílčí soubory a následně je jedním kliknutím sloučit. Dojde-li ke konfliktu ve startovních číslech, je uživatel vyzván k výběru správného záznamu. Výstupem slučování je opět další soubor ve formátu JSON, který slouží jako vyhledávací tabulka při záznamu časů.

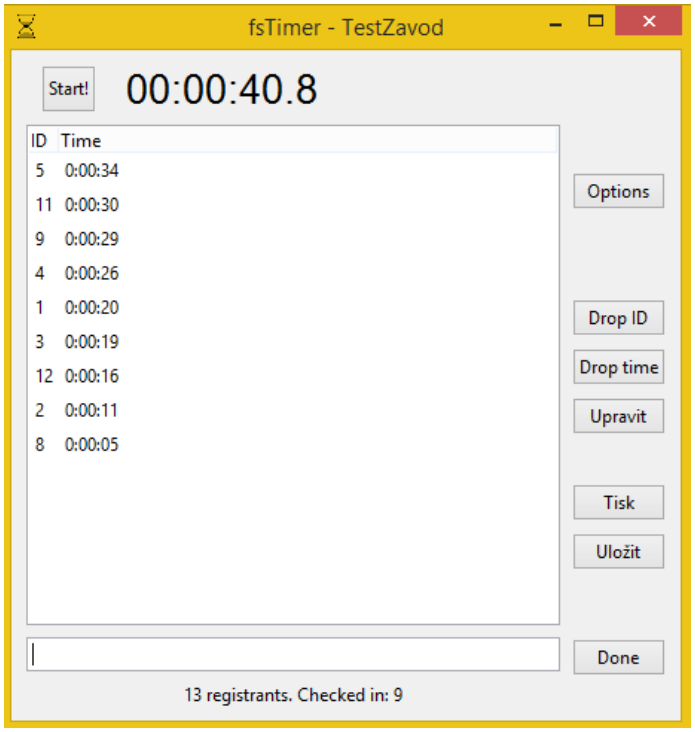
2.3.3 Záznam výsledků v fsTimeru

Záznam výsledků závodu začíná kliknutím na tlačítko „Time“ v hlavní nabídce. Uživateli se otevře dialog, v němž si musí zvolit soubor obsahující vyhledávací tabulku („timing dictionary“) pro záznam časů. Dále si může zvolit klávesu, jejíž zmáčknutí bude představovat proběhnutí závodníka a tzv. nulové startovní číslo. Toto číslo se používá v případě, že zapisovatel nestíhá zaznamenávat čísla přibíhajících závodníků. Musí však do seznamu vložit něco, co programu řekne, že k danému času vloží startovní číslo později, jinak dojde ke ztrátě synchronizace mezi časy a čísly.

Po odkliknutí tohoto dialogu se pořadateli otevře okno jako na obrázku 2.9. Jako první musí uživatel zapnout časomíru přesně v okamžik startu závodu. Po zapnutí časomíry lze zachycovat časy stisknutím klávesy určené v předchozím okně. Napíšeme-li do spodního pole startovní číslo a stiskneme klávesu ENTER, napsané číslo se přiřadí k nejstaršímu zachycenému času, který ještě žádné číslo nemá. Takto postupujeme až do chvíle, kdy všichni účastníci doběhli a jsou přiřazena všechna čísla ke správným časům.

Aplikace také umožňuje časy měnit, mazat a přesouvat. Tlačítko „Uložit“ zapíše aktuální stav zaznamenávání, který lze například po nuceném restartu aplikace znovu načíst. Dále je možné časy uložit do formátu CSV, sloučit naměřené výsledky z jiného počítače anebo exportovat do HTML pomocí tlačítka „Tisk“.

Tisk výsledkových listin spočívá v exportu dvou souborů ve formátu



Obrázek 2.9: Záznam výsledků v fsTimeru

Place	Time	Name	Bib ID	Gender	Age
1	0:00:05	Hipolito Stiefel	8	male	72
2	0:00:11	Eunice Roundtree	2	female	68
3	0:00:16	Roslyn Bland	12	female	53
4	0:00:19	Josephine Miller	3	female	58
5	0:00:20	Paul Richey	1	male	78
6	0:00:26	Nathan Todd	4	male	62
7	0:00:29	Keith Strong	9	male	19
8	0:00:30	Richard Newton	11	male	34
9	0:00:34	Elizabeth Turner	5	female	60

Race timing with fsTimer - free, open source software for race timing. http://fstimer.org

Obrázek 2.10: Výsledková listina z programu fsTimer

HTML. Jeden z nich je souhrnná výsledková listina se všemi účastníky (viz. obrázek 2.10), druhý soubor obsahuje jednu výsledkovou tabulku pro každou kategorii definovanou při vytváření závodu.

2.3.4 Shrnutí výhod a nevýhod programu

Tato sekce shrnuje poznatky z analýzy fsTimeru do přehledného seznamu dobrých nápadů, které stojí za to napodobit, a naopak věcí, které by bylo vhodné doladit nebo se jim vyhnout v jakémkoli novém systému. Mezi hlavní výhody fsTimeru patří:

- Nevyžaduje příliš mnoho ze strany organizátora – ke všemu stačí jeden počítač. Na druhou stranu lze přidáním více počítačů například zrychlit registraci nebo zpřesnit záznam výsledků.
- Umí zaznamenávat čas pouze pomocí startovního čísla a výkonu, pak ho automaticky přiřadit k závodníkovi a vygenerovat výsledkové listiny. To je obrovské vylepšení vůči řešení tabulkovým editorem.
- Pořadatel si může sám nadefinovat, které údaje chce o účastnících sbírat.
- Na webových stránkách aplikace je k dispozici rozsáhlá a velmi nápomocná dokumentace, která dokonce obsahuje užitečné rady k organizaci závodu tak, aby bylo použití fsTimeru co nejefektivnější a nejpřesnější.

Naopak nevýhody systému jsou například:

- Uživatelské rozhraní je poměrně nepřehledné. Je sice na první pohled jasné, co která část aplikace dělá, problém je v tom, že pořadatel nemá vždy přístup ke všem informacím. Například během záznamu výsledků se nemůže podívat na startovní listiny. Navíc je aplikace strukturována velmi roztroušeně, všude se otevírá příliš mnoho oken.
- Program nemá jednoduše použitelný instalační balíček ani pro OS Windows. Uživatel musí nejprve instalovat Python, až pak může začít používat fsTimer.
- Program vytváří ke každému závodu obrovské množství pomocných souborů. Na testovacím závodě jsem jich napočítal 11. Není v silách programu ani uživatele zaručit, že data budou konzistentní napříč všemi těmito soubory.
- Ukládání dat musí proběhnout ručně, což ne každý uživatel dělá automaticky při každé změně. Při nuceném restartu aplikace může dojít ke ztrátě dat.

- Systém rozdělování účastníků do kategorií je implementován pouze částečně. Každý účastník může patřit do více kategorií.
- Na cílové čáře potřebujeme počítač s desktopovým operačním systémem a klávesnicí, musí tam tedy na něj být dost místa a případně i přívod elektřiny.

Přestože je fsTimer užitečným řešením, tento poslední nedostatek by na mnoha závodech znamenal, že bychom jím vůbec čas měřit nemohli. Jelikož nijak nepodporuje manuální vkládání časů, stal by se zbytečným. V následující sekci analyzuji pohodlnější možnost záznamu časů, a sice postřednictvím chytrého telefonu.

2.4 Android aplikace Race Timer

Pro zjištění, zda existuje mobilní aplikace umožňující záznam časů a startovních čísel, jsem se obrátil na obchod Google Play. Na vyhledávací výraz „race timer“ nalezne sice spoustu aplikací s funkcí stopek, zadávání doběhnutých čísel však podporuje jen jedna. Tou je Race Timer [9].

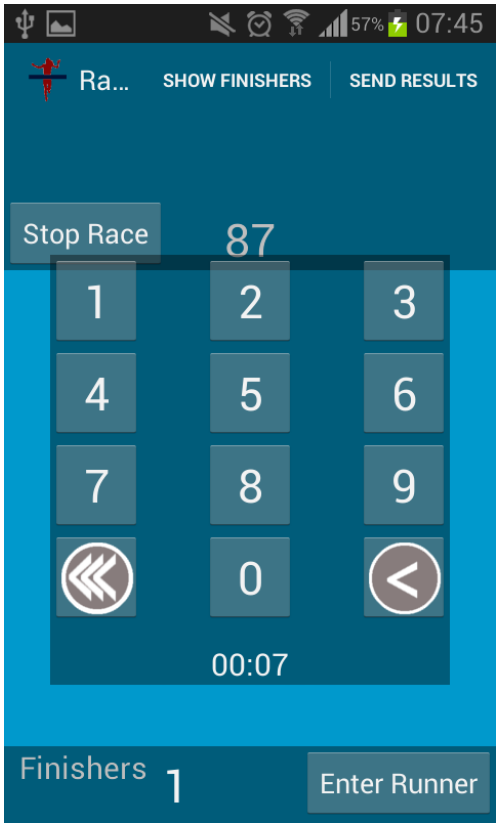
Situace by možná vypadala jinak na obchodě s aplikacemi pro operační systém iOS, nicméně vzhledem k vyšší relativní ceně zařízení s iOS a menšímu rozšíření platformy u nás se tímto ekosystémem nebudu zabývat.

Race Timer je tedy mobilní aplikace pro operační systém Android, dostupná zdarma. Seznam jejích funkcí se omezuje na spuštění stopek, napsání čísla na numerické klávesnici a odkliknutí proběhnutí závodníka, což k zadanému číslu přiřadí aktuální čas od spuštění stopek. Dále si lze prohlédnout zachycené záznamy, zastavit běžící časomíru a odeslat výsledky ve formátu CSV přes standardní kontextové sdílecí menu platformy Android.

Snímek obrazovky uživatelského rozhraní Race Timeru je na obrázku 2.11. Ač neoplývá krásou a místy se prvky i podivně překrývají, je zde vše důležité. Hlavním prvkem je velká numerická klávesnice pro vkládání startovních čísel. Pořadatel tedy nemusí čekat na to, než mu vyjede klávesnice systémová, pokaždé, když chce zadat číslo. Užitečná jsou také tlačítka v rozích klávesnice pro smazání jedné číslice nebo celého zadaného čísla.

Nachází se zde tlačítko pro rozběhnutí a následné zastavení časomíry. Ukončení časomíry musí uživatel ještě potvrdit. Pod tlačítkem „SHOW FINISHERS“ se skrývá jednoduchá tabulka naměřených časů a k nim přiřazených čísel.

Tímto výčet funkcí aplikace končí. Z chybějící funkcí je nejvýznamnější absence jakékoli perzistence dat. Je-li aplikace ukončena, vše je ztraceno. Tímto je také řešeno vrácení aplikace do původního stavu. Další nevýhodou aplikace je fakt, že záznam čísla a záznam času je svázán dohromady. Zaznamená-li uživatel čas bez čísla, další vložené číslo se přiřadí až k dalšímu naměřenému času. To představuje velký problém, protože není vždy možné si prohlédnout



Obrázek 2.11: Uživatelské rozhraní aplikace Race Timer

přibíhající čísla už před cílovou čarou. Ideálně bychom chtěli mít možnost přiřazovat čísla i poté, co závodníci proběhnou cílem.

2.5 Shrnutí

Tato kapitola analyzovala několik softwarových řešení, kterých můžou pořadatelé běžeckých závodů využít pro usnadnění organizace. Zaměřovala se především na online propagaci a předběžnou registraci, fyzický zápis běžců do závodu a následný záznam výsledků na cílové čáře.

Rozsáhlé možnosti propagace a zápisu závodníků v placených systémech jako ActiveEndurance tato práce nemůže napodobit v celé míře, je však možné si od nich vypůjčit základní myšlenky, neboť ty stojí na velkém množství času stráveného analyzováním potřeb organizátorů závodů. Konkrétně bude v novém systému implementováno online vypisování propozic závodu, možnost organizovat více závodů v rámci jedné akce, předběžná registrace na závod bez platby startovného a zveřejňování výsledků jednotlivých závodů souhrnně i dle věkových kategorií.

Co se týče bezčipového záznamu výsledků, systém použitý aplikací fsTimer je v rámci možností velmi robustní, aplikace Race Timer zase nevyžaduje přítomnost celého počítače. V systému vyvíjeném v této práci se pokusím to nejlepší z obou řešení sloučit dohromady. Hlavní převzaté myšlenky pro mou implementaci z těchto dvou aplikací jsou:

- Dynamická definice věkových kategorií
- Registrace účastníků do závodu pomocí formuláře
- Záznam výsledků pomocí startovního čísla a cílového času
- Automatické zpracovávání výsledků do výsledkových listin
- Běh časomíry na chytrém telefonu
- Záznam doběhnutých závodníků na chytrém telefonu

Kapitola 3

Návrh řešení

V této kapitole navrhnu vlastní řešení problematiky. Nejprve je potřeba jasně definovat požadavky na nový systém, podle nich pak bude možné vytvořit jeho architekturu. Nakonec převedu požadavky na konkrétní případy užití jednotlivých částí systému.

3.1 Požadavky na nový systém

Požadavky na vyvíjený systém vycházejí především z technologických omezení, kterým pořadatelé díky nedostatku peněz čelí. Jejich formulace vyplývá z analýzy existujících řešení a zkušeností autora.

Mým největším cílem je usnadnit pořadatelům práci se startovními a výsledkovými listinami. Chci zajistit, aby nebylo potřeba ručně data třídit do kategorií a řadit podle časů, aby se časy nikam nemusely ručně psát a aby nebylo zapotřebí použití papíru při zaznamenávání výsledků na cílové čáře. Mým dalším důležitým cílem je zkrátit čas, který musí účastníci závodu po doběhnutí čekat, než jsou ohlašovány výsledky, a také usnadnit registraci účastníků. Nakonec se chci zabývat možností přenosu mezičasů z tratě k cílové čáře, kde diváci netrpělivě čekají na zprávy z průběhu závodu.

3.1.1 Funkční požadavky

Tyto základní cíle jsou rozvedeny do konkrétních funkčních požadavků v tabulce 3.1 a popsány níže. Jak je z tabulky patrné, funkčních požadavků je poměrně mnoho. Každé skupině je tedy přiřazena priorita od 1 do 5, přičemž nejnižší číslo znamená nejvyšší prioritu. Skupina s nižší prioritou vždy závisí na implementaci všech skupin s prioritou vyšší a požadavky v jedné konkrétní skupině musí být implementovány najednou, neboť jejich rozdělení by znamenalo ztrátu logické návaznosti.

Systém budu vyvíjet v iteracích podle této tabulky, jednu skupinu funkčních požadavků po druhé. Tímto zaručím, že v případě nedostatečného množ-

ství času pro splnění všech požadavků budou splněny ty, které mají pro organizátory závodů největší přínos.

Číslo	Popis požadavku	Priorita
Základní nástroje pro správu účastníků		
FRQ.1.1	Založení nového závodu v systému	1
FRQ.1.2	Registrace účastníků do závodu	1
FRQ.1.3	Tisk startovních listin	1
FRQ.1.4	Manuální vkládání a úprava výsledků	1
FRQ.1.5	Tisk výsledkových listin	1
Časomíra a digitální záznam výsledků na cílové čáře		
FRQ.2.1	Běh časomíry na Android aplikaci	2
FRQ.2.2	Záznam doběhů na Android aplikaci	2
FRQ.2.3	Přenos výsledků do PC po skončení závodu	2
Vypisování, registrace a výsledky přes webové rozhraní		
FRQ.3.1	Vypsání závodu a zveřejnění na webu	3
FRQ.3.2	Možnost předběžné registrace na závod	3
FRQ.3.3	Automatické zveřejnění výsledků závodu na webu	3
FRQ.3.4	Upozornění na změny v závodě	3
Přenos zaznamenaných výsledků v reálném čase		
FRQ.4.1	Záznam mezičasů na trati	4
FRQ.4.2	Okamžitý přenos výsledků do PC a jejich zobrazení	4
Správa seriálu závodů		
FRQ.5.1	Založení seriálu závodů a sdružení závodů do seriálu	5
FRQ.5.2	Nastavení pravidel souhrnného hodnocení v seriálu	5
FRQ.5.3	Zveřejňování souhrnných výsledků seriálu	5

Tabulka 3.1: Souhrn funkčních požadavků na nový systém

3.1.2 Nefunkční požadavky

Nefunkční požadavky na systém nemají přiřazenou úroveň priority, jelikož musí být splněny úplně všechny, aby byl systém vůbec schopný plnit svoji funkci. Přehled nefunkčních požadavků je uveden v tabulce 3.2.

Číslo	Popis požadavku
Použitelnost a spolehlivost	
NFR.1.1	Dostupnost vypsaných závodů a výsledků z internetu
NFR.1.2	Dostupnost nástrojů pro správu závodu mimo internet
NFR.1.3	Měření času a záznam výsledků v podobě mobilní aplikace
NFR.1.4	Přenos výsledků do PC mimo dostupnost inernetu
NFR.1.5	Obsluhovatelnost časomíry a záznamu výsledků jednou osobou
NFR.1.6	Odolnost časomíry vůči násilnému ukončení běhu aplikace
Finanční omezení	
NFR.2.1	Provozovatelnost systému na běžném výpočetním vybavení
NFR.2.2	Efektivní využití SMS zpráv pro přenos mezičasů
Požadavky na testování	
NFR.3.1	Pokrytí veškerého netriviálního kódu unit testy
NFR.3.2	Integrační testy pro všechna interní rozhraní
NFR.3.3	Otestování systému na simulovaném závodě

Tabulka 3.2: Souhrn nefunkčních požadavků na nový systém

3.1.3 Popisy jednotlivých funkčních požadavků

FRQ.1.1: Uživatel bude moci v systému vytvořit nový závod a přidat k němu údaje jako název závodu, datum, čas a místo konání, délku trasy, startovné, ceny pro vítěze a další popis. V rámci vytváření závodu si pořadatel zvolí, jak budou účastníci rozřazováni do kategorií podle věku a pohlaví. Tato pravidla po vytvoření závodu již nebude možné měnit. Tento požadavek neřeší zveřejnění informací, pouze jejich evidenci v systému.

FRQ.1.2: Před konáním závodu bude moci pořadatel pomocí systému registrovat účastníky do závodu. Systém bude o účastníkovi povinně evidovat celé jméno, datum narození a pohlaví, nepovinně pak sportovní oddíl, jehož je součástí. Během registrace je každému účastníkovi přiděleno unikátní startovní číslo, pomocí něhož lze zaznamenávat výsledky. Startovní číslo lze vybrat buď ručně, nebo ho systém přidělí automaticky. Jako součást registrace také systém automaticky provede zařazení účastníka do jedné z kategorií závodu, které pořadatel specifikoval.

FRQ.1.3: Před začátkem závodu bude mít pořadatel možnost vytisknout startovní listiny, což jsou seznamy registrovaných závodníků seřazené dle

startovního čísla. Může si zvolit souhrnnou listinu (všechny kategorie), nebo listiny jednotlivých kategorií.

FRQ.1.4: Systém bude umožňovat manuální vkládání, úpravu a mazání výsledků závodu. Výsledkem závodu se rozumí dvojice startovního čísla a cílového času. Systém podle startovního čísla automaticky přiřadí čas k účastníkovi a podle času sestaví konečné pořadí běžců.

FRQ.1.5: Na konci závodu bude mít pořadatel možnost vytisknout výsledkové listiny, což jsou seznamy registrovaných závodníků seřazené vzestupně dle cílového času. Pořadatel si může zvolit souhrnnou listinu (všechny kategorie), nebo listiny jednotlivých kategorií.

FRQ.2.1: Systém bude umožňovat provozovat hlavní časomíru závodu na chytrém telefonu s operačním systémem Android pomocí specializované aplikace. Časomíra se manuálně spustí při startu závodu a zastaví se opět ručně až po doběhnutí posledního účastníka.

FRQ.2.2: Časoměřič bude moci pomocí Android aplikace zaznamenávat výsledky závodu. Stisknutím tlačítka na obrazovce bude moci zaznamenat čas doběhnutí účastníka, posléze bude moci k němu přiřadit startovní číslo. To zadá na numerické klávesnici. V případě, že se splete, bude moci výsledky libovolně přesouvat, mazat i měnit. Rozhraní musí být velmi rychle ovladatelné, neboť při zaznamenávání výsledků nemusí být mezi běžci velké rozestupy. Hlavním kritériem hodnocení splnění tohoto požadavku je zvládnutelnost celého procesu na cílové čáře jedním člověkem.

FRQ.2.3: Po ukončení časomíry bude možné spustit přenos zaznamenaných výsledků z telefonu do počítače. Tam se budou výsledky chovat stejně jako ty, které byly zadány manuálně.

