České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství

Tvorba aplikace pro správu registrací a výsledků na závodech

Development of Races Management Application

BAKALÁŘSKÁ PRÁCE

Vypracoval: Vojtěch Zahradník
Vedoucí práce: RNDr. Petr Kubera, Ph.D.
Rok: 2021

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství Akademický rok 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Vojtěch Zahradník**
Studijní program:	**Aplikace přírodních věd**
Obor:	**Aplikace softwarového inženýrství**
Název práce česky:	**Tvorba aplikace pro správu registrací a výsledků na závodech**
Název práce anglicky:	**Development of Races Management Application**

Pokyny pro vypracování:

1. Proveďte rešerši aplikací pro správu registrací a výsledků na závodech.
2. Seznamte se s požadavky na aplikaci od pořadatelů závodů.
3. Navrhněte přehledné a spolehlivé GUI.
4. Implementujte aplikaci
5. Ověřte funkčnost aplikace při reálném nasazení.

Doporučená literatura:

[1] HENLEY, A.J a D. WOLF. *Learn Data Analysis with Python.* New York: Apress Media, 2018. ISBN 978-1484234853.

[2] SUMMERFIELD, M. *Python Výukový kurz 3.* Brno: Computer Press Brno, 2013. ISBN 978-80-251-2737-7.

[3] PILGRIM, M. *Ponořme se do Python(u) 3.* 3. vydání. Praha: CZ.NIC, 2010. ISBN 978-80-904248-2-1.

Jméno a pracoviště vedoucího práce:

RNDr. Petr Kubera, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

.....
vedoucí práce

Datum zadání bakalářské práce: 16. 10. 2020

Termín odevzdání bakalářské práce: 7. 7. 2021

Doba platnosti zadání je dva roky od data zadání.

.....
garant oboru

.....
vedoucí katedry

.....
děkan

V Praze dne 16. 10. 2020

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Děčíně dne .....

.....
Vojtěch Zahradník

Poděkování

Děkuji RNDr. Petru Kuberovi, Ph.D. za vedení mé bakalářské práce, za skvělou komunikaci při vzniklém problému a za duchaplné návrhy, které práci obohatily.

Vojtěch Zahradník

Název práce:
Tvorba aplikace pro správu registrací a výsledků na závodech

Autor: Vojtěch Zahradník

Studijní program: Aplikace přírodních věd
Obor: Aplikace softwarového inženýrství
Druh práce: Bakalářská práce

Vedoucí práce: RNDr. Petr Kubera, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant: –

Abstrakt: Cílem této bakalářské práce bylo vytvořit univerzální aplikaci pro správu primárně cyklistických a běžeckých závodů. Nejdříve jsem provedl rešerši o aktuálním stavu aplikací na trhu, pro danou problematiku a také zjistil požadavky na aplikaci od organizátorů závodů. Poté jsem na základně těchto informací navrhl databázi a začal vypracovávat funkční aplikaci s přívětivým GUI. K realizaci byl využit hlavně programovací jazyk Python a jazyk SQLite pro realizaci na straně databáze.
Klíčová slova: správa závodů, PyQt5, Python, SQLite, výsledky

Title:
Development of Races Management Application

Author: Vojtěch Zahradník

Abstract: The aim of this bachelor thesis was to create a universal application for the management of primary cycling and running races. First, I demonstrated a search on the current state of applications on the market for a given issue and also found out the requirements for the application from the race organizers. Then, based on this information of the proposed databases, I started preparing functional applications with a friendly graphical user interface. The Python programming language was mainly used for the implementation. Furthermore, for the database programming language SQLite.
Key words: races management, PyQt5, Python, SQLite, results

Obsah

Úvod

V dnešní době, kdy více lidí začíná sportovat, jsou sportovní akce nedílnou součástí sportu jako takového. Do sportovních akcí se samozřejmě řadí hlavně závody. Dnešní závody už nejsou převážně pouze pro zatvrzelé závodníky, ale také pro lidi, kteří daný sport mají pouze jako koníček. Není tedy překvapením, že takovýchto akcí přibývá, ať už za účelem vytvoření skvělé atmosféry nebo zisku.