FRQ.3.1: Pořadatel závodu bude mít možnost skrz systém informovat potenciální účastníky závodu o jeho konání. To bude realizováno prostřednictvím webového rozhraní, které bude zobrazovat informace o všech vypsaných závodech v systému. Tyto informace budou shodné s těmi, které jsou popsány v požadavku FRQ.1.1. Ve webovém rozhraní se budou zakládat „akce“. Akce se koná vždy v nějaký daný den a sdružuje v sobě jeden či více závodů lišících se od sebe například délkou tratě, věkovou hranicí apod. Webové rozhraní bude umožňovat prohlédnout si propozice akce na samostatné stránce.

FRQ.3.2: Potenciální účastník závodu bude mít možnost se předběžně zaregistrovat na závod skrz webové rozhraní. Bude muset zadat svou e-mailovou adresu, celé jméno a popřípadě kterého závodu v rámci akce se

zúčastní. Pořadatel si bude moci seznam předregistrovaných účastníků prohlédnout, aby měl představu o tom, kolik lidí se chystá na závod přijet. Předregistrovaný účastník si bude moci předem vybrat preferované startovní číslo. Během samotné registrace jsou pak jeho údaje v systému vyhledány, a pokud je preferované startovní číslo v tu chvíli ještě volné, bude účastníkovi automaticky přiřazeno.

FRQ.3.3: U již proběhnutých závodů si budou moci návštěvníci webového rozhraní prohlédnout výsledkové listiny. Nahrávání výsledků na web bude iniciováno pořadatelem závodu po jeho skončení a proběhne zcela automaticky. Po nahrání výsledků závodu na web je již nebude možné nijakým způsobem měnit.

FRQ.3.4: Pokud pořadatel změní nějaký údaj u zveřejněného závodu, například čas zahájení, bude všem předregistrovaným účastníkům automaticky odeslán e-mail s informací o změně.

FRQ.4.1: Systém bude umožňovat využití více mobilních telefonů pro záznam mezičasů na trati. Pro vytvoření jednoho kontrolního bodu na trati bude zapotřebí pouze jednoho dalšího časoměřiče s telefonem. Záznam mezičasů bude probíhat stejně jako záznam cílových časů.

FRQ.4.2: Systém bude podporovat okamžitý přenos jakýchkoli naměřených výsledků do počítače, kde mohou být ihned zobrazovány divákům na velké obrazovce. Mechanismus manuálního vkládání výsledků, stejně jako možnost přenést všechny výsledky až po skončení závodu, je zachován, v případě využití okamžitého přenosu však již nebude zapotřebí žádná další práce po skončení závodu. Výsledky budou zpracovány automaticky.

FRQ.5.1: Pořadatel závodu bude moci sdružit jednotlivé závody do seriálů. U seriálu bude uveden název, krátký popis a seznam pořádaných závodů, které jsou jeho součástí. Při vytváření závodu bude pořadatel dotázán, zda si přeje závod zařadit do některého ze svých seriálů.

FRQ.5.2: Pořadatel seriálu závodu bude moci nastavit pravidla pro souhrnné hodnocení. Bude si muset v rámci seriálu určit pravidla pro rozřazování do věkových kategorií napříč všemi závody. U závodu, který je součástí seriálu, tato pravidla nebude možné specifikovat, jelikož je bude závod dědit od seriálu.

FRQ.5.3: Systém bude automaticky sdružovat výsledky totožných účastníků napříč závody podle jména a data narození. V případě chybného

spárování bude možné výsledky sdružit ručně. Souhrnné (průběžné i konečné) výsledky budou vypočítány a zveřejněny návštěvníkům skrz webové rozhraní.

3.1.4 Popisy jednotlivých nefunkčních požadavků

NFR.1.1: Část systému umožňující zobrazování informací o závodě, možnost předregistrace a prohlížení zveřejněných výsledků musí být přístupná z širokého internetu.

NFR.1.2: Část systému umožňující správu jednoho závodu v průběhu jeho konání musí být dostupná i v lokalitách, kde není k dispozici připojení k internetu.

NFR.1.3: Část systému umožňující zaznamenávání čísel závodníků a jejich časů na cílové čáře musí být Android aplikace schopná běžet na jakémkoli mobilním telefonu s OS Android 4.0 nebo vyšším.

NFR.1.4: Přenos výsledků a mezičasů z telefonu do počítače musí být schopen fungovat bez připojení k širšímu internetu.

NFR.1.5: Mimo záznamu mezičasů na trati musí celý systém být schopna obsluhovat jediná osoba, která se bude moci zároveň věnovat ostatním úkonům spojeným s pořádáním závodu.

NFR.1.6: Běžící časomíra spolu s již zaznamenanými výsledky musí být odolná proti pádu aplikace, násilnému ukončení jejího běhu nebo vypnutí telefonu. Jakákoli ztráta zaznamenaných dat způsobená nesprávným fungováním systému je nepřípustná. Dosud neukončenou časomíru bude možné po opětovném spuštění aplikace znovu rozběhnout z hodnoty, kterou by měla, kdyby nikdy nebyla zastavena.

NFR.2.1: Použití systému nesmí být podmíněno větší investicí než na provoz jednoho webhostingového serveru a pořízení nebo provoz jednoho přenosného počítače, jedné tiskárny a jednoho chytrého telefonu s OS Android 4.0 nebo vyšší. Využití více chytrých telefonů pro záznam mezičasů bude nepovinné. Použití jedné části systému nebude podmíněno použitím všech ostatních.

NFR.2.2: Při využití možnosti zaznamenávání mezičasů musí být využita co nejefektivněji kapacita jedné SMS zprávy, aby se minimalizovala útrata u operátora.

NFR.3.1: Veškerý netriviální kód musí být pokryt unit testy.

NFR.3.2: Systém musí obsahovat integrační testy pro všechna vnitřní rozhraní, například mezi mobilní aplikací a zbytkem systému. Žádná vnější rozhraní kromě tisku nejsou vyžadována.

NFR.3.3: Systém musí být otestován na simulovaném závodě a prohlášen za použitelný organizátorem běžeckých závodů.

3.2 Přehled architektury

Z uvedených požadavků vyplývá, že se systém musí skládat z více částí. Již na první pohled je jasné, že je potřeba rozlišit mobilní aplikaci od zbytku systému. Je však nutné také oddělit webové rozhraní od aplikace pro správu závodu na místě. Bohužel není možné, aby se pořadatel přihlašoval na web, zatímco registruje účastníky nebo zaznamenává výsledky, neboť místa, na kterých se závody pořádají, jsou často odlehlá a ani mobilní internet na nich nedosahuje dostatečné spolehlivosti.

Systém tedy bude mít tři části: webové rozhraní, desktopovou aplikaci pro správu závodu a Anroid aplikaci pro záznam cílových časů a přenos mezičasů. Pro zkrácení budu používat pro tyto části názvy webová aplikace, desktopová aplikace a mobilní aplikace. Výsledná architektura a propojení jednotlivých částí pak bude vypadat jako na diagramu 3.1.

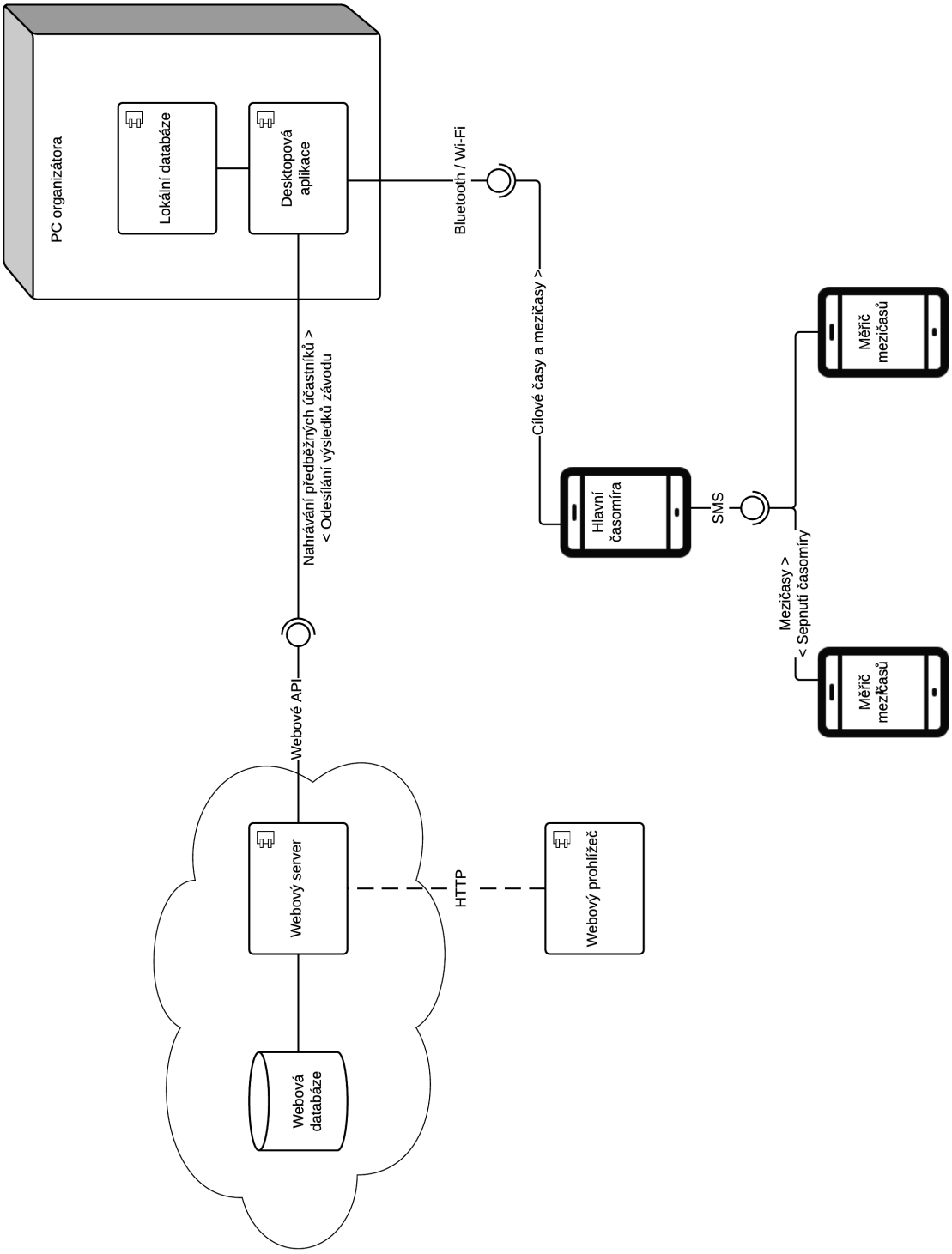
Architektura webové aplikace se nijak neliší od klasického pojetí databáze a webového serveru, k němuž se připojuje internetový prohlížeč, ať už organizátora či potenciálního účastníka. K webovému serveru se však také bude připojovat desktopová aplikace, která si vždy před závodem stáhne údaje o závodě jako věkové kategorie a seznam předběžně registrovaných účastníků, a která po skončení závodu nahraje na server konečné výsledky.

Propojení desktopové aplikace s webem nebude povinné. Desktopová aplikace bude umožňovat vytvořit závod, aniž by byl stažen z webu. Některé pokročilé funkce ale nemusí být pro takto vytvořené závody dostupné.

Desktopová aplikace si musí na lokálním souborovém systému počítače organizátora držet zadaná data, neboť ty tam musí vydržet do té doby, než bude organizátor znovu připojen k internetu a schopný je odeslat na web. Aplikace by si měla data ukládat automaticky při jakékoli změně, aby nemohlo dojít ke ztrátě.

Ve spodní části diagramu je zobrazen systém časoměrných telefonů, na kterých bude běžet naše mobilní aplikace. Uživatel jich může použít kolik chce, jejich počet není omezen ani shora, ani zdola.

Komunikace telefonů s desktopovou aplikací spočívá především v nahrávání zaznamenaných cílových časů a pořadí doběhnutých čísel do počítače. V souladu s požadavkem **FRQ.2.3** bude napřed implementována možnost přenosu výsledků přes Bluetooth po skončení závodu. V rámci **FRQ.4.2** pak vytvořím mechanismus okamžitého a automatického přenosu.



Obrázek 3.1: Celková architektura systému

Na tento přenos se hodí ad-hoc wi-fi síť, neboť se dá velice jednoduše nastavit, nevyžaduje připojení k širokému internetu a její krátký dosah nám nevadí, neboť předpokládám, že počítač a hlavní časoměrný telefon se nebudou v průběhu závodu nacházet moc daleko od sebe.

Komunikace hlavního telefonu a zařízení měřících mezičasy už nemůže probíhat prostřednictvím wi-fi, neboť vzdálenosti mezi telefony na trati budou příliš velké. Mobilní připojení k internetu také použít nelze. Spolehnu se však na to, že klasický GSM signál k dispozici bude a pro přenos dat bude využit systém krátkých zpráv SMS. To představuje několik implementačních oříšků, neboť tato služba je zpoplatněna mobilními operátory a její neefektivní využití by mohlo pořadatele závodu zruinovat. Bude také nutné vyřešit synchronizaci časomíry mezi jednotlivými telefony vzhledem k tomu, že doručování textových zpráv není příliš časově spolehlivé. Řešení těchto problémů jsou detailně popsána v kapitole o mobilní aplikaci.

3.3 Případy užití

Případy užití pokrývající funkční požadavky jsou rozděleny do tří částí podle architektury systému: mobilní, desktopové a webové. Podrobnější popis je uveden pouze u těch z nich, jejichž průběh není na první pohled patrný.

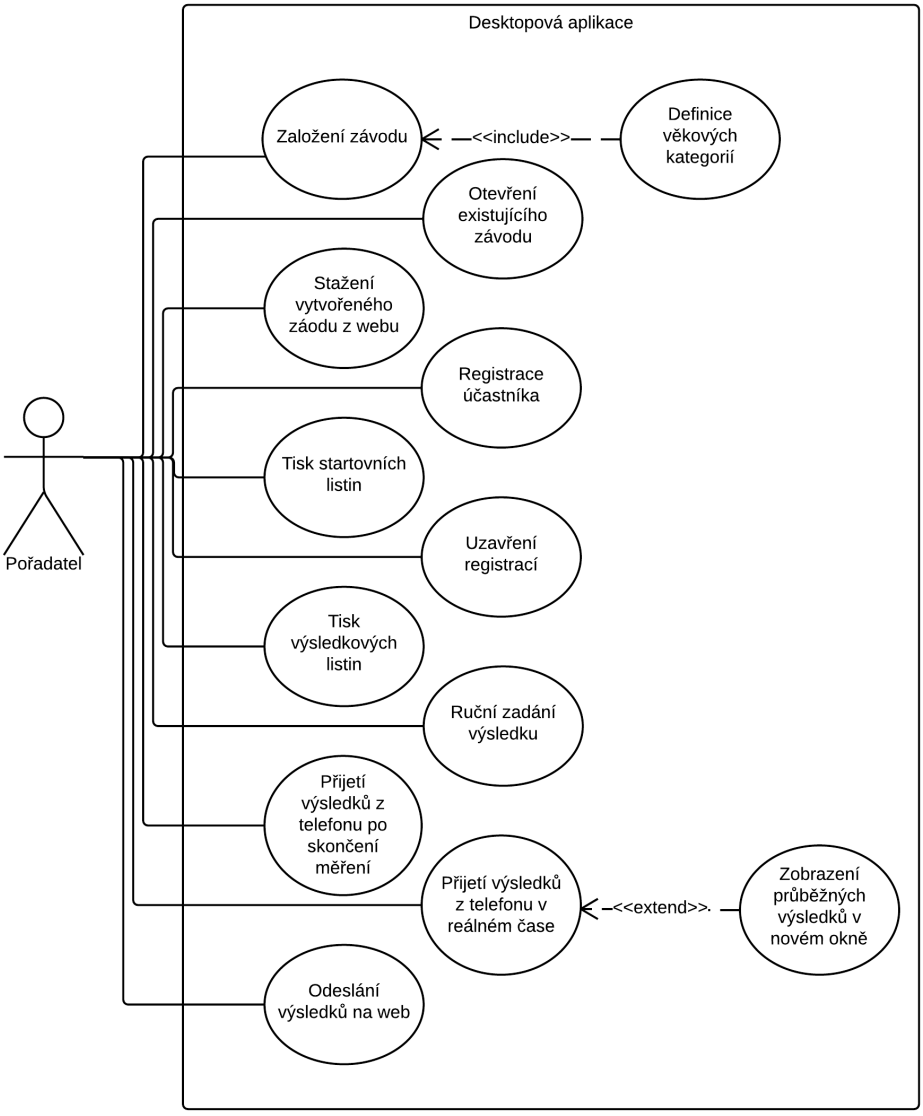
3.3.1 Případy užití desktopové aplikace

Přehled případů užití desktopové aplikace je uveden na obrázku 3.2

Založení závodu Ve většině případů by se závod v desktopové aplikaci zakládat vůbec neměl. Hlavní scénář založení závodu směřuje přes webovou aplikaci a následné stažení do desktopové. V rámci modularizace systému a v duchu možnosti používat pouze tu část systému, kterou organizátor potřebuje, je však možné založit závod i bez jeho uveřejnění na webu.

Definice věkových kategorií Tento úkon se provádí pouze u lokálně založených závodů. Závody vytvořené na webu již definici kategorií mají. Jedná se o rozdělení možného věkového rozhraní účastníků do jednotlivých intervalů. Jedno takové rozdělení definuje pořadatel u mužů, druhé u žen. Bude možné specifikovat, zda rozhodující je pro zařazení do kategorií fyzický věk v den registrace, či pouze rok narození.

Registrace účastníka Fyzická registrace probíhá vyplněním několika předem definovaných údajů. Pořadatel má na obrazovce neustále k dispozici seznam registrovaných běžců a může záznamy mazat. Pro předregistrované účastníky je celý proces zrychlen, stačí jen zadat jejich jméno a potvrdit registraci.



Obrázek 3.2: Případy užití desktopové aplikace

Ruční zadání výsledku Manuální zadání výsledku spočívá ve vepsání startovního čísla a konečného času do formuláře v aplikaci. Záznamy má pořadatel neustále k dispozici a může je mazat.

Přijetí výsledků z telefonu po skončení měření Jedná se o synchronizaci výsledků pomocí Bluetooth. Jakmile pořadatel ukončí časomíru na telefonu, může připojit telefon k počítači a přenést naměřené výsledky. Není tedy potřeba využít manuálního vepisování čísel a časů, v případě chyb v přenosu je však stále k dispozici.

Přijetí výsledků z telefonu v reálném čase Tímto je myšlen okamžitý přenos výsledků, ať už mezičasů nebo cílových časů, z telefonu do desktopové aplikace. Před použitím této funkcionality je potřeba ji přes rozhraní desktopové aplikace nastavit.

Zobrazení průběžných výsledků v novém okně Tímto je umožněno zobrazování mezičasů a cílových časů přijatých v reálném čase divákům například na externím monitoru.

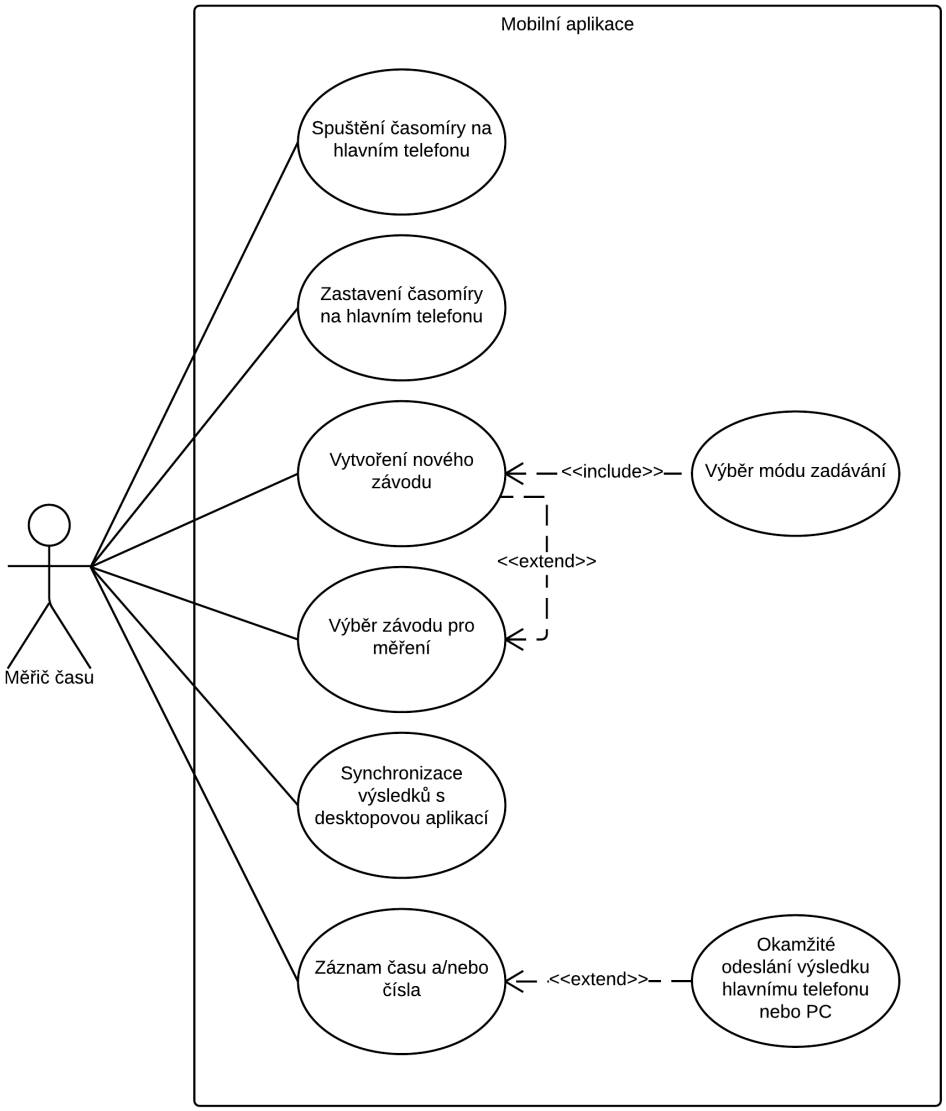
3.3.2 Případy užití mobilní aplikace

Přehled případů užití mobilní aplikace je uveden na obrázku 3.3.

Spuštění a zastavení časomíry na hlavním telefonu Spuštění časomíry zahrnuje odeslání informace o spuštění spolu s přesným časovým razítkem všem ostatním časoměrným telefonům na trati. Informaci o zastavení časomíry již není nutné rozesílat, aplikace se ale musí uživatele zeptat, zda opravdu časomíru zamýšlí zastavit.

Výběr módu zadávání Aplikace umožňuje čtyři různé způsoby zadávání výsledků:

- Zadávání časů a čísel zvlášť. Měřič času nejprve stiskne tlačítko pro záznam času, a až poté k času doplní číslo závodníka, který právě doběhl. Tímto způsobem lze lépe zvládat shluky závodníků.
- Zadávání časů a čísel dohromady. Měřič času zadá číslo dobíhajícího závodníka a stiskne tlačítko pro záznam času. Tím se zaznamená číslo spolu s časem podobně, jako to dělá analyzovaná aplikace Race Timer.
- Zadávání pouze čísel nebo pouze časů. Tento způsob se dá vuyžít na závodech, kde jsou na jeden bod dostupní dva měřiči. Jeden z nich zaznamenává časy, druhý čísla. Výsledky se zkombinují až po odeslání do desktopové aplikace.



Obrázek 3.3: Případy užití mobilní aplikace

Synchronizace výsledků s desktopovou aplikací Tento případ užití znamená synchronizaci všech výsledků najednou přes Bluetooth do počítače až po zastavení časomíry.

Okamžité odeslání výsledků hlavnímu telefonu nebo PC Vztahuje se pouze na závody, kde se organizátor rozhodl okamžité odesílání výsledků použít a kde na to existují technické možnosti (dosah komunikace mezi PC a telefonem je dostatečný, na místě je GSM signál pro přenos mezičasů apod.)

3.3.3 Případy užití webové aplikace

Přehled případů užití mobilní aplikace je uveden na obrázku 3.4. Dědičnost mezi pořadatelem a neregistrovaným uživatelem znamená, že přihlášenému pořadateli zůstávají k dispozici všechny funkce, ke kterým měl přístup před přihlášením.

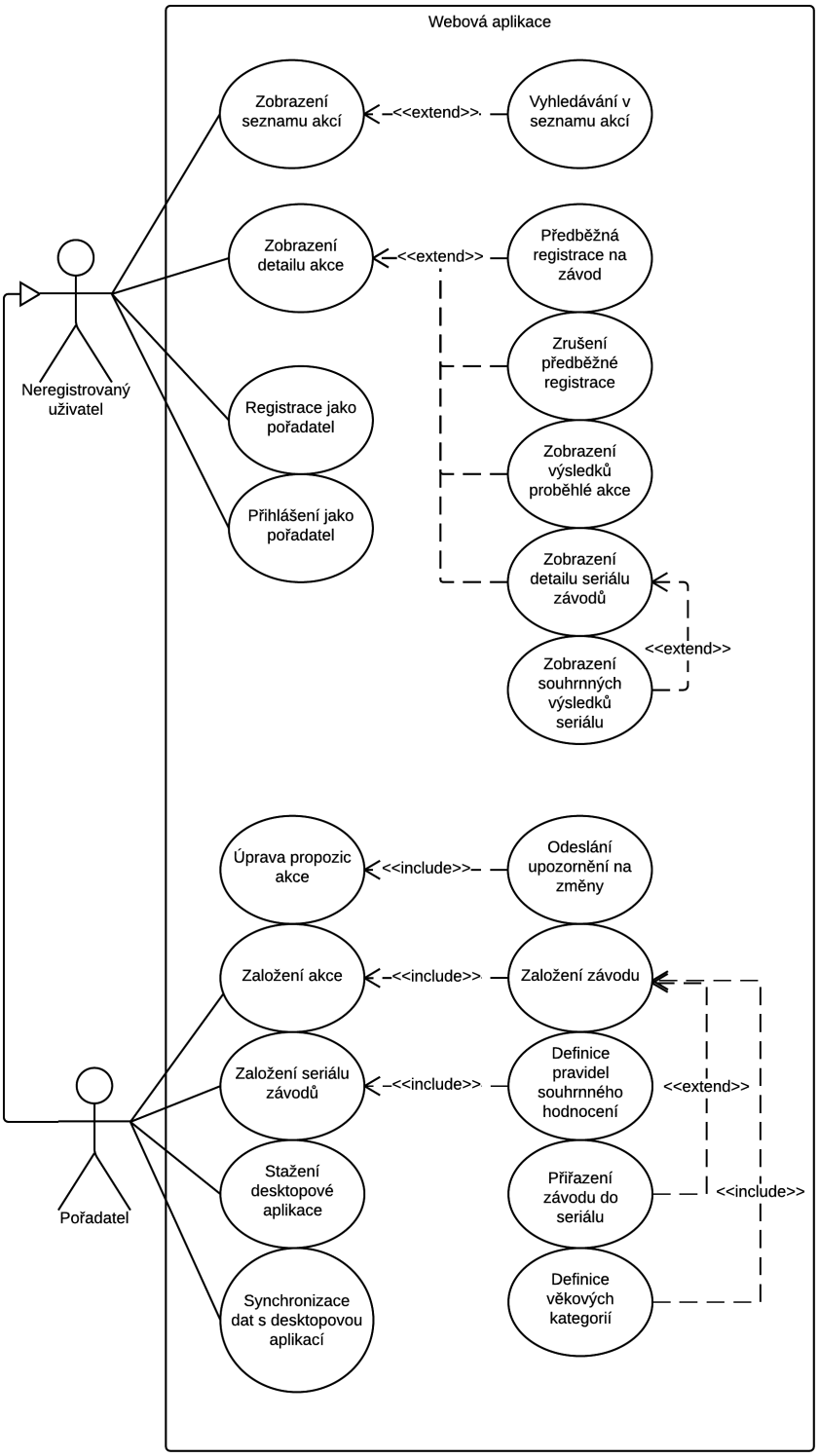
Předběžná registrace a její rušení Návštěvník se může předběžně registrovat na závod. To obnáší výběr závodu ze všech pořádaných na dané akci, vyplnění všech údajů, které pak budou potřeba k fyzické registraci, a zadání kontaktního e-mailu. Po dokončení předběžné registrace přijde účastníkovi e-mailem potvrzení s odkazem, který může využít pro zrušení registrace. Při jakýchkoli změnách v údajích o akci, o kterých bude chtít pořadatel informovat, přijde účastníkovi zpráva.

Registrace jako pořadatel Tato funkce v současné době nebude nijak omezena, závody bude moci zakládat kdokoliv.

Synchronizace dat s desktopovou aplikací Směr nahrávání dat si zvolí pořadatel v desktopové aplikaci při zahájení synchronizace. Může buď nahrát informace o závodě a předregistrovaných účastnících do desktopové aplikace, nebo nahrát výsledky z desktopové aplikace na web. Při nahrávání výsledků závodu, který je součástí seriálu, se musí automaticky přiřadit výsledky k existujícím účastníkům. V případě konfliktů či nejasností je může pořadatel přiřadit ručně.

3.4 Vztah mezi funkčními požadavky a případy užití

Tabulky 3.3, 3.4 a 3.5 ukazují vztah mezi případy užití a funkčními požadavky. Šedě vyplněné pole v tabulce znamená, že příslušný případ užití naplňuje požadavek s číslem v hlavičce daného sloupce. Všechny sloupce mají alespoň jedno vyplněné pole, každý funkční požadavek je tedy pokryt minimálně jedním případem užití.



Obrázek 3.4: Případy užití webové aplikace

V tabulkách jsou uvedeny jen ty případy užití, bez jejichž implementace by funkční požadavek nebyl naplněn. Ostatní podpůrné případy užití jsou z tabulky vypuštěny.

Případ užití	Číslo požadavku (FRQ)							
	1.1	1.2	1.3	1.4	1.5	2.3	3.3	4.2
Založení závodu	■							
Definice věkových kategorií	■							
Registrace účastníka		■						
Tisk startovních listin			■					
Tisk výsledkových listin					■			
Ruční záznam výsledků				■				
Přijetí výsledků z telefonu						■		
po skončení měření						■		
Přijetí výsledků v reálném čase								■
Odeslání výsledků na web							■	

Tabulka 3.3: Tabulka pokrytí funkčních požadavků případy užití desktopové aplikace

Případ užití	Číslo požadavku (FRQ)				
	2.1	2.2	2.3	4.1	4.2
Spuštění časomíry na telefonu	■				
Zastavení časomíry na telefonu	■				
Vytvoření nového závodu	■				
Výběr módu zadávání		■			
Synchronizace s desktop aplikací			■		
Záznam času / čísla		■		■	
Okamžité odeslání výsledku					■

Tabulka 3.4: Tabulka pokrytí funkčních požadavků případy užití mobilní aplikace

3.5 Shrnutí

Tato kapitola stanovila funkční i nefunkční požadavky na vyvíjený systém a představila architekturu, kterou lze naplnění požadavků dosáhnout. Nakonec

Případ užití	Číslo požadavku (FRQ)						
	3.1	3.2	3.3	3.4	5.1	5.2	5.3
Webová aplikace							
Založení akce / závodu	■						
Zobrazení seznamu a detailů akcí	■						
Synchronizace s desktop aplikací		■	■				
Předběžná registrace na závod		■		■			
Odeslání upozornění na změny				■			
Zobrazení výsledků proběhlé akce			■				
Založení seriálu závodů					■	■	
Přiřazení závodu do seriálu					■		
Zobrazení výsledků seriálu							■

Tabulka 3.5: Tabulka pokrytí funkčních požadavků případy užití webové aplikace

definovala případy užití jednotlivých částí systému. Navržená architektura o třech komponentách umožňuje realizovat všechny funkční i nefunkční požadavky. Vypuštění kterékoli části by znamenalo nesplnění nefunkčních požadavků. Ve zbytku práce se budu zabývat realizací zde popsaného návrhu.

Kapitola 4

Realizace

Tato kapitola popisuje konkrétní způsob, jakým byly všechny tři části systému implementovány. U každé části obsahuje zdůvodnění výběru technologií, detaily týkající se implementace a rozhraní vůči jiným částem systému. Dohromady pak utváří popis výsledného produktu implementace jak z uživatelského, tak z vývojářského pohledu.

Poznámka o názvu systému V této kapitole a ve zdrojovém kódu je možno se setkat s výrazem „Paseka“. Toto je označení systému používané ve zdrojovém kódu. Je to název místa konání závodu, který dal vzniknout nápadu tento systém implementovat [1].

Poznámka o stavu realizace Předchozí kapitola definuje pět implementačních iterací, z nichž každý má přiřazenou prioritu od jedné do pěti. Samotná implementace v době psaní této práce vzhledem k rozsahu navrženého řešení dospěla až k začátku čtvrté iterace, jsou tedy implementovány a otestovány všechny požadavky s prioritou 1, 2 a 3. Jejich realizace je zde podrobně popsána.

Vzhledem k rozpracovanosti zbývajících požadavků může zde popsaný návrh implementace být v některých místech obsáhlejší než implementace samotná. Taková místa jsou v textu jasně označena. Požadavky, u kterých ještě nebyl vytvořen technický návrh, jsou z této kapitoly vypuštěny úplně.

4.1 Desktopová aplikace

Desktopová aplikace byla vyvíjena jako první, neboť stojí uprostřed navržené architektury celého systému a má největší potenciál usnadnit pořadatelům práci.

4.1.1 Volba technologií

Desktopovou aplikaci jsem se rozhodl implementovat v programovacím jazyce Python. Python je interpretovaný, objektově orientovaný, dynamicky typovaný jazyk na vysoké úrovni abstrakce [10]. Jeho standardní distribuce obsahuje velkou knihovnu nástrojů pro širokou škálu využití.

Python jsem zvolil zejména díky tomu, že je to vysokoúrovňový a dynamicky typovaný jazyk. Důležitou výhodou jsou nižší nároky na čas strávený programováním a také expresivita jazyka. Nevýhoda vysokoúrovňových jazyků – nižší efektivita využití systémových zdrojů – zde nepředstavuje velký problém, jelikož desktopová aplikace neprovádí žádné složité počty. Dalším argumentem ve prospěch Pythonu je fakt, že je nezávislý na operačním systému.

Pro Python existuje několik knihoven umožňujících tvorbu grafického uživatelského rozhraní [11]. Přibalena přímo k distribuci Pythonu je knihovna TkInter, která je však podle téměř všech zdrojů zastaralá, pomalá a chybí v ní zásadní prvky [11] [12]. Proto jsem se rozhodl se jí vyhnout.

Grafické rozhraní aplikace je řešeno pomocí knihovny PyQt. Jedná se o Pythonovský „obal“ knihovny Qt od společnosti The Qt Company Ltd., která je určená pro aplikace napsané v C++ [13]. Hlavními výhodami použití PyQt jsou platformní nezávislost, přehledná a obsáhlá dokumentace, nativní vzhled na všech podporovaných platformách [12] a také promyšlená podpora MVC paradigmatu při práci se seznamy a tabulkami [14]. Desktopová aplikace s tabulkami pracuje na mnoha místech.

Na samotnou tvorbu grafických prvků jsem použil program Qt Designer a skript `pyuic5`, který je součástí distribuce PyQt a který transformuje výstup Qt Designeru přímo do Pythonu. Tam lze vygenerované třídy používat, rozšiřovat a měnit bez zásahu do vytvořeného kódu, čímž je zachována možnost ho vygenerovat znovu po nějaké změně.

Knihovna Qt je duálně licencované dílo. Tato práce využívá Qt pro nevýdělečné účely a v souladu s licencí GPL, díky čemuž ji dle licenčních podmínek Qt lze používat. Obdobně je licencováno i PyQt.

Jako systém pro ukládání dat jsem zvolil relační databázi SQLite [15]. Pro desktopové aplikace je nejpoužívanějším řešením jednoduchý zápis do souboru, ten má však několik nevýhod. Největší z nich je fakt, že pro uložení sebemenší změny v perzistentních datech je potřeba znovu ukládat celý soubor. Spousta aplikací rozhodnutí uložit data nechává na uživateli, což mívá nepříjemné následky v podobě ztráty dat, pokud je běh aplikace či systému násilně ukončen.

Relační databáze naopak pracují s předpokladem, že je vždy potřeba uchovat vše, což v této aplikaci je. Mají navíc tu výhodu, že používají standardizované rozhraní (SQL), a proto je pro ně jednoduché vyvíjet knihovny pro usnadnění práce. Jednu takovou knihovnu, Peewee [16], používá i desktopová aplikace. Peewee je knihovna pro objektově-relační mapování (ORM), díky níž lze obejít psaní zvláštního kódu pro interakci s databází. Peewee automaticky

vytváří a aktualizuje databázové tabulky, stará se o ukládání a načítání doménových objektů a realizuje vztahy mezi entitami, což významně urychluje vývoj aplikace.

Pro přístup k Bluetooth adaptéru na operačních systémech Windows a Linux je využita knihovna PyBluez. Jiné řešení, které by podporovalo Service Discovery Protocol, jež vyžaduje Android klient, v současnosti pro Python neexistuje. Tím je bohužel ztracena podpora Bluetooth synchronizace na systému Mac OS. Pro ten existují specializované knihovny, neměl jsem však přístup k testovacímu prostředí s tímto operačním systémem.

4.1.2 Implementační detaily

Architektura desktopové aplikace je vícevrstvá a rozdělená do Python modulů, které vymezují a oddělují kód reprezentující doménové entity, kód, který s těmito entitami manipuluje, a kód, který má na starosti uživatelské rozhraní. Implementace desktopové aplikace probíhala po vrstvách od reprezentace domény až po grafické uživatelské rozhraní.

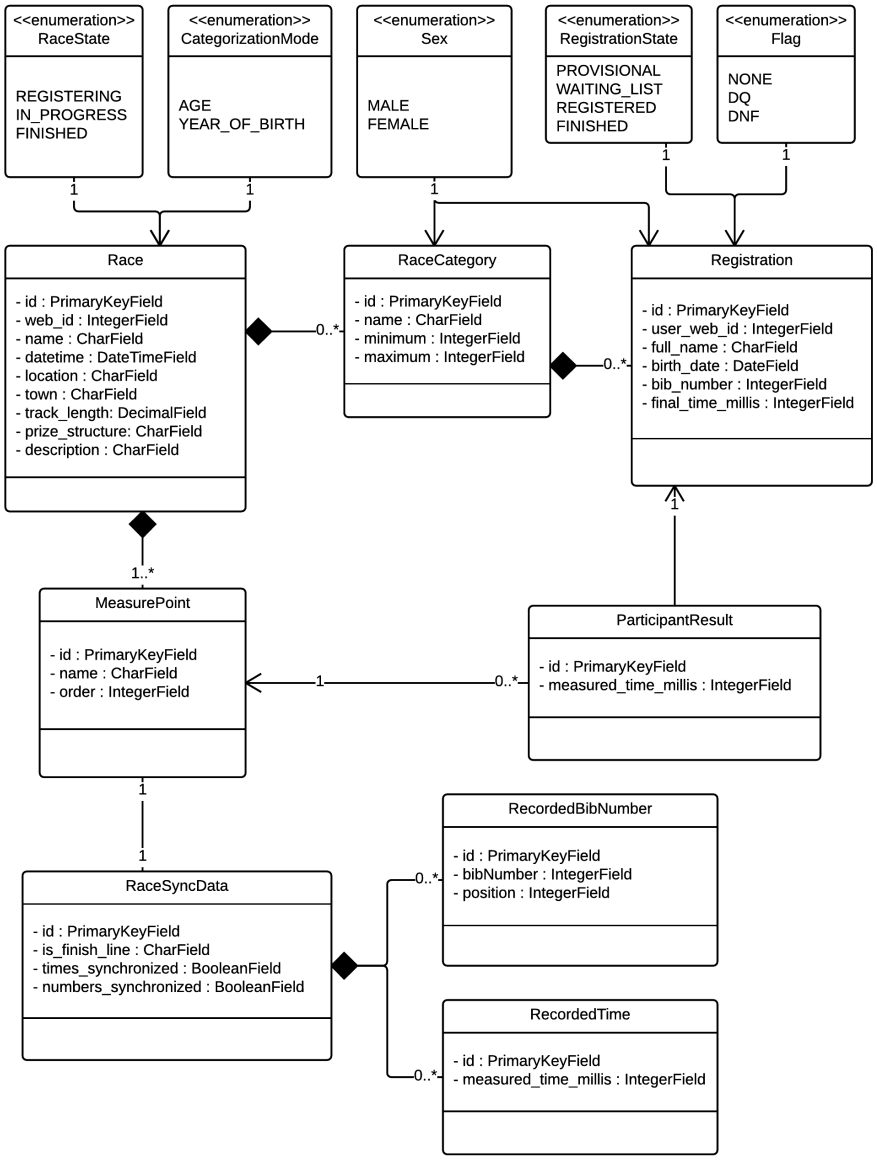
Třídy doménového modelu pro desktopovou aplikaci jsou znázorněny na UML diagramu tříd v obrázku 4.1. Diagram z důvodu úspory místa nezobrazuje jména instančních proměnných na vztazích mezi třídami, ty však povětšinou kopírují názvy koncových tříd a pro pochopení vztahu nejsou potřeba.