Na vzestupu je hlavně cyklistika a běh. Tyto dva sporty se stávají velmi oblíbeným prostředkem pro trávení volného času, odpočinku a nebo setkávání s přáteli. Díky vyšší pořizovací ceně kola se dost často člověk rozhodne spíše pro běh než cyklistiku. Kvůli přirozené lidské vlastnosti, tedy soutěživosti, se stávají závody velmi vyhledávanou akcí převážně o víkendech. Na jediný závod se dokáže sjet od sta až po desetitisíce závodníků.

Při přibývajícím počtu účastníků, ale samozřejmě přestávají fungovat běžné metody, ať už registrace a nebo správy výsledků. Například těžko vyhodnotíte na papíře výsledky tisíce závodníků. Závodníci se tedy čím dal častěji setkávají se situacemi, kdy nejsou výsledky k dispozici včas a nebo se start závodu musí odsunout kvůli velkým frontám u registrací. Celkově vzato se setkávají s nedostatky ze stran organizátorů. V mnoha případech však za to organizátoři nemohou. Ve zkratce nemají dostatek prostředků k zajištění všech těchto náležitostí, ať už zajištění více pomocníků nebo lepší systém správy.

Většina aplikací na trhu, zabývající se touto problematikou, je totiž velice drahých a nebo organizátor potřebuje čipy na změření času, které jsou také velmi drahé. Velice drahých myslím řády desítek až stovek tisíc korun. Což je pro menšího organizátora, dělající sportovní akce pro své přátele, nepředstavitelná částka. Dost často si tedy organizátor pokusí samotnou aplikaci, pro správu závodu, opatřit sám. Tyto aplikace jsou ale zastaralé, pomalé a samozřejmě nějaké pozdější vylepšení nejsou možná.

Téma mé bakalářské práce jsem si tedy vybral, kvůli zlepšení organizace a pozdější správy závodů pro menší organizátory v mém okolí. Také můj zájem o programování a o programovací jazyk *Python* [19] narostl do takové výše, že jsem si chtěl zkusit vývoj nějaké robustnější aplikace s reálným využitím. Dalším důvodem bylo to, že se sám podílím na organizaci převážně běžeckých závodů a tak vidím všechna úskalí, co může toto odvětví přinést.

Základní dva důvody, proč využít moji aplikaci proti jiným na trhu, jsou jednoduché. Za prvé bude využívat nových technologií a tím bude stabilní i při vysokém zatížení. Na využití bude stačit i starší notebook s průměrným výkonem. Dalším důvodem je cena, která oproti jiným aplikacím je zanedbatelná. Samozřejmě si uvědomuji, že nebudu moct nabídnout profesionální aplikaci a systém, s kterým se můžeme setkat například

u zahraničních firem. Avšak tyto aplikace se spíše hodí pro organizátory závodů národní úrovně s velkou účastí, nikoli menších závodů.

Mým hlavním cílem je vytvořit samostatně fungující systém, který bude sloužit pro účely tvorby a následné správy závodů. Samotný systém bude koncipován převážně na využití na místě konání akce, jelikož většina funkcí bude využitelná až v den akce. Organizátorovi by aplikace měla nabídnout přehledné a rychlé *GUI*[1] s minimální potřebou znalostí práce na počítači. Zároveň aplikace musí být funkční bez internetu, protože sportovní akce jsou dost často na špatně přístupných místech bez signálu. Nemohu tedy počítat s připojením k internetu, proto jsem také zvolil formu desktopové aplikace místo webové aplikace.

V práci nalezneme několik kapitol. V první kapitole nalezneme rešerši aplikací na trhu zabývající se tímto problémem, která shrne jaké možnosti samotný organizátor má při organizování nějaké akce. V druhé kapitole ve zkratce popíši použité technologie, jako například *PyQt5* [27] nebo *SQLite* [1]. Třetí kapitola popisuje návrh aplikace z pohledu databáze a *GUI*. Ve čtvrté kapitole nalezneme přiblížení implementace samotné aplikace. Práce také bude obsahovat uživatelskou příručku, kterou nalezneme v přílohách, pro nové uživatele aplikace, která by měla být srozumitelná i pro úplné začátečníky s počítačem. Finální stav aplikace otestuji na několika závodech. Testování by mělo proběhnout na začátku jara roku 2021. Samotné testování bude popsáno v příloze.