Ani jedna znázorněná třída nemá žádné metody. To je návrhové rozhodnutí, neboť jakékoli změny stavu domény musí jít vždy přes vrstvu business logiky. Všechny doménové entity však dědí od třídy `Peewee.Model`, podporují tedy základní operace pro ukládání a čtení z databáze.

Hlavní entitou celé domény je třída `Race`, která představuje pořádaný závod. Pole **name** a **datetime** slouží k identifikaci závodu v uživatelském rozhraní, pole **web_id** pak umožňuje zpětné přiřazení výsledků při synchronizaci s webovou aplikací. Ostatní pole využívá pouze webová aplikace pro zobrazení informací účastníkům.

Účastník závodu je reprezentován třídou `Registration` a spojen s objektem `Race` pomocí třídy `RaceCategory`, jejíž instance představují věkové kategorie, které pořadatel pro závod definoval a do jedné z nichž je každý účastník jednoznačně přiřazen.

Systém podporuje dva druhy kategorizace, a sice řazení na základě fyzického věku účastníka, nebo jeho roku narození. Enumerační typ `CategorizationMode` představuje tyto dva způsoby, ze kterých si může uživatel při tvorbě závodu vybrat. Na první pohled by se mohlo zdát, že není důvod řadit jinak než podle věku, na většině závodů je tomu však naopak. Jedním důvodem je, že je takovéto rozřazování jednodušší, zvláště, když se musí provádět manuálně. Zadruhé, je-li závod zařazen do nějakého seriálu, může se stát, že se účastník během konání seriálu přehoupne z jedné kategorie do druhé, což znamená velké problémy pro počítání souhrnného hodnocení.



Obrázek 4.1: Diagram tříd domény desktopové aplikace

Z těchto důvodů jsou instanční proměnné třídy `RaceCategory` zobecněné na minimum a maximum. Obě hodnoty můžou obsahovat buď letopočet, nebo věk. Vždy však platí, že `RaceCategory.minimum` je menší než nebo rovno `RaceCategory.maximum`, při případné změně z jednoho módu do jiného se tak musí tyto dvě přepočítané hodnoty prohodit.

Zbývající třídy ve spodní části diagramu se zabývají měřením časů. Třída `MeasurePoint` představuje jeden bod na trati, kde je měřen čas průběhu závodníků. Každý závod má minimálně jeden takový bod – cílovou čáru. Může jich však být víc, pokud se organizátor rozhodl pro měření mezičasů. Jednotlivé body jsou seřazeny polem **order** od bodu nejblíž k místu startu až po cílovou čáru.

Třída `ParticipantResult` je realizací vztahu n ku m mezi `MeasurePoint` a `Registration`. Obsahuje informaci o tom, za jaký čas od spuštění časomíry uběhl daný závodník úsek od startovní čáry do měřeného bodu. Seřazením účastníků podle těchto časů získáme výsledkovou listinu.

V současném stavu implementace neexistují třídy `MeasurePoint` a `ParticipantResult`, neboť měření mezičasů nebylo ještě implementováno. Každý závod má tím pádem jen jeden bod měření času a každý závodník jen jeden naměřený čas. Ten je uložen v poli **finish_time_millis** třídy `Registration`. V současnosti také existuje přímá vazba jedna ku jedné mezi `Race` a `RaceSyncData`.

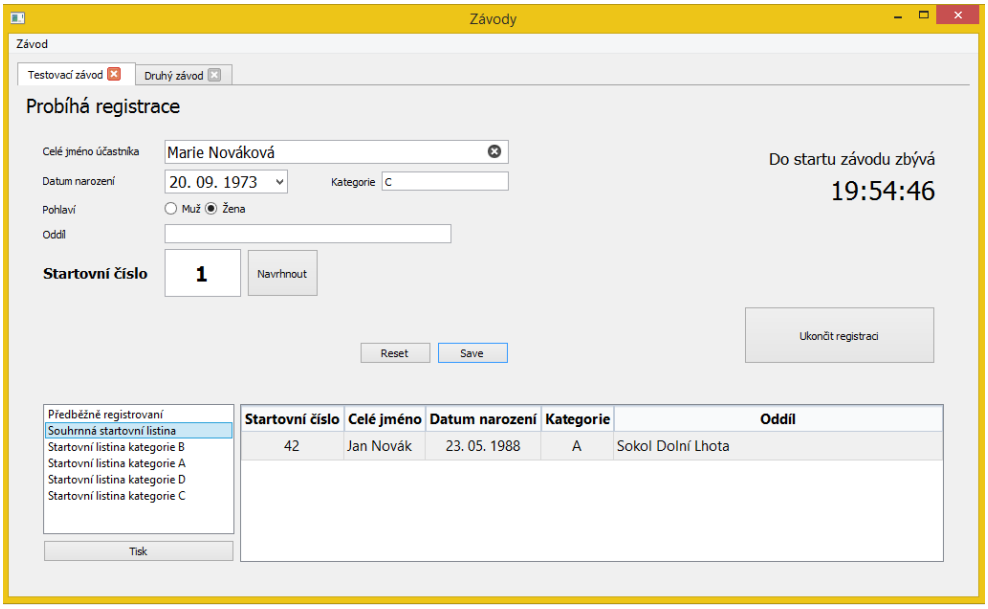
Trojice entit v nejspodnější části diagramu představuje data, která byla přijata od mobilní aplikace. Struktura těchto dat je podrobněji popsána v sekci 4.2.4.

4.1.2.1 Business logika

Metody, které manipulují s doménovými entitami, jsou seskupeny do jediného Python modulu, `services.py`. Ten realizuje návrhový vzor „fasáda“ [17] a funguje jako jediný přístupový bod k doménovým objektům.

Při implementaci této fasády jsem upustil od objektového přístupu, neboť mým cílem bylo vytvořit sadu funkcí, jež samy o sobě nedrží žádný vnitřní stav. Tím jsem dosáhl lepší testovatelnosti a přehlednosti kódu. Tyto a další výhody funkcionálnějšího stylu programování uvádí [18]. Jakmile byla business vrstva napsána a otestována, stačilo zajistit, že kód uživatelského rozhraní nebude přistupovat k modelům přímo, a vyhnul jsem se tak hledání problémů a nekonzistencí v modelové reprezentaci při psaní už tak složité GUI vrstvy.

Modul obsahuje metody pro vytváření a upravování závodů, registrací, práci se starovními čísly, věkovými kategoriemi, generování dat pro startovní a výsledkové listiny, záznam, úpravu a mazání výsledků a zpracování vstupů i výstupů pro rozhraní vůči mobilní i webové aplikaci. Samotné odesílání a přijímání dat v rámci těchto rozhraní je však implementováno zvlášť v modulu `integration`.



Obrázek 4.2: Registrace běžců v desktopové aplikaci

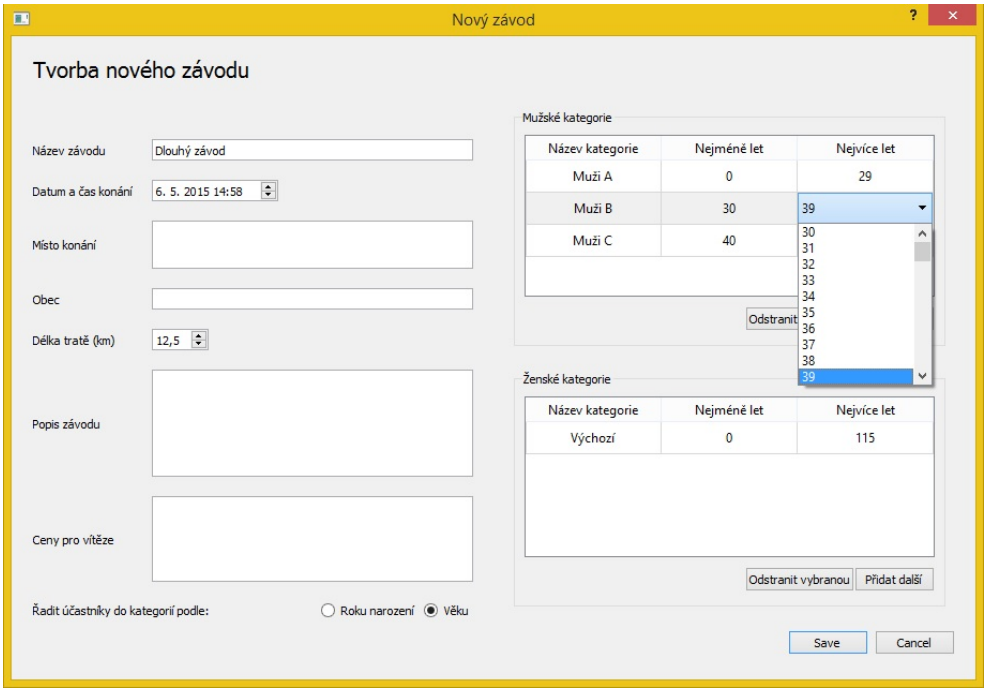
Kromě `services` obsahuje business vrstva aplikace ještě pomocné moduly `errors`, který definuje aplikační výjimky, a `categorization`, který pro změnu implementuje návrhový vzor „template method“ a abstrahuje logiku přiřazování účastníků do věkových kategorií. Obsahuje celkem tři třídy, jednu pro každý způsob kategorizace, a společnou nadtřídu. Každá z podtříd obsahuje vlastní implementaci určení nejdřívějšího a nejpozdějšího možného data narození účastníka, aby mohl být přiřazen do dané kategorie. Nadtřída pak tyto informace použije k určení, do které ze všech definovaných kategorií pro závod má být běžec zapsán.

4.1.3 Uživatelské rozhraní

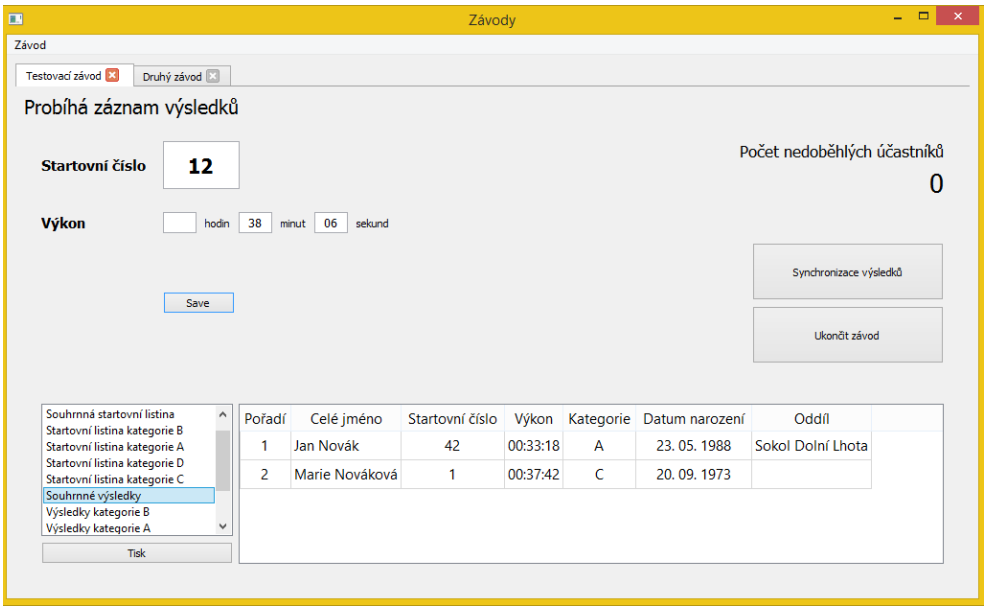
Pro tvorbu uživatelského rozhraní jsem využil služeb programu Qt Designer, který je součástí distribuce knihovny Qt. Jedná se o grafický nástroj, kde je možné navrhnout hrubou strukturu jednotlivých oken a komponent a jejich prostorové rozložení. Výstup Qt Designeru je XML soubor, jenž je skriptem `pyuic5` transformován do Python kódu.

Přeložené výstupy Qt Designeru se nachází v balíčku `ui.generated`. Třídy definované v těchto souborech se pouze importují do samotných UI prvků, které s nimi mohou libovolně manipulovat, není tedy třeba vygenerované soubory jakkoli měnit a zůstává možnost je z Designeru exportovat znovu.

Architektura uživatelského rozhraní se skládá z dialogů, které se starají o správu uložených závodů a hlavního okna obsahujícího jeden nebo více otevřených *kontextů*. Kontext je sada prvků grafického rozhraní umožňující správu



Obrázek 4.3: Tvorba závodů v desktopové aplikaci



Obrázek 4.4: Záznam výsledků v desktopové aplikaci

jediného závodu. Aplikace také obsahuje vedlejší okna, která se zobrazují při přijímaní či odesílání dat.

Při startu aplikace se uživateli zobrazí nabídka, zda chce vytvořit nový závod, otevřít uložený, nebo stáhnout data z webu. Při výběru volby „založit nový závod“ se ukáže dialog znázorněný na obrázku 4.3, který umožňuje vytvořit závod v desktopové aplikaci a definovat k němu rozdělení věkových kategorií. Je realizován třídou `RaceDefinitionDialog`. Výběr uloženého závodu implementován ve třídě `SelectRaceDialog`. Tento dialog obsahuje jednoduchou tabulku všech uložených závodů s možností jejich otevření nebo smazání.

Na obrázku 4.2 je zobrazen stav aplikace po otevření dvou závodů. Kořenem hierarchie grafických elementů je hlavní okno, `PMainWindow`, obsahující záložkovou lištu držící otevřené kontexty. Při zavření všech kontextů se aplikace vrací do stejného stavu, v jakém byla ihned po spuštění.

Kontext závodu Uživatelské rozhraní pro správu jednoho závodu je implementováno třídou `PContext`. Na obrázku 4.4 je to vše, co se nachází pod lištou záložek. Kontext je dále rozdělen na dvě oblasti – pracovní plochu a tabulku účastníků.

V horní polovině kontextu se nachází plocha, která se mění dle toho, v jaké fázi se zrovna daný závod nachází. Zatímco probíhá registrace, jak je vidět na obrázku 4.2, je zde zobrazen registrační formulář (třída `PRegistrationWorkspace`). Po ukončení registrace se zde zobrazuje formulář pro ruční vkládání výsledků a tlačítka vedoucí k synchronizaci s mobilní časomírou či webovou aplikací. Výsledkovou pracovní plochu implementuje třída `PResultWorkspace`.

Spodní polovinu kontextu zabírá tabulka účastníků. Na její levé straně je ovládací prvek, který pořadateli umožňuje vybrat data, která zrovna chce vidět. Při probíhající registraci je zde seznam předregistrovaných účastníků, souhrnná startovní listina a startovní listiny každé věkové kategorie závodu. Jakmile je registrace ukončena, přibudou zde výsledkové listiny, opět jedna souhrnná a jedna od každé kategorie. Vždy je možné aktuálně vybranou tabulku vytisknout.

Registrační a výsledkové tabulky zobrazují mírně odlišná data, naprogramovány jsou však všechny stejně. O vše se stará třída `PRaceTable` a další pomocné třídy definované ve stejném modulu. Ta obsahuje mechanismus dynamického přidávání sloupců, z nichž každý je realizován vlastní třídou. Pokud bude potřeba do budoucna zobrazovat ještě jiné pohledy na registrované účastníky, tento mechanismus jejich implementaci výrazně usnadní, neboť bude stačit dopsat nové sloupce.

Tabulka uvnitř kontextu závodu je jedna z největších předností desktopové aplikace. Seznamy běžců a výsledky má pořadatel neustále k dispozici, bez ohledu na to, jaký úkon v aplikaci zrovna provádí.

4.2 Mobilní aplikace

Mobilní aplikace je podstatně menší projekt z hlediska implementace než obě zbývající části systému. Je to dáno především menším počtem případů užití a zredukováním problémové domény na pouhé tři koncepty: závod, startovní číslo a změřený čas. Tímto zjednodušením zaniká nutnost synchronizovat data do mobilního telefonu před závodem a díky němu zůstává uživatelské rozhraní mobilní aplikace velmi snadno ovladatelné. Tento přístup však přináší i několik implementačních problémů.

4.2.1 Volba technologií

Hlavní otázka, která se u mobilní aplikace nabízí, je volba platformy. Platformně nezávislou implementaci pomocí Javascriptových knihoven jsem zavrhl, neboť ty neumožňují snadný přístup k hardwarovým funkcím jako Bluetooth nebo SMS. V tomto vycházím z vlastní zkušenosti, neboť vývoji mobilních aplikací na Javascriptové knihovně Cordova jsem se dva roky věnoval.

Jako cílovou platformu jsem zvolil Android, neboť je u uživatelů rozšířenější než alternativy v podobě iOS a Windows Phone [19] a relativně nízká pořizovací cena některých zařízení s OS Android je více v souladu s filozofií této práce, kterou je nabídnout co nejlevnější řešení.

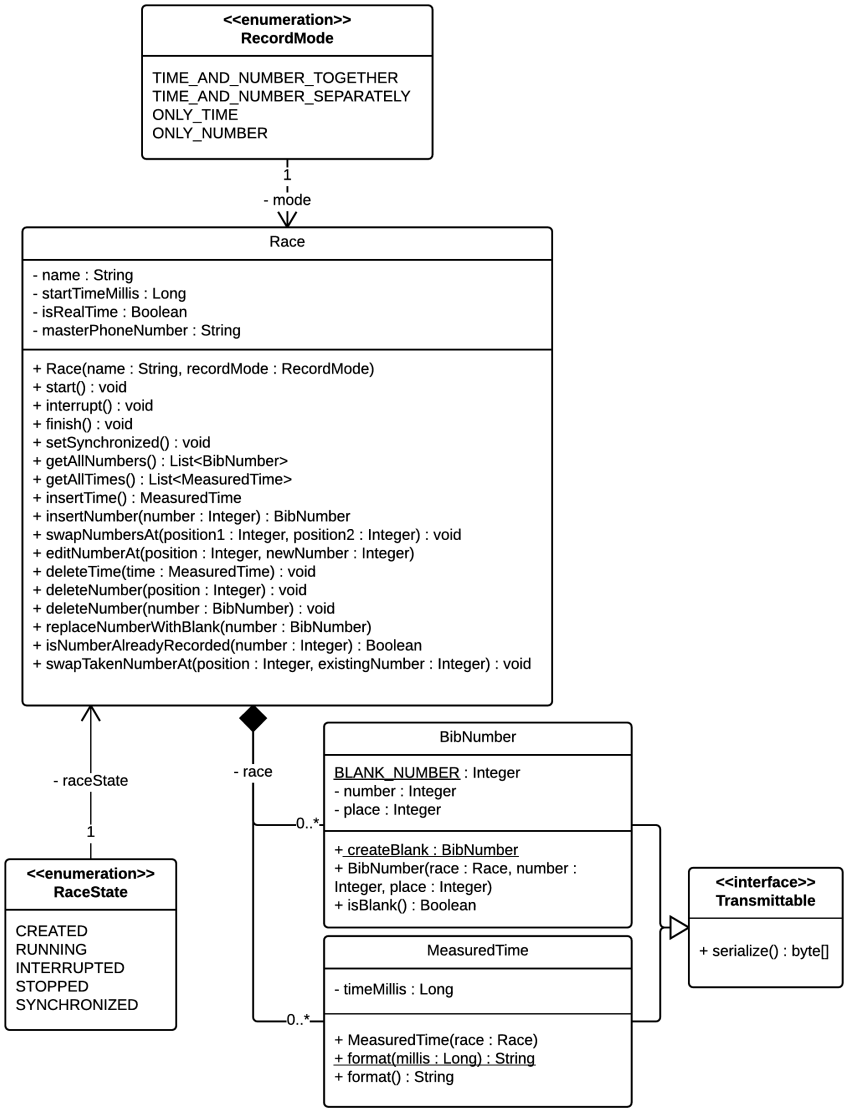
Stejně jako u desktopové aplikace jsem se rozhodl implementovat perzistentní ukládání dat prostřednictvím databáze SQLite. Na rozdíl od desktopových aplikací, v mobilním světě je toto běžná praxe a navíc zde ještě více než u desktopové aplikace potřebujeme zajistit, aby se neztratila data – například běžící časomíra – při násilném ukončení procesu. Nad databází používám jednoduchou mapovací knihovnu Sugar ORM [20].

4.2.2 Implementační detaily

Obrázek 4.5 ukazuje doménové třídy mobilní aplikace. Třídy obsahující stav jsou zde pouze tři, zbytek jsou pomocné enumerace a rozhraní.

Třída `Race` představuje jeden měřený závod, u něhož je ukládáno jméno a systémový čas v době spuštění časomíry, pokud už byla spuštěna. Pole **isRealTime** a **masterPhoneNumber** slouží ke komunikaci s desktopovou aplikací v reálném čase a k přenosu mezičasů hlavní časomíře a v současnosti nejsou implementována.

Nejdůležitější data ukládaná mobilní aplikací mají podobu tříd `BibNumber`, které představuje startovní číslo, a `MeasuredTime`, což je jeden změřený čas. Významná je absence vazby mezi těmito dvěma entitami. Mobilní aplikace umožňuje záznam čísel a časů zvlášť, je také možné využít dvou telefonů na jednom měřeném místě a zaznamenávat pouze čísla nebo pouze časy a sloučit je až v desktopové aplikaci. Jediná logická korelace mezi časy a čísly je tedy místo v pořadí, v jakém byla data pořízena. Rozhraní mobilní aplikace zob-



Obrázek 4.5: Diagram tříd domény mobilní aplikace

razuje čísla a časy v jediné tabulce, aby bylo poznat, co se po synchronizaci přiřadí k čemu, toto přiřazování je však výhodnější dělat až na tom místě v systému, kde je to nezbytně nutné. Jelikož mobilní aplikace nemá žádná data o registrovaných účastnících a věkových kategoriích, je zde výhodnější nechat data rozpojená.

Další rozdíl oproti desktopové aplikaci je v tom, že zde třída `Race` obsahuje velké množství metod. Nižší složitost mobilní aplikace eliminuje potřebu pro zvláštní business fasádu a veškerá manipulace dat se v mobilní aplikaci provádí výhradně v kontextu jednoho závodu, pro jednoduchost jsou tedy tyto metody seskupeny dohromady s entitní třídou představující závod. I tak je zachována myšlenka jediného přístupového bodu k celé doméně v zájmu přehlednosti a testovatelnosti a `Race` obsahuje i metody manipulující stavem ostatních dvou entit.

4.2.3 Uživatelské rozhraní

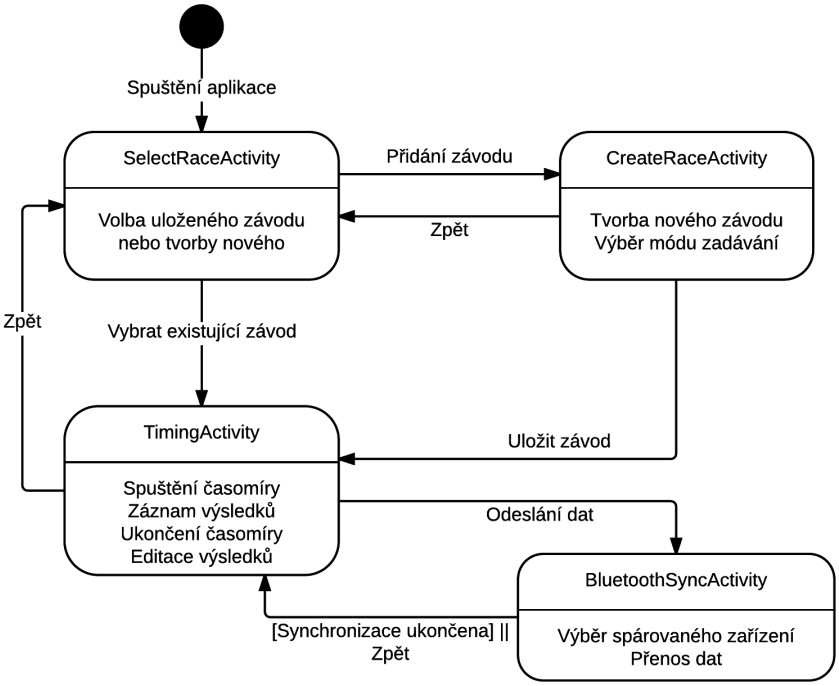
Uživatelské rozhraní v aplikacích pro OS Android se skládá z tzv. aktivit, což jsou sady grafických prvků, které při běhu aktivity zabírají celou obrazovku. Naše mobilní aplikace má aktivity čtyři a všechny jsou znázorněny na stavovém diagramu UML v obrázku 4.6. Přechody mezi aktivitami jsou vždy uživatelské akce, pokud nejsou uvedeny v hranatých závorkách. V takovém případě se jedná o událost vzniklou za běhu aplikace.

Následuje popis tří ze čtyř implementovaných aktivit. Poslední, `BluetoothSyncActivity`, je popsána v sekci 4.2.4.

SelectRaceActivity a CreateRaceActivity Tyto dvě aktivity slouží k výběru či tvorbě závodu, ke kterému chce uživatel zaznamenávat výsledky. Pro přehlednost se na seznamu závodů zobrazuje i stav každého z nich. Jeden z nefunkčních požadavků na mobilní měření času je schopnost obnovy měření i po nuceném ukončení aplikace. Na úvodní obrazovce aplikace lze tedy vybrat i ty závody, které uživatel dosud neukončil a po vybrání takového závodu se časomíra spustí znovu ze správného stavu vůči času odstartování.

Tvorba nového závodu je zde, na rozdíl od tvorby závodu v desktopové či webové aplikaci, velmi jednoduchá. Uživatel zadá pouze jméno závodu, jehož jediný účel je identifikace závodu v seznamu na předchozí obrazovce. Poté musí ještě vybrat jeden ze čtyř módů zadávání. Je zde možnost zadávat pouze čísla nebo pouze časy s tím, že pro správné zpracování výsledků v desktopové aplikaci je potřeba, aby byly použity telefony dva – každý z nich zaznamenávající jeden z těchto dvou typů údajů.

Je také možné zadávat časy i čísla dohromady. V takovém případě má ještě uživatel možnost si vybrat, zda chce spojit záznam čísla a času do jednoho úkonu, jako je tomu například u aplikace Race Timer, nebo zda bude zadávat čísla zvlášť a časy zvlášť. V druhém ze jmenovaných případů je pak



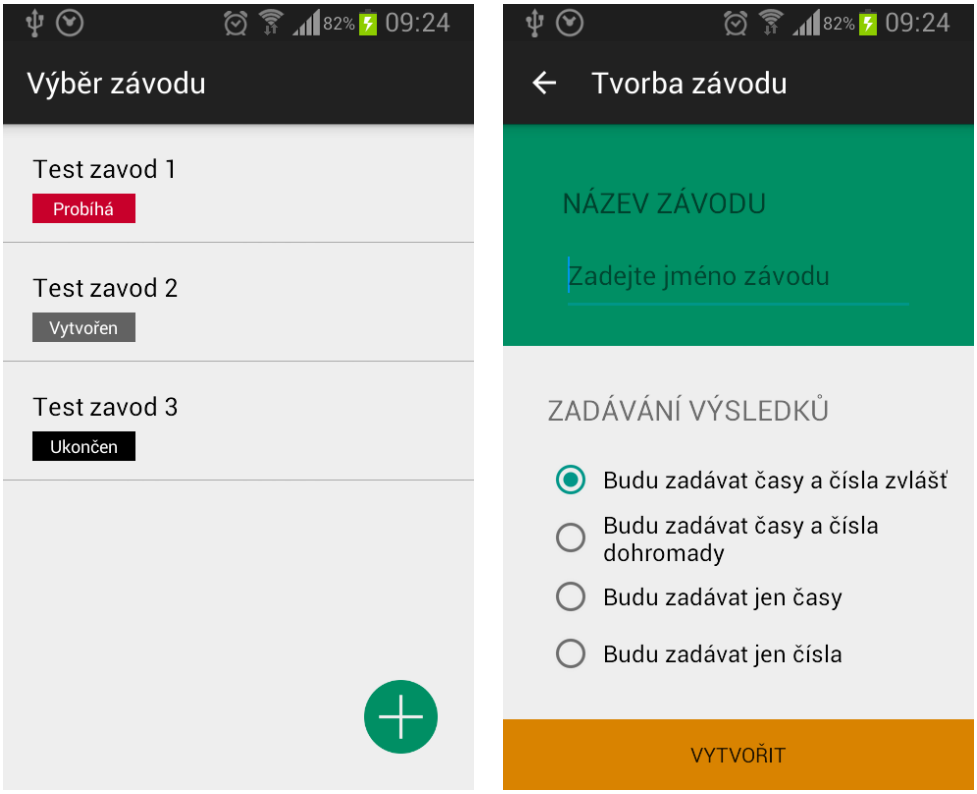
Obrázek 4.6: Stavový diagram GUI mobilní aplikace

záznam času ovládán jiným prvkem než záznam čísla. Po kliknutí na tlačítko „vytvořit“ přejde aplikace rovnou na `TimingActivity`.

TimingActivity Tato aktivita je stěžejním prvkem celé mobilní aplikace, neboť zde probíhá samotný záznam výsledků. Její uživatelské rozhraní je zobrazeno na levé polovině obrázku 4.2.3. Vévodí mu běžící časomíra a velká numerická klávesnice pro zadávání čísel, která kliknutím na tlačítko „schovat“ zmizí a udělá tak více prostoru seznamu čísel a časů, který se nachází nad ní. Tento seznam umožňuje mazat a upravovat pořízená data.

Podle toho, jaký si uživatel vybral mód zadávání při tvorbě závodu, se může rozhraní `TimingActivity` zobrazovat odlišně. Snímek obrazovky na obrázku 4.2.3 ukazuje rozhraní pro zadávání čísel a časů zvlášť. V případě, že uživatel zadává čísla spolu s časy, tlačítko „číslo“ se nezobrazuje. Pro zadávání pouze časů navíc rozhraní neobsahuje ani numerickou klávesnici. Pokud naopak uživatel zadává pouze čísla, nezobrazí se časomíra ani spodní tlačítko.

Zastavení běžící časomíry a přechod na aktivitu pro odeslání dat do desk-



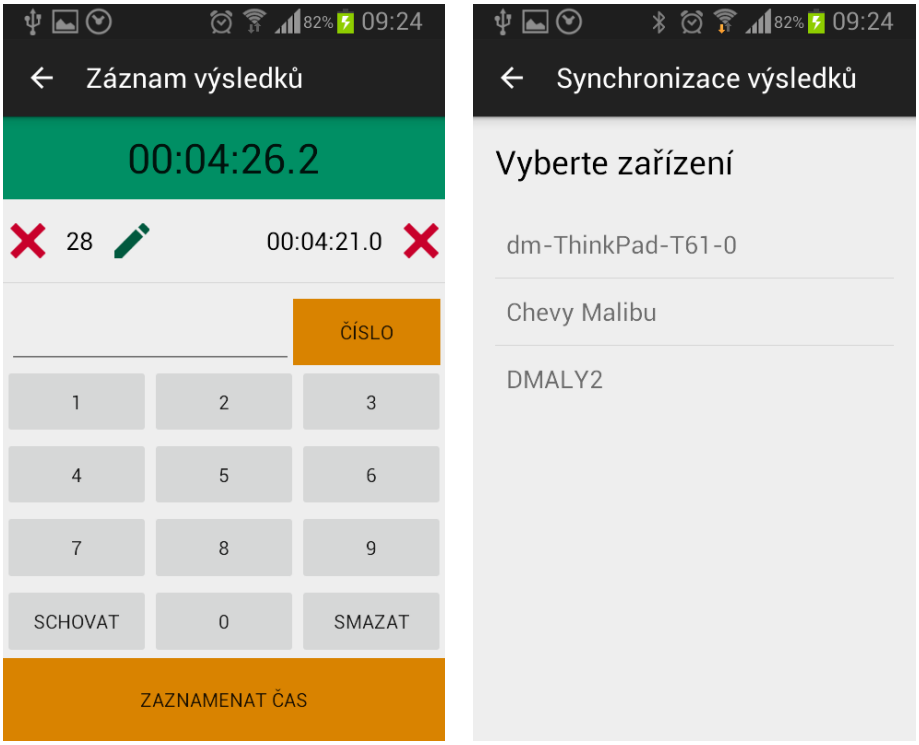
Obrázek 4.7: Výběr a tvorba nového závodu v mobilní aplikaci

topové aplikace je možné přes kontextové menu Androidu, ke kterému se přistupuje až do verze Android 5.0 pomocí hardwarového tlačítka.

4.2.4 Rozhraní s desktopovou aplikací

Rozhraní mobilní aplikace s desktopovou aplikací využívá technologie Bluetooth. Není to úplně ideální řešení, neboť pro fungování přenosu dat musí mít počítač, na kterém desktopová aplikace běží, Bluetooth adaptér. Tím jsou kladeny další nároky na použitý hardware. Původní záměr bylo využít rozhraní USB a přenášet data po kabelu, tato varianta se však ukázala být implementačně příliš náročná díky tomu, že API operačního systému Android vůbec nepočítá s tím, že by programátor chtěl přistupovat k USB komunikaci s počítačem.

Naproti tomu obsahuje Android široké možnosti využití Bluetooth adaptéru. Rozhodnutí využít Bluetooth jsem založil na tom, že všechny v současnosti používané chytré telefony a drtivá většina moderních notebooků Bluetooth podporují. Není příliš reálné, aby s sebou organizátor vozil na závod stolní počítač. Pokud přece jenom jedno z používaných zařízení Bluetooth nepodporuje, pořád umožňuje desktopová aplikace výsledky vkládat ručně.



Obrázek 4.8: Záznam výsledků a odesílání dat v mobilní aplikaci

4.2.4.1 Odeslání výsledků z mobilní aplikace

O komunikaci přes Bluetooth se v mobilní aplikaci stará aktivita `BluetoothSyncActivity`, jejíž uživatelské rozhraní je na obrázku 4.2.3 vpravo. Sestává pouze ze seznamu zařízení, z nichž si uživatel vybere počítač s běžící desktopovou aplikací, kde předtím spustil čekání na výsledky.

Aby mohla nějaká komunikace přes Bluetooth vůbec proběhnout, musí být obě zařízení spárována. Rozhodl jsem se v aplikaci neimplementovat mechanismus párování zařízení, neboť by to vyžadovalo speciální oprávnění, kterých se uživatelé často zaleknou a odmítnou aplikaci nainstalovat. Bylo by to navíc zbytečné, neboť párování je zcela jistě mnohem lépe implementováno v nastaveních telefonu.

Uživatel si tedy za předpokladu, že už má spárováno, vybere ze seznamu zařízení a aplikace se s ním pokusí navázat spojení. Aktivita vytvoří nové vlákno, třídu `BluetoothConnectThread`, které se snaží získat připojený RFCOMM socket. Jakmile se mu to povede, vlákno se ukončí a aktivita vytvoří nové. Tentokrát se jedná o instanci `BluetoothSyncThread`, které provádí přenos na pozadí, zatímco se uživateli zobrazuje dialog s indikací, že přenos probíhá (tj. točící se kolečko). Není zde potřeba žádné pokročilé synchronizace vláken, neboť aktivitu běžící na hlavním vlákně zajímá pouze výsledek pře-

nosu.

V případě neúspěchu v jakékoli fázi přenosu se zobrazí dialog s chybovou hláškou a aplikace se vrátí zpět na `TimingActivity`.

4.2.4.2 Schéma komunikace

Komunikace samotná probíhá přes protokol RFCOMM, který emuluje sériový port a poskytuje záruku spolehlivého přenosu dat ve správném pořadí, podobně jako například protokol TCP [21]. Pro RFCOMM existuje jak v knihovně PyBluez, tak v OS Android nejširší podpora. Navázání spojení probíhá přes Service Discovery Protocol (SDP), neboť je to běžná praxe a Android jeho použití vyžaduje. Desktopová aplikace nabízí službu s unikátním identifikátorem, který mobilní aplikace použije k nalezení této služby a navázání spojení.

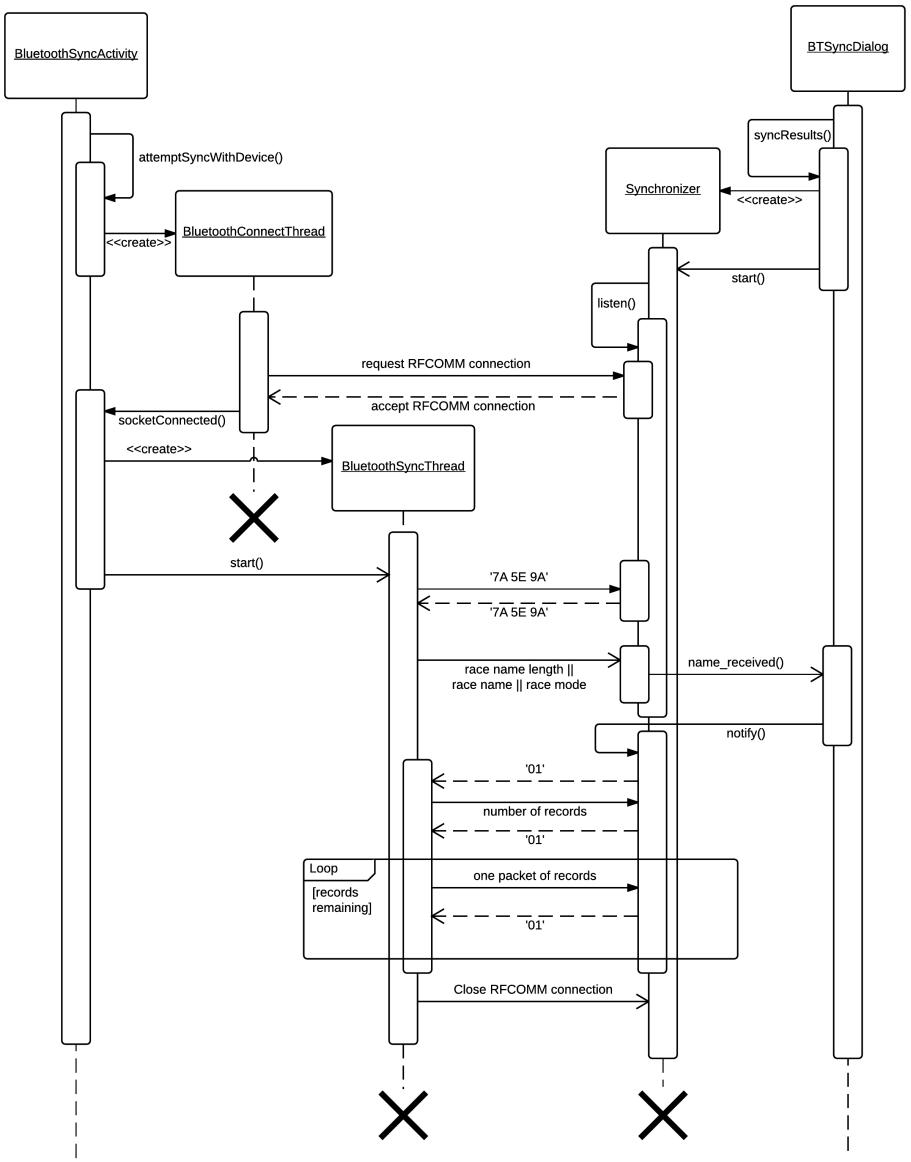
Obrázek 4.9 detailně popisuje schéma komunikace. Na levé straně je zobrazena mobilní aplikace a její synchronizační vlákna, na pravé straně desktopová aplikace. Celý proces je ještě doplněn o synchronizaci mezi vlákny a odchytávání chyb, zejména uvnitř jednotlivých komponent je tedy o něco složitější. Data, která jdou fyzicky přes Bluetooth, jsou však na diagramu zobrazena všechna.

Jako první je desktopovou aplikací vystavena SDP služba a vytvořen serverový RFCOMM socket, k němuž se klient (mobilní aplikace) připojí. Na začátku komunikace je oběma stranami odeslána sekvence tří bajtů, `7A 5E 9A`, jako další ověření na aplikační vrstvě, že vše funguje, jak má. Poté odešle mobilní aplikace tři údaje: délku jména závodu (4 bajty), samotné jméno závodu a mód zadávání (1 bajt). Mód můžou být časy i čísla dohromady, pouze časy, nebo pouze čísla.