[1] Grafické uživatelské rozhraní

Kapitola 1

Rešerše

V této části mé práce popisuji elementární požadavky kladené na aplikaci. Jsou zde nadefinovány základní funkce, které aplikace musí obsahovat. Tato část obsahuje valnou většinu informací od organizátorů závodů v mém okolí. Na základě těchto informací v dalších částech mé práce vybírám správnou technologii a sestavuji samotnou aplikaci. Dále popisuji možná aktuální řešení na trhu a zmiňuji jejich nedostatky. Jako aplikace řešící mé zadání jsem vybral aplikace *Bikebase*, *RaceResult* a webovou aplikaci *naZavody.cz*. Tyto aplikace jsem zkoušel pouze na mém počítači a nikoli přímo na závodě. Jejich nedostatky jsem vypozoroval v pouhém testování a nebo mi byly sděleny od organizátorů.

1.1 Požadavky na aplikaci

Při prvotní rešerši jsem byl obeznámen s několika nutnými funkcemi, které aplikace musí obsahovat. Každou funkci jsem jednoduše popsal a uvedl její budoucí fungování v mé aplikaci v jednotlivé podkapitole.

1.1.1 Tvorba závodu

Tato funkce bude umožňovat vytvořit si vlastní závod. K vytvoření bude potřeba několik základních informací, jako například jméno nebo nepovinný údaj typ závodu. Typem myšleno například běh a nebo silniční cyklistika. Další potřebnou informací také bude datum konání dané akce a zda závod patří do nějakého seriálu. Informací navíc bude ročník závodu a nebo maximální počet závodníků. Tyto informace budou sloužit pouze pro organizátora. Dále si sám organizátor může napsat poznámky k závodu a v pozdější fázi je smazat.

1.1.2 Tvorba seriálu

Aplikace musí nabízet vytvoření seriálu a pozdější přiřazení závodu do seriálu. Pojmem seriál je rozuměno více závodů, které budou mít společné vyhodnocení. Obvykle se tomu

tak děje na konci závodní sezóny. Bude nutné tedy generovat i výsledky různých seriálů a vytvořit tak systém bodování.

1.1.3 Přidání kategorie

Po tvorbě závodu je nutné přidat kategorie. S povinnou informací název kategorie je také nutné zadat rozmezí ročníků kategorie. Pokud uživatel nechce zadat rozmezí kategorie musí potvrdit, že se jedná o kategorii bez ročníků, tedy o kategorii, která bude přidělována závodníkům ručně. Povinným prvkem je zadání pohlaví. Pokud pohlaví nebude zadáno jedná se o takzvanou unisex kategorii a závodníci v ní budou moci být muži i ženy. Dále si organizátor může připsat informace ke každé kategorii, jako je maximální počet závodníků, počet okruhů na trati a nebo startovné. Aplikace potom sama bude počítat organizátorovi vybrané peníze podle přihlášených závodníků v kategorii.

1.1.4 Archiv

Aplikace musí obsahovat zařazení závodů do archivu. Pro lepší manipulaci s aktuálními závody je potřeba mít možnost staré závody přemístit do archivu. U archivu by mělo být možné zobrazit startovní listinu a výsledky. Zároveň závod z archivu by měl být schopen se znovu obnovit či smazat.

1.1.5 Import

Import bude jedna z nejdůležitějších funkcí. Bude se jednat o možnost vložení dat z externí aplikace. Jelikož si plně uvědomuji problém rozdílného formátu, bude potřeba vytvořit konvertor, který tento problém vyřeší. Po správném vložení dat se data nahrají do databáze a budou se nadále chovat jako by byla vytvořena v mé aplikaci. Konvertor musí také zvládnout import neznámých sloupců s informacemi, jako například datum platby a datum registrace čistě pro pohodlí organizátora. Tyto neznámé sloupce však v aplikaci nebudou editovatelné.