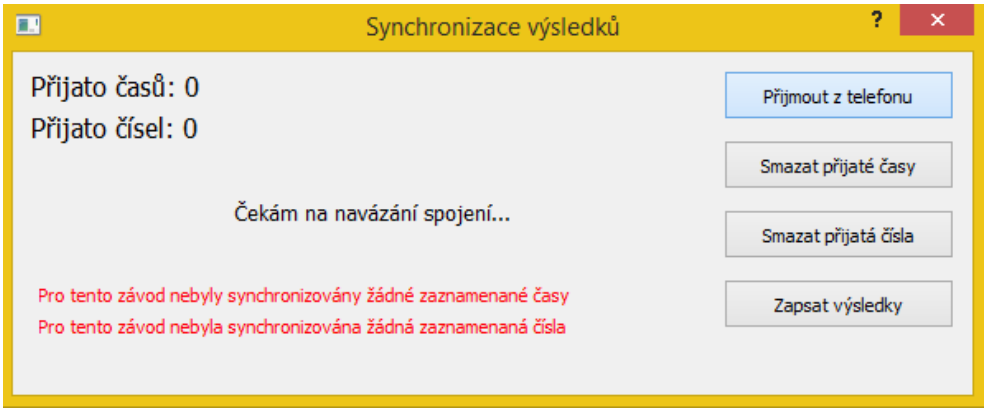
V tuto chvíli je prostor pro potvrzení ze strany desktopové aplikace, zda data na základě těchto informací chce přijmout. Serverové vlákno blokuje do té doby, než obdrží od hlavního vlákna potvrzení, že může pokračovat, nebo je zrušeno. Hlavní vlákno musí nějakým způsobem rozhodnout, zda má přenos proběhnout. V některých případech je vyžadováno potvrzení od uživatele (viz. následující sekce). Jakmile je přenos odsouhlasen, klientovi je poslán nazpátek potvrzovací bajt (`01`) a nastává fáze přenosu samotných záznamů.

Jako první klient odešle celkový počet záznamů, tedy počet startovních čísel plus počet naměřených časů (4 bajty), a opět čeká na potvrzovací paket. Poté posílá všechny záznamy už dohromady. Rozděluje je pouze tak, aby po odeslání jednoho plného Bluetooth paketu (1024 bajtů) vždy počkal na potvrzovací bajt od serveru.

Jeden záznam má vždy 10 bajtů. Na začátku je dvoubajtový identifikátor označující, o jaký druh záznamu jde. Poté následují v případě, že se jedná o startovní číslo, čtyři bajty samotného startovního čísla a čtyři bajty jako jeho pořadí v žebříčku. V případě změřeného času po identifikátoru následuje osm bajtů času v milisekundách od spuštění časomíry.



Obrázek 4.9: Sekvenční diagram Bluetooth synchronizace mezi mobilní a desktopovou aplikací



Obrázek 4.10: Příjem dat z mobilní aplikace

Jakmile jsou všechny záznamy odeslány a potvrzeny, spojení je uzavřeno bez dalšího zapojení aplikační vrstvy.

4.2.4.3 Přijímání výsledků v desktopové aplikaci

Z pohledu uživatele je přijímání dat v desktopové aplikaci o něco složitější než jejich odeslání z mobilu. Je to dáno tím, že na rozdíl od desktopu se mobilní aplikace ani nesnaží zajišťovat konzistenci dat, neboť jí k tomu chybí informace o registrovaných závodnících. Po přijetí těchto dat na desktopu je z tohoto důvodu potřeba je dočasně oddělit od „ostrých“ dat, dokud nejsou validována, případně opravena. Nevalidovaná data se musí ukládat, protože aplikace umožňuje synchronizovat část dat z jednoho zařízení a druhou část z jiného zařízení. Třída `RaceSyncData` v doménovém modelu desktopové aplikace (viz. obrázek 4.1) sdružuje přijatá čísla a časy, jejichž struktura kopíruje schéma ukládání dat v mobilní aplikaci.

Přístup k serverové části Bluetooth synchronizace je umožněn skrz tlačítko na pracovní ploše výsledků závodu (viz. obrázek 4.4). Kliknutím na toto tlačítko se otevře dialog synchronizace jako na obrázku 4.10. Zde může uživatel spustit čekání na připojení mobilní aplikace, může smazat již jednou přijaté sady dat a může se pokusit o aplikování přijatých dat na reálné registrované závodníky.

Čistému aplikování přijatých dat může bránit několik druhů nekonzistencí. Může se stát, že je přijato buď víc časů než čísel nebo naopak. Může se také stát, že je zaznamenáno číslo, které v závodě není nikomu přiřazeno. Aplikace vždy nabídne uživateli možnost nekonzistenci jednoduše vyřešit. V případě odlišných počtů záznamů se mohou zahodit ty, které po seřazení přebývají na konci. Pokud bylo přijato špatné startovní číslo, může uživatel záznam opravit. Pokud nejsou tato řešení dostačující, může celou sadu přijatých dat smazat, opravit je v mobilní aplikaci a synchronizovat je ještě jednou. Případné

chybějící záznamy pak může vložit ručně.

Jakmile data neobsahují žádné nekonzistence, jsou aplikována na zbytek doménového modelu stejným mechanismem, jako kdyby je uživatel vložil ručně.

4.3 Webová aplikace

Webová aplikace je nerozsáhlejší část systému, přestože pokrývá méně důležité funkční požadavky než desktopová i mobilní aplikace. Správně implementovat webovou aplikaci úplně od základu již v dnešní době není triviální záležitost, neboť existuje celá řada zaběhnutých mechanismů, které uživatelé od aplikace čekají.

Na druhou stranu je webová aplikace tou částí systému, kde existují nepřeberné možnosti z hlediska dalšího rozšiřování. Můžeme například implementovat placení startovného po internetu, uživatelské profily pro běžce, kde se budou shromažďovat výsledky závodů a zobrazovat statistiky, interaktivní mapy tras a desítky dalších funkcí. Při implementaci jsem se tedy soustředil na robustní základ systému – uživatele a autentizaci, databázové schéma, předběžné registrace a maximální rozšiřitelnost.

V důsledku toho se implementace zatím příliš nezabývala grafickou podobou stránek a navenek vypadá aplikace spíše jako prototyp, přestože je všechna základní funkcionalita na svém místě a otestována. Věřím, že díky pevným základům bude další vývoj aplikace efektivní a bezbolestný.

4.3.1 Volba technologií

Rozhodl jsem se postavit webovou aplikaci na známé a prověřené knihovně Django [22] pro programovací jazyk Python. Rozhodnutí použít webový framework vzešlo z názoru, že je zbytečné znovu implementovat něco, co už je vyřešené desítkami způsobů, ze kterých si lze vybrat. V některých případech, jako například u implementace systému přihlašování a ukládání hesel, je to dokonce nebezpečné. Použití Pythonu bylo také jednoduché rozhodnutí. Aplikují se zde stejné výhody – vysokoúrovňovost, dynamické typování, expresivita – jako u desktopové aplikace, navíc jsem při její implementaci získal s Pythonem mnoho zkušeností. Nabízí se zde také možnost znovupoužití částí kódu tam, kde mají obě aplikace společnou logiku.

Pro Python existují tři hlavní knihovny pro webový vývoj: Django, Flask a Pyramid. Článek [23] uvádí jejich základní charakteristiky spolu s podrobným srovnáním. Vyplývá z něj, že Django obsahuje více funkcionalit a nástrojů, ale klade větší nároky na strukturu projektu než Flask i Pyramid. Zatímco dvě poslední jmenované knihovny vynikají jednoduchostí a rozšiřitelností, přednost Djanga je funkční kompletnost.

Vyvíjená webová aplikace nemá žádné zvláštní nároky na strukturu nebo nestandardní rozšiřování. Rozhodl jsem se použít Django, neboť se tím vyhnu

přidávání dalších knihoven, například pro ORM, které spolu nemusí správně pracovat. Django má také přehlednější dokumentaci, větší internetovou komunitu a existuje pro něj více doplňků. Jedním takovým doplňkem je django-rest-framework, který rozšiřuje systém stránek Djanga pro použití v RESTových webových službách a v této aplikaci jsem ho využil k postavení API pro připojení desktopové aplikace.

Databáze pro webovou aplikaci už v tomto případě není SQLite, neboť zde může jeho neschopnost zpracovávat paralelní přístupy znamenat zpomalení aplikace při větší zátěži. Zvolil jsem MySQL, databázi vyvíjenou a optimalizovanou pro webové aplikace.

Klientská část webové aplikace, čili kaskádové styly a Javascript pro zpříjemnění vzhledu a fungování stránek, se opírá o knihovny JQuery [24] a Bootstrap [25]. Ty řeší některé problémy s kompatibilitou webových prohlížečů a výrazně snižují časové nároky na vývoj webového front-endu, který je v tomto vývojovém stádiu nejméně důležitým aspektem systému.

4.3.2 Implementační detaily

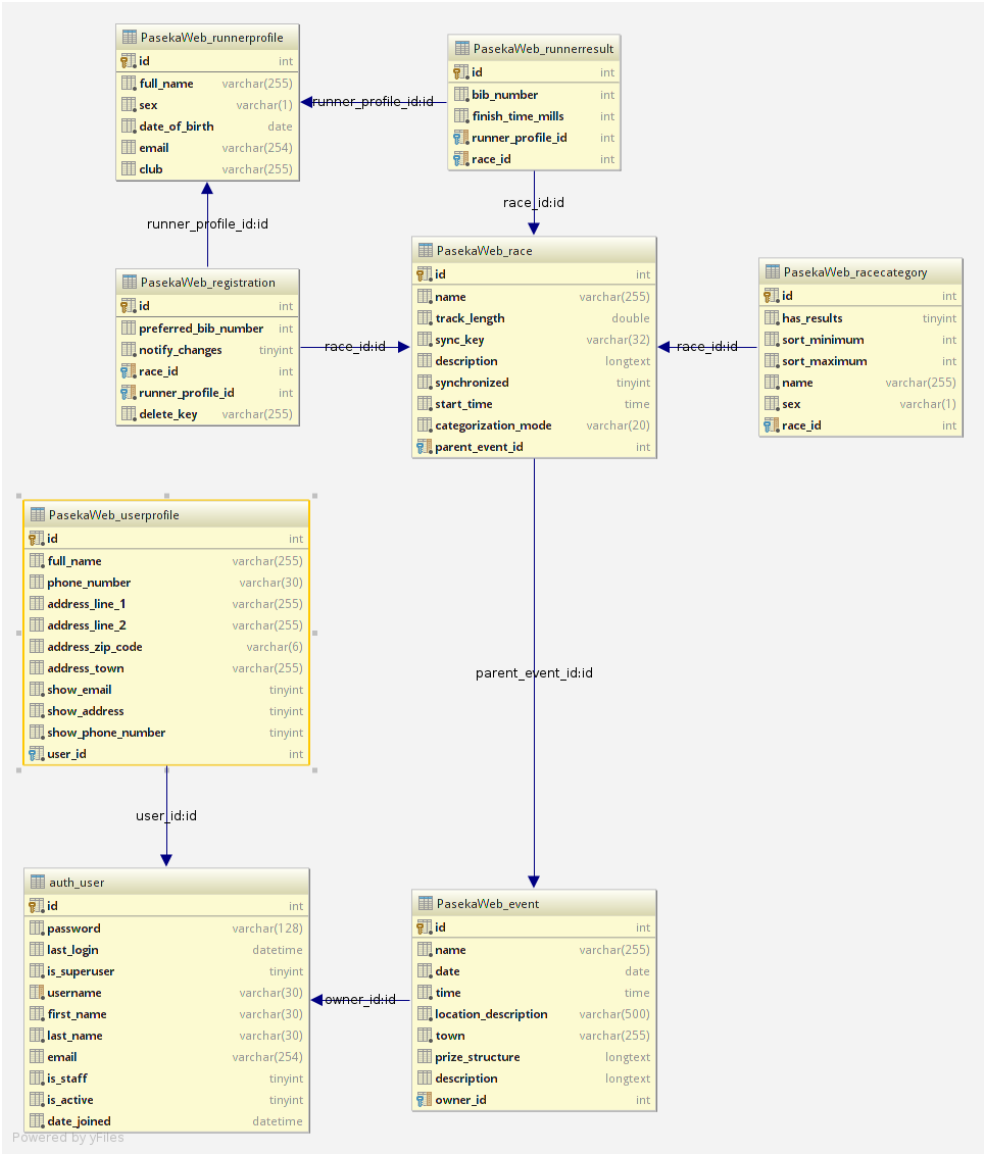
Detailní popis implementace webové aplikace začnu i tentokrát popisem doménového modelu. Ten je realizován databázovým schématem vzešlým z modelových tříd definovaných v rámci frameworku Django. Poté popíšu jednotlivé části a stránky aplikace a nakonec se budu věnovat integraci s desktopovou aplikací.

4.3.2.1 Databázové schéma

E-R diagram databázového schématu je ukázán na obrázku 4.11. Django ještě do schématu přidává další pomocné třídy zejména pro správu povolení a uživatelských skupin, na diagramu jsou však zachyceny jenom entity specifické pro tuto aplikaci. Výjimkou je tabulka `auth_user`, která je znázorněna pro ukázání vazeb, které na ni mají ostatní entity.

V horní části diagramu se nachází tabulky představující jádro doménového modelu v podobném složení jako u desktopové aplikace. Je tu jedna tabulka pro závody, které mají každý několik registrací. Jeden z rozdílů oproti desktopové aplikaci je ten, že se registrace váží přímo na závod a ne na věkovou kategorii. Tímto je umožněno, aby pořadatel mohl kategorie ještě změnit na webu poté, co se do něj účastníci registrují.

Entita `Registration` představuje prozatím pouze předběžnou registraci a výsledek závodu, `RunnerResult`, se na ni nijak nevztahuje. Na diagramu by se mohlo zdát, že jsou tyto dvě tabulky spojené tabulkou `RunnerProfile`, ale není tomu tak, neboť `RunnerProfile` nemá v aplikaci vlastní identitu. Slouží jenom jako agregátor vlastností, které mají registrace i výsledek společné. V budoucnosti by samozřejmě bylo lepší, aby registrace a výsledek nebyly na sobě nezávislé. Bude potřeba vyřešit několik potenciálních zdrojů nekonzis-



Obrázek 4.11: E-R diagram databázového schématu webové aplikace

tencí, například uživatelé, co se neregistrovali na závod přes web, ale získá se tím možnost zřídit uživatelský profil účastníka.

Ve spodní části diagramu je zakreslena entita `Event`. Ta modeluje sportovní událost, při níž je pořádáno více závodů ve stejný den. Událost je pro webovou aplikaci důležitějším pojmem než závod, neboť uživatelské rozhraní zobrazuje především události. Jednotlivé závody jsou pak viditelné až na detailu akce. Většina akcí bude mít nejspíš pouze jediný závod, přidání této funkcionality však pokrývá častý případ užití a na uživatele nemá žádný nepříznivý dopad.

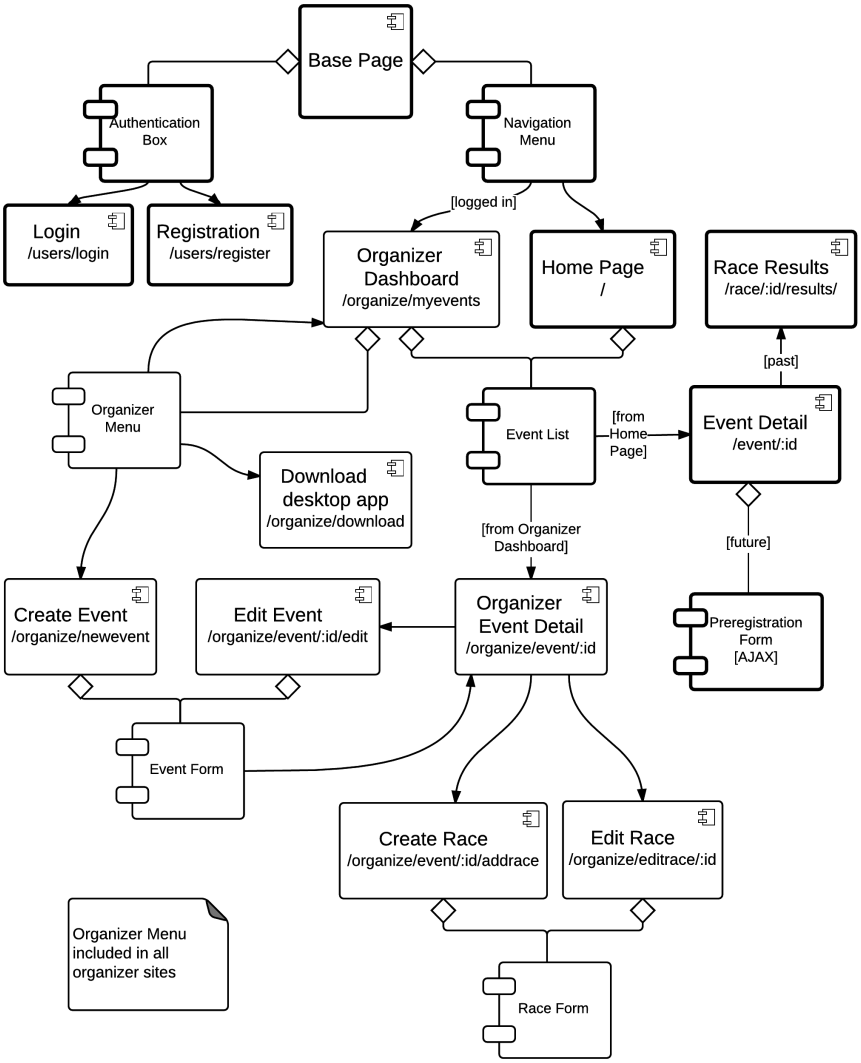
Zbývá ještě zmínit entity `User` a `UserProfile`. První jmenovaná patří do autentizačního modulu frameworku Django a představuje uživatele, který se může do aplikace přihlásit. V současnosti existuje jediný druh registrovaného uživatele, a sice pořadatel, autentizační mechanismus však lze snadno rozšířit. Tabulka `UserProfile` existuje, protože Django neumožňuje rozšířit informace ukládané o uživatelích jinak než specializovanou tabulkou se vztahem jedna ku jedné s uživatelem. Jelikož má aplikace zobrazovat kontaktní údaje organizátora, je potřeba si tyto údaje o pořadatelích ukládat.

4.3.2.2 Webové rozhraní

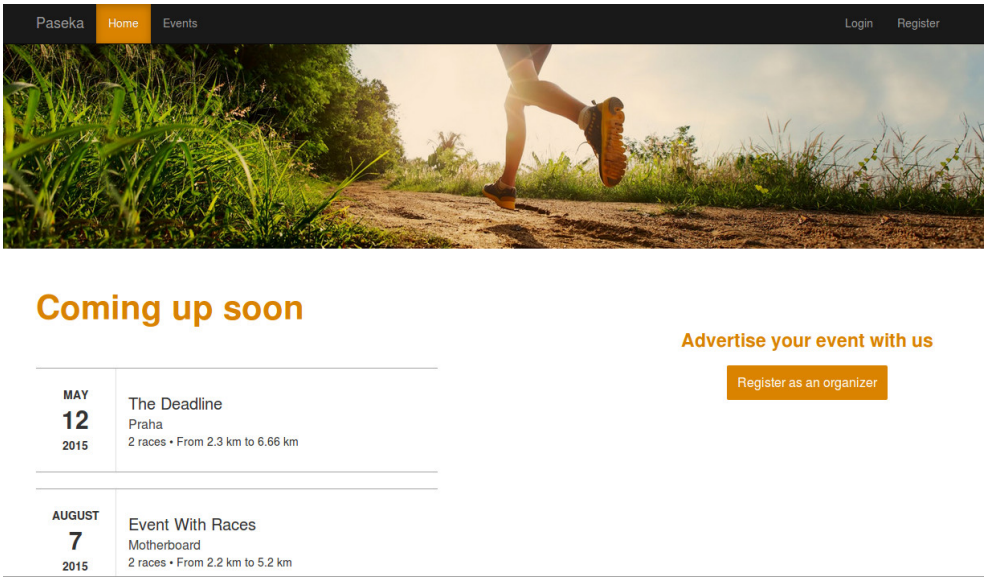
Webová aplikace nemá zvláštní business vrstvu, která by se nijak nezabývala prezentací, neboť je manipulace s daty mnohem více vázaná na to, na které stránce se uživatel zrovna nachází a co přesně přišlo od jeho prohlížeče za požadavek. Navíc framework Django poskytuje mechanismus formulářů a pohledů („views“), které abstrahují konstrukci HTML odpovědí do tzv. šablon. Ty jsou definovány ve zvláštních souborech. Kód odpovídající na požadavky se tedy může zabývat rovnou vykonáváním business logiky s tou výhodou, že má přístup k celému kontextu požadavku. Jako odpověď pak pošle data do nějaké šablony, Django ji přeloží do HTML a odešle zpět. Dokumentace stránek a uživatelského rozhraní webové aplikace je tedy zároveň dokumentací logiky manipulující s doménovými entitami.

Nepodařilo se mi najít jeden zavedený a akceptovaný postup, jak staticky zdokumentovat strukturu jednotlivých stránek složitější webové aplikace. Ideálně by k tomu posloužil interaktivní prototyp, já jsem se rozhodl pro tyto potřeby adaptovat UML diagram komponent. Výsledek je vidět na obrázku 4.12. Obdélníky s obrázkem komponenty v pravém horním rohu představují jednotlivé stránky. Je v nich uveden název či popis, k čemu slouží, a URL adresa. Komponenty bez URL adresy jsou nějaké kusy uživatelského rozhraní, které mohou být zahrnuty ve vícero stránkách. Lomené čáry končící kosočtvercem znázorňují komponentu vloženou ve stránce, čáry končící šipkou označují možný přechod z jedné stránky či komponenty na jinou stránku.

Webová aplikace je jako jediná část systému dostupná ve dvou jazykových verzích. Anglická verze používá URL adresy začínající řetězcem `/en/`, česká je dostupná pod adresami začínajícími na `/cs/`. Tyto jazykové identifikátory



Obrázek 4.12: Navigační struktura stránek webové aplikace



Obrázek 4.13: Domovská stránka webové aplikace

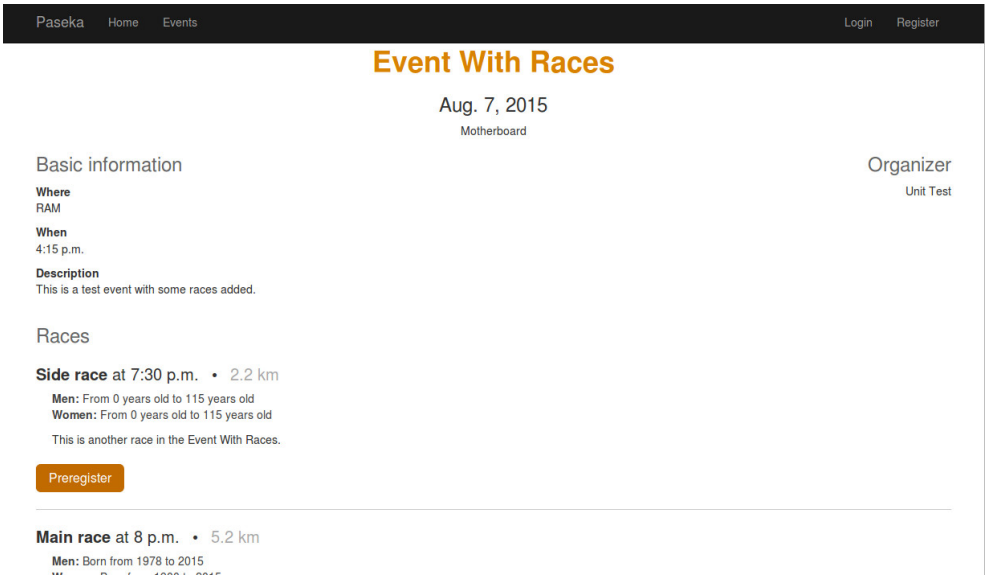
následují hned po internetové doméně, URL adresy uvedené v diagramu tedy začínají až za nimi.

Všechny stránky v aplikaci „dědí“ od Base Page, obsahují tedy navigační nabídku spolu s odkazy pro přihlášení, registraci a odhlášení (není znázorněno). Stránky obtažené silnější čarou jsou přístupné všem návštěvníkům, ostatní stránky jsou určeny pouze registrovaným pořadatelům. Všechny organizátorské stránky obsahují komponentu Organizer Menu, tato vazba není pro přehlednost znázorněna.

Z hlediska realizace v kódu je každá znázorněná stránka jedním objektem typu `View` ve frameworku Django. Komponenty jsou buď Django formuláře nebo jen HTML šablony, do kterých každá stránka pošle jiná data.

Webová aplikace má kromě znázorněných stránek ještě další součásti. Jedná se o stránky, které mažou závody, akce a předregistrace. Ty pouze vykonají svůj úkon a odkážou prohlížeč jinam. Podobně funguje i stránka, která zpracovává AJAX formulář pro předběžnou registraci. Ta jen odešle zpět případné chyby ve validaci formuláře.

Stránky samotné jsou velmi přímočaré. Veřejná část pro účastníky závod se skládá z domovské stránky (obr. 4.13), která obsahuje výběr nejbližších nadcházejících událostí. Z navigačního menu vede odkaz ještě na dedikovaný výpis nadcházejících i proběhlých akcí, který umožnuje vyhledávání a filtrování podle data konání. Tento výpis není na diagramu zobrazen. Z výpisů akcí se uživatel dostane na detail události (obr. 4.14), kde uvidí propozice a jednotlivé vypsané závody. Pod tlačítkem u názvu závodu se skrývá registrační formulář, který po kliknutí na tlačítko vyjede. Odeslání tohoto formuláře je vyřešeno



Obrázek 4.14: Detail závodu ve webové aplikaci

přes AJAX požadavek, aby se uživateli nemusela znovu načítat celá stránka.

Úspěšně zapsanému účastníkovi je odeslán e-mail s potvrzením registrace a také s odkazem, přes který může svou registraci zase smazat. Pokud navíc zaškrtnul volbu, že chce být informován o případných změnách v propozicích a pořadatel změní nějaký údaj buď akce nebo závodu, dostane účastník další e-mail s upozorněním na změny.

V případě, že datum konání akce již proběhlo, registrační formuláře se neobjeví. Pokud již pořadatel nahrál výsledky závodů, zobrazí se místo nich odkazy na výsledky jednotlivých závodů. Na stránce výsledků si může uživatel vybrat, kterou chce vidět kategorii, a zobrazení této stránky je optimalizováno pro tisk.

Hlavní stránka pro organizátora závodů obsahuje seznam akcí, které v systému založil. Detail akce pro jejího organizátora je velmi podobný stránce určené pro neregistrované návštěvníky. Jsou zde vidět údaje o akci a seznam závodů, navíc jsou zde však i odkazy pro editaci propozic akce a odkazy pro přidání, editaci a mazání závodů. Ty vedou na jednoduché formuláře, které po uložení vrátí uživatele zpět na detail akce.

U formuláře pro tvorbu a editaci jednotlivých závodů bylo potřeba implementovat definici a validaci věkových kategorií. Django pochopitelně žádný podobný formulářový element neobsahuje, musel jsem tedy implementovat speciální formulářové pole, `RaceCategoryField`, spolu s jeho HTML zobrazením, kódem pro validaci a klientskou logikou v Javascriptu. Formulář pro editaci či tvorbu závodu obsahuje dvě taková pole – jedno pro mužské kategorie a jedno pro ženské. Po odeslání formuláře se provádí ještě validace napříč

oběma prvky pro ověření, že všechny názvy kategorií jsou unikátní.

4.3.3 Synchronizace s desktopovou aplikací

Komunikace mezi desktopovou a webovou aplikací je vyřešena skrze webové aplikační rozhraní (Web API) postavené podle architektury REST tak, jak ji definuje [26]. Je to tedy datově orientovaná služba využívající protokolu HTTP.

Využití HTTP je jediné logické řešení, neboť webová aplikace už na HTTP serveru běží a k jiným protokolům či nižším vrstvám komunikace by přistupovala jen obtížně. Rozhraní pro desktopovou aplikaci je tedy jen variací na rozhraní pro klasický prohlížeč. REST je pro toto využití nejvhodnějším řešením, neboť je komunikace orientovaná na data. Účelem je posílat informace o závodě jedním směrem a výsledky závodu směrem druhým. Procedurově orientované architektury (například SOAP) jsou navíc zbytečně složité pro rozhraní, které definuje pouhé tři koncové body. Konečně pro framework Django existuje výborná knihovna pro tvorbu REST rozhraní, django-rest-framework.

Serverové API Rozhraní využívá autentizační schéma HTTP Basic, všechny požadavky na REST rozhraní musí tedy obsahovat autentizační hlavičku s uživatelským jménem a heslem pořadatele. Tento způsob není přes nezašifrované spojení bezpečný, v produkčním prostředí bude potřeba opatřit server certifikátem a vynucovat použití HTTPS.

API definuje tři možné požadavky:

- `GET /api/races/`
  Vrátí seznam všech závodů, které na webu založil pořadatel, jehož autentizační údaje jsou v požadavku. Jelikož desktopová aplikace nezná koncept akcí, informace o akci, pod níž je daný závod založen, jsou vloženy mezi informace o závodě. Tento požadavek odesílá desktopová aplikace pro zobrazení seznamu závodů, z nichž si uživatel může vybrat ten, který chce synchronizovat. Data vrácená tímto požadavkem neobsahují informace o věkových kategoriích ani registrovaných běžcích. Pro získání těchto informací slouží následující požadavek.

- `GET /api/races/:id/`
  Vrátí detailní informace o závodě, jehož hodnota `id` je v URL. Vlastníkem závodu musí být uživatel, jehož autentizační údaje jsou v požadavku. Tento požadavek vrací i vnořená data o kategoriích spolu se seznamem předběžných registrací. Takto desktopová aplikace „stahuje“ závod z webu.

- `POST /api/races/:id/results/`
  Uloží výsledky závodu uvedené v těle požadavku. Vlastníkem závodu musí být uživatel, jehož autentizační údaje jsou v požadavku. Pokud

již webová aplikace jednou výsledky k danému závodu přijala, staré výsledky jsou smazány. Tímto požadavkem odesílá desktopová aplikace výsledky na web. Samotné záznamy v těle požadavku obsahují startovní číslo, změřený čas a registrační údaje účastníka. Neobsahují informace o konečném pořadí a přiřazení do kategorie, tyto údaje si webová aplikace vypočítá sama.

Klientská část Iniciace přenosu dat mezi webovou a desktopovou aplikací se děje z desktopového uživatelského rozhraní. Hned po otevření aplikace se zobrazí menu, jehož poslední položka má popis „Stáhnout závod z webu“. Po vybrání této volby se uživateli zobrazí dialog pro zadání autentizačních údajů na web. Jakmile je aplikace dostane, pokusí se o stažení seznamu uživatelových závodů pomocí prvního ze tří popsaných požadavků. V případě, že se seznam podaří stáhnout, zobrazí se podobné okno jako při výběru lokálně uloženého závodu – v obou případech se využívá stejné třídy, `SelectRaceDialog`.

Vybere-li si uživatel závod pro stažení, aplikace odešle požadavek na detaily vybraného závodu a přijme tak informace o věkových kategoriích a předběžných registracích. Uživatel už nemusí znovu zadávat autentizační údaje, neboť ty si aplikace drží od minulého požadavku. Přijatá data aplikace rovnou uloží do lokální databáze a otevře kontext právě staženého závodu.

Výhoda předběžných registrací spočívá především v tom, že umožňují zrychlit proces registrace přímo na místě závodu. Desktopová aplikace toto realizuje předvyplněním registračního formuláře údaji staženými z webu. V tabulce ve spodní části kontextu závodu lze zvolit zobrazení s názvem „předběžně registrovaní“. Dvojitým kliknutím na nějaký řádek v této tabulce se přenesou údaje o běžci do registračního formuláře, takže je pořadatel nemusí vyplňovat ručně. Další uložení registrace pak probíhá stejným způsobem jako u klasických registrací.

Odeslání výsledků závodu na web je možné až po úplném ukončení závodu v desktopové aplikaci, aby se předešlo nečekaným změnám výsledků po jejich zveřejnění. Tlačítko „Odeslat výsledky“ se tedy objeví až na úplně posledním kroku životního cyklu závodu. Jeho aktivování vede opět na dialog s formulářem pro zadání přihlašovacích údajů, po jehož odkliknutí se ihned spustí odesílání. Není zde potřeba žádného výběru závodu jako při stahování, protože spolu se staženými závody je ukládán jejich primární klíč na webu. Aplikace tak automaticky ví, ke kterému závodu má výsledky posílat.

U odesílání může nastat situace, kdy závod není stažen z webu, a přesto se uživatel pokusí odeslat na web výsledky. V takovém případě vrátí webová aplikace HTTP kód 404 a uživatel je na tuto skutečnost upozorněn.

4.4 Shrnutí

V této kapitole jsem popsal realizaci architektury navržené v kapitole 3. Během realizace nevyvstaly žádné větší komplikace a všechny tři hlavní komponenty dohromady realizují systém, jenž z funkčního hlediska splňuje hlavní definované požadavky. To ukazuje na správnost zvolené architektury vzhledem ke zmíněným požadavkům. Technologie zvolené pro realizaci jednotlivých komponent jsou volně dostupné a tudíž v souladu s filozofií systému zachovat co možná nejnižší náklady na provoz a pořízení ze strany pořadatelů závodů.

Ze seznamu funkčních požadavků v kapitole 3 byly implementovány požadavky s prioritou 1, 2 a 3. Přestože systému některé kusy funkcionality chybí, mobilní, desktopová i webová aplikace mohou být využity již v současném stavu k pořádání běžeckého závodu bez přílišných obav ze ztráty dat či nedostačujících schopností. Komunikační mechanismy mezi jednotlivými částmi systému jsou robustní a mohou být považovány za hotové. Následující kapitola o testování tato tvrzení ověřuje.

Kapitola 5

Testování

Nedílnou součástí vývoje jakéhokoli informačního systému je automatizované a ruční testování výsledného produktu. Systém vyvíjený v této práci má oproti některým jiným softwarovým produktům vyšší požadavky na spolehlivost, neboť mnoho úkonů lze v běžeckém závodě provést jen jednou a oprava může být obtížná či nemožná. Jedná se především o záznam časů běžců, ale i selhání systému při probíhající registraci může mít nepříjemné důsledky pro celý závod.

Uskutečněné testy vyvíjeného systému lze rozdělit do tří skupin. Unit testy nebo také jednotkové testy ověřují správné fungování oddělených částí systému na úrovni metod a tříd. Integrační testy se zabývají komunikací mezi jednotlivými komponentami a systémové testy prověřují celý systém z pohledu uživatele a jako jediné ze zmíněných tří typů jsou prováděny ručně.

5.1 Unit testy

Implementace desktopové i mobilní aplikace probíhala vždy metodou zdola nahoru, neboli od reprezentace dat až po kompletní uživatelské rozhraní. Vyšší vrstvy implementace se tedy nabalovaly na nižší. Potřeba psát automatizované testy, které ověřují funkčnost oddělených částí kódu, vznikla přirozeně z tohoto postupu, protože bez existence GUI nebyla možnost ručního testování. Pokrytí business a databázové logiky těchto dvou částí systému jednotkovými testy je téměř stoprocentní. Naproti tomu pro uživatelské rozhraní unit testy psány nebyly.

U webové aplikace unit testování nižších vrstev kódu nebylo nutné, protože žádné nižší vrstvy s netriviální logikou webová aplikace nemá. Testy psané pro web pracují už s celou aplikací a spadají tak spíše do kategorie automatizovaných systémových testů.

5.1.1 Desktopová aplikace

Unit testy desktopové aplikace využívají základní testovací knihovny pro Python, `unittest`. Testy se nachází v adresáři `paseka/test/unit` a sestávají z pěti skriptů plus společných testovacích dat.

První ze sad testů, `TestPersistenceLayer`, ověřuje funkčnost napojení na knihovnu Peewee a správnou definici doménového modelu. Pokusí se vytvořit SQLite databázi v paměti a uložit závod spolu s registrací. Všechny další testovací sady ověřují správné fungování jednotlivých funkcí modulu `services.py`, neboli business vrstvy. Každá netriviální funkce má svůj unit test. Výjimku tvoří sada `TestSyncServices`, která testuje několik celých scénářů přijetí dat od mobilní aplikace.

Aby bylo docíleno větší objektivity unit testů, využívá testovací kód desktopové aplikace tzv. „mock“ objektů. Tyto objekty jsou vytvářeny testovací knihovnou a při testování nahrazují závislosti testovaných objektů. Zde jsou například mock objekty využívány k nahrazení databázové vrstvy. Výhoda mock objektů spočívá v tom, že si může být autor testu jistý, že testuje chování opravdu jen jednotky, pro kterou test píše. Chování mock objektů si definuje sám a může pak ověřovat, zda interakce s mock objektem ze strany testované jednotky proběhla tak, jak měla. Mock objekty nejsou využívány u všech testů, neboť některé business metody vrací rovnou výsledek dotazu do databáze. V takovém případě nás zajímá spíše správnost vrácených dat než syntaxe dotazu.

5.1.2 Mobilní aplikace

Pro unit testování mobilní aplikace byly využity standardní testovací nástroje platformy Android. To zahrnuje především testovací knihovnu JUnit pro programovací jazyk Java, adaptovanou pro testování mobilních aplikací v rámci Android SDK. Všechny testy prochází nad běžící aplikací v Android emulátoru s verzí operačního systému 5.0.

Mobilní aplikace obsahuje 8 testů, z nichž jeden opět testuje správné fungování lokální databáze a zbývající testy ověřují fungování metod třídy `Event`, která tvoří fasádu pro manipulaci s uloženými daty. Je důležité testovat ukládání čísel a časů pro všechny možné módy zadávání, neboť pro každý z nich se aplikace chová trochu jinak.

V mobilní aplikaci je méně ostře definováno rozdělení mezi databázovým a business kódem, takže nejsou vůbec potřeba mock objekty. Testy si vystačí s běžící testovací databází.

5.2 Integrační testy

Integrační testy mají za úkol ověřit správnou funkčnost napojení jednotlivých komponent systému mezi sebou. Testují správnost a konzistenci rozhraní na

koncových bodech, komunikaci mezi komponentami a výstup z modulů, které se starají o serializaci a deserializaci dat.

Testovaný systém obsahuje dvě implementovaná rozhraní mezi komponentami: Bluetooth synchronizační mechanismus pro přenos výsledků z mobilního telefonu do PC a webové REST API pro komunikaci mezi webovou a desktopovou aplikací. Pro obě tato rozhraní byly napsány důkladné automatizované testy, jejichž struktura a testovací strategie se mezi sebou mírně liší.

5.2.1 Bluetooth synchronizace

Automatizovaný test Bluetooth rozhraní mezi mobilní a desktopovou aplikací je implementován pomocí klasického unit testu na každé z obou stran komunikace. Nejdříve je potřeba jako unit test spustit skript `test_bluetooth_sync` na desktopu, kde se nastartuje Bluetooth server tak, jako je tomu u ostrého provozu. V tuto chvíli je možné spustit stejnojmenný test i na mobilní aplikaci.

Na straně mobilní aplikace je situace trochu komplikovanější. Na rozdíl od unit testů nelze spustit integrační test na Android emulátoru, protože ten vůbec nepodporuje přístup k hostitelskému Bluetooth adaptéru. Je potřeba tedy k počítači připojit opravdové Android zařízení se zapnutým Bluetooth adaptérem, které bylo navíc s počítačem, kde běží serverový test, předem spárováno. V kódu mobilního testu je nutné ještě změnit konstantu s identifikátorem serveru, aby aplikace věděla, ke kterému zařízení se má pokusit připojit. Celý zbytek testu už proběhne automaticky.

Samotný průběh testu kopíruje schéma komunikace na diagramu 4.9. Mobilní aplikace vytvoří testovací závod, vloží do něj několik naměřených výsledků a pokusí se je přes Bluetooth odeslat. Desktopová aplikace testuje pouze úspěšné navázání spojení a přijetí dat, neboť jejich deserializaci a zpracování pokrývají unit testy `test_sync_services`.

5.2.2 Webové API

Komunikace mezi webovou a desktopovou aplikací je implementačně jednodušší než Bluetooth rozhraní, jelikož je realizována nad aplikačním protokolem (HTTP) na rozdíl od transportního (RFCOMM). Pro její testování je méně důležité ověřovat funkčnost navázání a udržení spojení, protože se jím testovaný kód nezabývá. Je naopak důležitější testovat správnou serializaci a deserializaci dat, protože webová i desktopová aplikace používají mírně odlišné reprezentace.

Testování webového API jsem navrhl tak, aby pro většinu testů nebylo potřeba spouštět proces na obou stranách. Testy jsou rozděleny do tří částí: unit test serializace a správného volání modulu `requests` na desktopu, test navázání spojení s webovým API a unit test serverové části API. Popsané testy vždy běží jako součást všech unit testů dané komponenty.

Na straně desktopové aplikace je testování realizováno nahrazením modulu `requests` a odpovědí od serveru za mock objekty. Jsou ověřovány správná URL adresa a formát odchozích dat u odesílání výsledků nebo správné zpracování uložených dat u stahování informací o závodech. Pro každý ze tří požadavků vyslaných desktopovou aplikací je napsán jeden test. Podobně je řešen i test odpovědí na požadavky webovým rozhraním. Místo mock objektů je zde využit testovací klient knihovny django-rest-framework.

Test navázání spojení s API spočívá pouze v odeslání jednoho špatně zformovaného požadavku (bez autentizace) a očekávání odpovědi 403 od serveru. Tímto nejsou kladeny žádné nároky na uložená data na webu a je možné test provést proti jakékoli běžící instanci webové aplikace.