1.1.6 Prezentace na místě

Prezentace neboli registrace na místě je společně s časomírou a výsledky jedna ze základních pilířů celé aplikace. Také se jedná o hlavní vstup dat do aplikace. U prezentace bude nutné se popasovat s automatickým počítáním startovních čísel a nebo také s automatickým našeptávačem podle příjmení závodníka. U závodníka bude nutné zadat jméno, příjmení, rok narození a pohlaví. Dobrovolné informace potom budou e-mail, tým a nebo telefonní číslo. Kategorie se buď sama určí podle roku narození a nebo ji sám uživatel určí z bez ročníkových kategorií. Zadané informace se ihned odešlou do databáze, kvůli možnému selhání aplikace.

1.1.7 Časomíra

Sekce časomíry bude sloužit pro vložení času, který měl závodník v cíli. Tato sekce bude mít dva základní způsoby vložení času. Buď uživatel přiřadí čas zadanému startovnímu číslu a nebo vloží čas přímo k závodníkovi v tabulce. Při každém vkládání času je nutné ještě zadat stav dojezdu nebo doběhu, pokud se jedná o běžecký závod. Myšleno tím tedy jestli závodník zdolal všechna nutná kola nebo například byl diskvalifikován. Je to nutné zadat kvůli následné generaci výsledků.

1.1.8 Výsledky

Výsledky budou finálním výstupem celé aplikace. Je to to kvůli čemu každý závodník přijede na závod. Část s výsledky musí tedy fungovat rychle a stabilně. Musí zobrazovat celkové výsledky a zároveň výsledky pro každou kategorii. Všechna data musí být schopna exportu do *PDF, XLSX* a nebo rovnou tisku. U každého závodníka se rovnou bude počítat pořadí a odstup od předešlého závodníka. Pokud závod bude součástí seriálu, tak k jednotlivému závodníkovi se vypíše i počet získaných bodů.

1.1.9 Editace

Všechna vložená data bude možno editovat, kvůli pozdějším úpravám. Editace by měla být dostatečně rychlá i při vyšším počtu dat. Editace by také měla mít různé „vychytávky“, jako po přidání kategorie přepočítat závodníky a pokud vyhovují přidané kategorii tak je do ní přidělit.

1.1.10 Export dat

Pro potřebu zobrazení startovní listiny nebo výsledků přímo na místě závodu např. na tabuli je nutno implementovat možnost tisku těchto dvou celků. Je také zapotřebí vytvořit možnost výběru exportovaných kategorií.

1.2 Řešení dostupné na trhu

Podle požadavků na aplikaci se může zdát, že vhodných řešení na trhu bude dostatek. Avšak tomu tak není. Většinu aplikací si nechají udělat časoměřičské firmy na zakázku a dále jí nesdílí. Avšak služba těchto firem je dost často velmi drahá a pro menší závody i zbytečná.