5.3 Systémové testování

Poslední krok v testovací strategii je test celého systému jako celku. Systémové testování se zaměřuje na zkoušení jednotlivých případů užití definovaných v rámci návrhu a kontroluje tak splnění funkčních i nefunkčních požadavků na systém. V tomto případě spadají do této kategorie dva druhy vykonaných testů. Tím prvním jsou automatizované testy webové aplikace, druhým je vyzkoušení použití všech tří aplikací uživatelem pro pořádání simulovaného závodu.

5.3.1 Testy webové aplikace

Pro webovou aplikaci byly napsány automatizované testy podobně jako pro ostatní dvě části systému. Zde však již nebyly testovány jednotlivé třídy a kusy business logiky, nýbrž celé chování aplikace. Jedná se o „black box“ testing, kdy je kontrolován výstup celého testovaného programu pro nějaký uživatelský vstup bez znalosti jeho vnitřní struktury či algoritmů.

Běžící test simuluje webový prohlížeč, který odesílá na server HTTP požadavky a kontroluje, zda má vrácená odpověď správný obsah. Při psaní testů jsem se co nejvíce snažil testovat pouze logický obsah odpovědí, aby se testy nemusely přepisovat při změně v HTML struktuře. Nejefektivnější způsob se ukázalo být hledat v odpovědi hypertextové odkazy. Tímto je ověřováno správné propojení uživatelského rozhraní (tak jak je ukázáno na diagramu 4.12). Dále je testován například správný počet zobrazených akcí na stránce.

Běh těchto testů je závislý na jednom konkrétním stavu databáze. Spolu se zdrojovým kódem je distribuován soubor obsahující testovací data, které framework Django nahrává do testovací databáze před každým během.

5.3.2 Testování na simulovaném závodě

Organizování simulovaného závodu proběhlo sérií několika kroků pokrývající co největší možný počet případů užití. Bylo testováno správné použití sys-

tému i několik běžných chyb, kterých se uživatel může dopustit, jako například špatně vyplněné formuláře na webu, nezapnutý Bluetooth adaptér na počítači i mobilním telefonu nebo chybně zadané přihlašovací údaje na web. Byly provedeny následující kroky a všechny úkony s nimi spojené:

1. Registrace a přihlášení pořadatele ve webové aplikaci.
2. Založení akce a přidání závodu do této akce, definování věkových kategorií.
3. Zobrazení akce z pohledu účastníka, předběžná registrace několika účastníků.
4. Spuštění desktopové aplikace, stažení informací o závodě z webu.
5. Zápis předběžně registrovaných účastníků i několika dalších, kteří se dostavili až na místě. Tisk startovních listin.
6. Vytvoření závodu v mobilní aplikaci, spuštění časomíry, záznam startovních čísel a cílových časů.
7. Odeslání naměřených výsledků do desktopové aplikace. Tisk výsledkových listin. Odeslání výsledků závodu na web.
8. Prohlížení výsledků závodu prostřednictvím webové aplikace.

5.4 Shrnutí

Tato kapitola představila obecnou testovací strategii vyvíjeného systému a popsala jednotlivé druhy testů – jednotkové, integrační a systémové. Většina z nich je automatizovaná a ze strany vývojáře je stačí jen spustit. Testování systému jako celku probíhá manuálně.

V posledních fázích vývoje probíhaly všechny testovací běhy bez komplikací a s očekávaným výsledkem. Testování ukázalo, že si systém z funkčního hlediska poradí s organizací jednoduchého závodu a může k tomuto účelu být využit v rámci reálného nasazení. Jakékoli překážky ostrému provozu (například nedostatečné zabezpečení API a minimální grafická úprava webové aplikace, viz. kapitola 4) jsou na straně nefunkčních požadavků a jejich odstranění není implementačně náročné. Neimplementované funkční požadavky mají nízkou prioritu a mohou být postupně doplňovány, jejich absence nijak nebrání smysluplnému využití systému.

Závěr

Cílem této práce bylo navrhnout a implementovat systém pro správu běžeckých závodů, který bude určen pro pořadatele malých regionálních akcí a nabídne alternativu ke zdlouhavé ruční práci s tabulkovým editorem. V prvé řadě měl tento systém obsahovat webové rozhraní pro správu jednotlivých závodů nebo seriálů závodů a umožňovat záznam cílových časů a mezičasů na trati pomoci jednoho či více chytrých telefonů. Tyto výsledky měly být po ukončení závodu zpřístupněny na webovém rozhraní.

Z analýzy funkčních požadavků, technologických omezení a zavedených postupů vyplynulo, že kromě zmíněných dvou částí musí systém obsahovat ještě třetí část. Jedná se o desktopovou aplikaci pro správu účastníků a výsledků, která řeší problém nedostupného internetového připojení v některých lokalitách.

Implementovaný systém splňuje zadané požadavky na správu závodů, přihlašování účastníků na webu, záznam výsledků na chytrém telefonu a synchronizaci naměřených výsledků zpět s ostatními částmi systému. Je navržený modulárně, aby si mohl pořadatel vybrat, které části chce používat a které ne. Všechny jeho komponenty jsou volně dostupné a jeho využití nevyžaduje nasazení žádné drahé technologie. Nenahraditelná data, jako jsou naměřené časy, jsou neustále ukládána a tím pádem chráněna proti pádu aplikace či vypnutí zařízení, na kterém běží. Testování na simulovaném závodě neukázalo žádné závažné funkční nedostatky. Cíl vytvořit použitelný systém pro pořadatele závodů považuji za splněný.

Díky časové náročnosti implementace tří aplikací najednou nebyly realizovány některé funkční požadavky s nižší prioritou. Systém v současné době neumožňuje záznam mezičasů na trati a přenos výsledků z mobilního telefonu do počítače v reálném čase. Není také možné spravovat seriály závodů. Počítám však s tím, že vývoj systému bude pokračovat a zbývající funkční požadavky budou doplněny. Mimo tyto požadavky je největší prostor pro vylepšování a rozšiřování systému na straně webové aplikace a tímto směrem se bude další vývoj ubírat.

Literatura

[1] Olomoucký deník: Paseka: Zkuste Koblížkový běh [online]. 2014, [cit. 2015-03-04]. Dostupné z: http://olomoucky.denik.cz/kratce/1351038-paseka-zkuste-koblizkovy-beh.html

[2] TJ Liga Stovkařů Olomouc: Propozice [online]. 2015, [cit. 2015-03-04]. Dostupné z: http://www.liga100.cz/?option=com_content&view=article&id=74

[3] Behej.com: Termínovka [online]. 2015, [cit. 2015-03-03]. Dostupné z: http://www.behej.com/terminovka

[4] Prague International Marathon: Partneři [online]. 2015, [cit. 2015-03-18]. Dostupné z: http://www.runczech.com/cs/o-nas/nasi-partneri/partneri/index.shtml

[5] Atletika UNI CZ: Služby [online]. 2015, [cit. 2015-03-18]. Dostupné z: http://www.atletikauni.cz/cz/s1483/c2121-Sluzby

[6] Atletika UNI CZ: BBP 9 - Běh kolem Myslivny [online]. 2015, [cit. 2015-03-18]. Dostupné z: http://www.atletikauni.cz/cz/s1522/Vysledky/Seznam-probehlych-akci/c2132-Detail-akce/ata140-BBP-9-Beh-kolem-Myslivny

[7] Active Network: Online Race Registration Software [online]. 2015, [cit. 2015-03-21]. Dostupné z: http://www.activeendurance.com/solutions/by-feature/event-management/online-race-registration-software

[8] Letham, B.: fsTimer [software]. 2015, [cit. 2015-03-25]. Dostupné z: http://fstimer.org

[9] Dalmith Applications: Race Timer [software]. 2014, [přístup 27. března 2015]. Dostupné z: https://play.google.com/store/apps/details?id=com.dalmith.racetimer

[10] Python Software Foundation: General Python FAQ [online]. 2015, [cit. 2015-03-29]. Dostupné z: https://docs.python.org/3/faq/general.html#what-is-python

[11] Fruit, J.: Overview of Python GUI development (Hello World) [online]. 2013, [cit. 2015-03-29]. Dostupné z: http://www.pythoncentral.io/introduction-python-gui-development/

[12] Rempt, B.: *GUI programming with python: Qt edition*. Oregon: Command Prompt, 2001, ISBN 0-970-03304-4.

[13] Riverbank Computing: What is PyQt? [online]. 2015, [cit. 2015-03-29]. Dostupné z: http://www.riverbankcomputing.co.uk/software/pyqt/intro

[14] The Qt Company: Model/View Programming [online]. 2015, [cit. 2015-03-29]. Dostupné z: http://doc.qt.io/qt-4.8/model-view-programming.html

[15] SQLite Consortium: SQLite [software]. [přístup 17. dubna 2015]. Dostupné z: http://www.sqlite.org

[16] Leifer, C.: peewee [software]. 2010, [přístup 17. dubna 2015]. Dostupné z: http://peewee.readthedocs.org

[17] Gamma, E.: *Design patterns : elements of reusable object-oriented software*. Reading, Massachusetts, USA: Addison-Wesley, 1995, ISBN 0-201-63361-2.

[18] Reitz, K.: Structuring Your Project [online]. 2014, [cit. 2015-04-23]. Dostupné z: http://docs.python-guide.org/en/latest/writing/structure/#object-oriented-programming

[19] IDC: Smartphone OS Market Share, Q4 2014 [online]. 2015, [cit. 2015-04-08]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[20] Narayan, S.: Sugar ORM [online]. 2015, [cit. 2015-04-08]. Dostupné z: http://satyan.github.io/sugar/

[21] Bluetooth Special Interest Group: Bluetooth Protocol Architecture (verze 1.0). 1999, [cit. 2015-04-26]. Dostupné z: http://bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=89&vId=128

[22] Django Software Foundation: Django [software]. 2005-, [přístup 13. dubna 2015]. Dostupné z: https://www.djangoproject.com/

[23] Brown, R.: Django vs Flask vs Pyramid [online]. 2014, [cit. 2015-04-19]. Dostupné z: `https://www.airpair.com/python/posts/django-flask-pyramid`

[24] jQuery Foundation: jQuery [software]. 2006-, [přístup 14. dubna 2015]. Dostupné z: `http://jquery.com`

[25] Otto, M.: Bootstrap [software]. 2011-, [přístup 14. dubna 2015]. Dostupné z: `http://getbootstrap.com`

[26] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*. Dizertační práce, University of California, USA, Irvine, 2000.

Příloha **A**

Seznam použitých zkratek

AJAX Asynchronous Javascript And XML

API Application Programming Interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JSON Javascript Object Notation

ORM Object-Relational Mapping

OS Operační systém

REST Representational State Transfer

RFCOMM Radio Frequency Communication

SDP Service Discovery Protocol

SOAP Simple Object Access Protocol

SQL Structured Query Language

URL Uniform Resource Locator

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... Stručný popis obsahu CD
- `Dist` ..... Balíčky pro instalaci systému
  - `Desktop` ..... Spustitelná forma desktopové aplikace na OS Windows
  - `Mobile` ..... Instalační balíček mobilní aplikace
  - `Web` ..... Docker obraz webové aplikace
    - `docker` ..... Zdrojové soubory pro tvorbu Docker obrazu
- `Code` ..... Zdrojové kódy implementace
  - `Desktop` ..... Zdrojové kódy desktopové aplikace
  - `Mobile` ..... Zdrojové kódy mobilní aplikace
  - `Web` ..... Zdrojové kódy webové aplikace
- `Paper` ..... Zdrojová forma práce ve formátu LaTeX
- `assignment.pdf` ..... Zadání práce ve formátu PDF
- `thesis.pdf` ..... Text práce ve formátu PDF

Příloha C

Instalační příručka

Tato příloha uvádí konkrétní kroky umožňující spuštění systému z distribučních balíčků či zdrojového kódu. Obojí je k nalezení na přiloženém CD. Pro každou ze tří částí systému je zde uveden přesný postup instalace a spuštění pro doporučenou platformu. Pro jiné platformy či konfigurace je uveden obecný návod, jak spustit komponentu ze zdrojových kódů, a seznam dependencí.

C.1 Mobilní aplikace

Mobilní aplikace je určena pro OS Android verze 4.0 nebo vyšší.

C.1.1 Spuštění z aplikačního archivu

V adresáři `/Dist/Mobile` je umístěn soubor `app-release.apk`, což je instalační balíček obsahující zkompilovaný kód mobilní aplikace. Po přenesení do kompatibilního mobilního zařízení může být spuštěn z jakéhokoli prohlížeče souborů.

Na většině telefonů je implicitně blokována možnost instalovat ze souborů APK, které nebyly staženy z nějaké ověřené distribuční platformy, jako například z obchodu Google Play. Pro spuštění přibalené mobilní aplikace je tedy potřeba v nastaveních telefonu dočasně instalaci neověřených aplikací povolit. Jakmile je instalace povolena, samotný proces spočívá pouze ve spuštění instalačního balíčku a udělení aplikaci oprávnění přistupovat k Bluetooth adaptéru.

Po skončení instalace lze aplikaci „PasekaMobile“ spustit jako každou jinou.

C.1.2 Spuštění z IDE Android Studio

Díky nástroji Gradle, kterým je aplikace kompilována, není potřeba instalace žádných dependencí. Jakákoli verze Android Studia by si měla s mobilní aplikací poradit. V době psaní této práce je nejnovější verze Android SDK číslo 22 a proti této verzi byla aplikace vyvíjena a testována. Bez problému by však měla fungovat i na všech budoucích verzích. Spuštění aplikace i testů z IDE probíhá zcela standardním způsobem pro OS Android. Kořenový adresář projektu na přiloženém CD je `/Code/Mobile`.

C.2 Desktopová aplikace

Desktopová aplikace je teoreticky spustitelná na jakékoli platformě, pro níž existuje distribuce Pythonu 3.4. Byla testována na OS Windows 8.1 a Linux, jiné operační systémy nejsou podporovány. Konkrétně na Mac OS X aplikace nefunguje, neboť pro tento operační systém neexistuje knihovna Pybluez. Neměl jsem přístup k testovacímu prostředí s tímto OS, abych ji mohl nahradit.

C.2.1 Spuštění zkompilované verze na OS Windows

Pro operační systém Windows (testováno na verzi 8.1 s architekturou amd64) je přibalen archiv, který obsahuje spustitelný soubor a podpůrné knihovny. Tato distribuce se nachází v adresáři `/Dist/Desktop/Windows_x64/`. Běh předkompilované verze vyžaduje instalaci Microsoft Visual C++ 2010, kterým byl zkompilován Python. Instalační soubor této knihovny je umístěn ve stejném adresáři.

Není zapotřebí žádné další instalace, stačí archiv rozbalit do libovolného zapisovatelného adresáře a spustit soubor `run.exe`. První spuštění může trvat až deset vteřin. Program ve výchozí konfiguraci používá adresu `http://127.0.0.1:8000/` pro připojení k webovému rozhraní. Chceme-li se připojit ke vzdálenému serveru nebo k běžícímu Docker kontejneru na jiném operačním systému než Linux (viz. níže), adresu je možné změnit v souboru `resources/web.ini`.

C.2.2 Spuštění ze zdrojového kódu

Desktopová aplikace je napsána v Pythonu, což je interpretovaný jazyk a lze tedy spustit přímo zdrojový kód aplikace v interpreteru. Ještě předtím je však potřeba nainstalovat všechny dependence. Je to poměrně obtížná operace, proto pro uživatelské testování doporučuji použít zkompilovaný exe soubor. Na OS Linux nemá uživatel jinou možnost než se do tohoto procesu pustit, tam je však o něco jednodušší.

První dependencí je grafická knihovna Qt 5 a její Pythonovský port, PyQt 5. Pro OS Windows lze vše potřebné získat instalátorem dostupným na

`http://www.riverbankcomputing.com/software/pyqt/download5`. Pro Linux je potřeba postupovat zvlášť pro Qt a pro PyQt. Z webových stránek Qt je dostupný grafický instalátor první jmenované knihovny. PyQt lze zkompilovat ze zdrojových kódů na stejném odkaze jako Windows instalátor, na distribucích založených na Debian Linux jej lze však mnohem jednodušeji získat skrz příkaz `sudo apt-get install PyQt5-dev-tools`.

Nyní je na operačním systému Linux potřeba nainstalovat knihovny, které umožňují přístup k Bluetooth adaptéru. Na Debian Linuxu a jeho odnožích toho lze docílit příkazem `sudo apt-get install libbluetooth-dev bluez-utils`.

Ostatní dependence jsou dostupné přes příkaz `pip`, případně `pip.exe`, který je součástí standardní distribuce Pythonu. Instalují se pomocí následujících příkazů:

```
pip install peewee
pip install requests
pip install pybluez
```

S instalací poslední knihovny, pybluez, vzniká problém na 64-bitové verzi OS Windows, který se mi nepodařilo rozumným způsobem vyřešit. Nejjednodušeji ho lze obejít nainstalováním předem zkompilované knihovny z `whl` archivu dostupného na následujícím odkazu: `http://www.lfd.uci.edu/~gohlke/pythonlibs/`. Soubor se nazývá `PyBluez_0.21_cp34_none_win_amd64.whl`. Celý instalační příkaz pak bude vypadat následovně:

```
pip.exe install [whl soubor]
```

Pak je již možné spustit aplikaci příkazem `python3 run.py` z kořenového adresáře jejího kódu, kterým je na přiloženém CD adresář `/Code/Desktop`.

C.3 Webová aplikace

Webová aplikace je strukturovaná podle konvencí frameworku Django, a tak její spuštění ve vývojovém prostředí nepředstavuje příliš velký oříšek pro někoho, kdo se s tímto druhem aplikace již setkal. Pro všechny ostatní je přibalen obraz pro systém Docker, který lze spustit jedním příkazem.

C.3.1 Spuštění pomocí Docker image

Docker je open-source platforma pro virtualizaci běhových prostředí. Oproti běžné virtualizaci má množství výhod. Není například potřeba dodat úplně celý operační systém, což výrazně snižuje velikost výsledné distribuce. Pomocí souboru Dockerfile lze navíc na cílové instanci Dockeru virtualizované prostředí automaticky sestavit a nabíhání prostředí je otázkou několika málo

vteřin. Více informací o Dockeru a instalační instrukce jsou dostupné na `https://docs.docker.com/`.

Na přiloženém CD se v adresáři `/Dist/Web/` nachází Docker obraz `paseka.dist.tar`, což je instance Ubuntu Linux 14.04 s nainstalovanou a nastavenou MySQL databází, webovou aplikací a všemi dependencemi. Obsahuje také počáteční testovací data pro databázi. Spouštěcí příkazy pro Docker jsou:

```
docker load < paseka.dist.tar
docker run -d -p 8000:8000 malydani/paseka:dist
```

Takto spuštěná instance webové aplikace bude dostupná z hostitelského systému na `http://DOCKER_IP:8000`. Na OS Linux je `DOCKER_IP` rovna `127.0.0.1`, na jiných systémech je potřeba ji zjistit pomocí příkazu `boot2docker ip`. Více informací o mapování IP adres a portů je dostupných opět na stránkách Dockeru. Kontejner vystavuje pro příchozí spojení vždy port 8000.

Sestavení obrazu pomocí Dockerfile Přiložené CD obsahuje i zdrojový soubor Dockerfile, pomocí něhož lze distribuční obraz sestavit přímo na cílovém systému. V adresáři `/Dist/Web/docker` stačí pouze spustit následující příkaz a obraz se sám nainstaluje. Spustit je ho pak možné druhým z dvojice příkazů uvedených výše.

```
docker build -t malydani/paseka:dist .
```

Poznámka Docker obraz na přiloženém CD není v žádném případě určen pro produkční použití, neboť není řádně zabezpečen, není nakonfigurován pro posílání e-mailů a obsahuje pouze vývojový Django server. Sestavení produkčního prostředí by však pomocí Dockeru nebylo obtížné.

C.3.2 Spuštění ve vývojovém prostředí

Kořenový adresář pro spouštění všech následujících příkazů je `/Code/Web/`, kde se nachází všechny zdrojové soubory webové aplikace.

Pro spuštění aplikace z těchto zdrojových kódů je opět potřeba nainstalovat dependence, tentokrát se to zvládne jedním příkazem:

```
pip install -r requirements.txt
```

Dále vyžaduje aplikace s výchozím nastavením běžící MySQL databázi na adrese `localhost:3306`, která se jmenuje `BP_dev` a ke které má plná práva uživatel `paseka_dev` s heslem `passdev2015`. První nastavení této databáze se musí provést následujícími příkazy:

```
python3 manage.py makemigrations
python3 manage.py migrate
python3 manage.py loaddata clean.json
```

Spustit vývojový server pak lze příkazem `python3 manage.py runserver`.