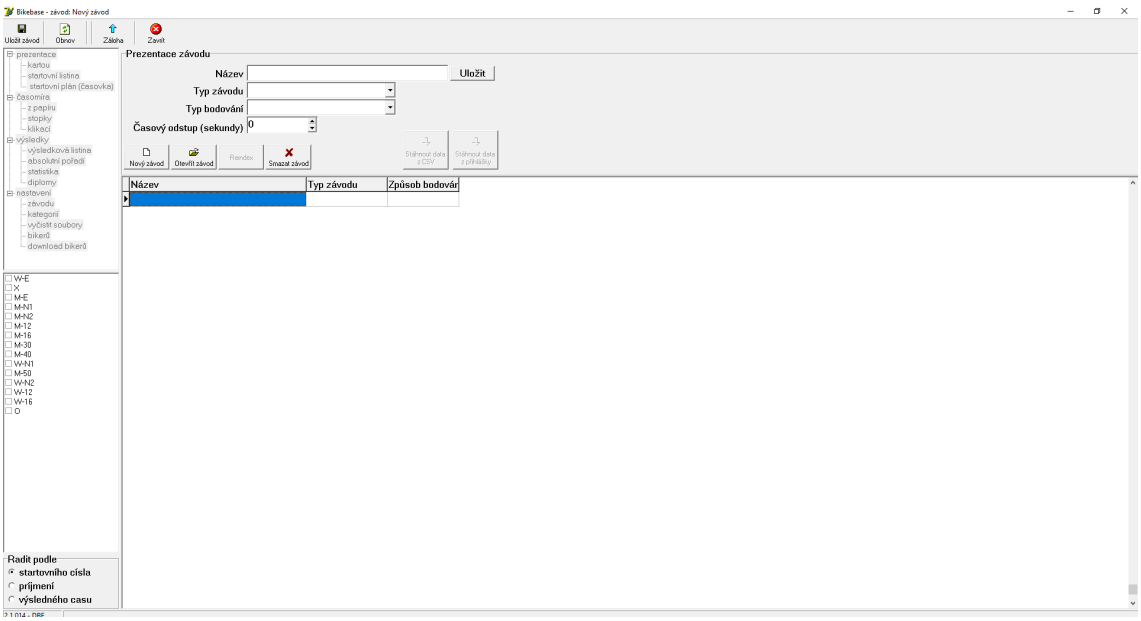
Přímo od organizátorů jsem dostal několik tipů na používané aplikace pro správu závodů. Nejvíce dominovala aplikace *Bikebase* a *RaceResult*. První z aplikací je vytvořena před více jak deseti lety a poukazuje na zastaralost aplikací na trhu. Druhou aplikací je *RaceResult*, která je vyvíjena zahraniční firmou tedy kvalita je velmi vysoká, ale pořizovací cena této aplikace a věcí spojených s aplikací je velmi odrazující. Poslední aplikací je

webová aplikace *naZavody.cz*. V této kapitole tyto aplikace v krátkosti popíši a rozeberu jejich nedostatky. Ať už se jedná o funkční stránku a nebo ekonomickou.

1.2.1 Bikebase

Aplikaci *Bikebase* si nechali udělat organizátoři, jednoho z největšího seriálů u nás, *Pekla Severu*. Slouží tedy pouze pro organizaci tohoto seriálu. Organizátoři ji už využívají přes 15 let a aplikace proto vypadá velmi staře (viz obr. 1.1). Vzhledově se jedná o velmi nepřehledné *GUI* a hlavně velmi pomalé. Prakticky načtení každé funkce pro správu trvá několik sekund a uložení ještě déle.

Naopak výhoda spočívá v samostatnosti a jednoduchosti aplikace. K aplikaci není nutné mít zapůjčené nebo dokonce koupené čipy na měření časomíry. Časy závodníků v cíli se jednoduše dají vepsat do aplikace, jako tomu bude i u mé aplikace. Organizátoři tedy mohou skloubit levnou variantu měření času pomocí stopek a zpracovat výsledky ve své aplikaci, což jim velmi ulehčí práci.



Obrázek 1.1: Aplikace Bikebase

1.2.2 RaceResult

Tato aplikace je od profesionální firmy ze zahraničí. Tato firma nabízí nejen koupi nebo zapůjčení aplikace, ale i půjčení nebo koupi příslušenství k ní, které je k fungování aplikace zapotřebí. Nejde tam tedy samostatně vložit časy závodníků v cíli. K aplikaci je tedy nutné vlastnit nebo mít půjčené čipy a cílové brány, které zaznamenají proběhnutí čipu. Firma dále nabízí i tisk diplomů a startovních čísel.

Co se týče dalších služeb firmy, tak firma poskytuje plné zaopatření závodu. Od tisku diplomů, startovních čísel a potisků na trička po vyhodnocení výsledků. Samozřejmě cena s těmito službami narůstá. Firma také nabízí časoměřičské balíčky přímo na určitý druh závodu. Samotnou firmu můžeme znát z velkých světových závodů, jako je například Celtic man triatlon.

Výhod je samozřejmě mnoho. S tímto systémem prakticky nemůže nastat chyba při zpracovávání výsledků, ale s velkou kvalitou se nese i velká cena. Cena na závodníka se sice snižuje s narůstajícím počtem (viz tabulka 1.1), avšak i tak je velmi vysoká. Cena aplikace je vyobrazena v následující tabulce. Do celkové ceny však nezahrnuji pronájem dalších nutných komponent k fungování aplikace.

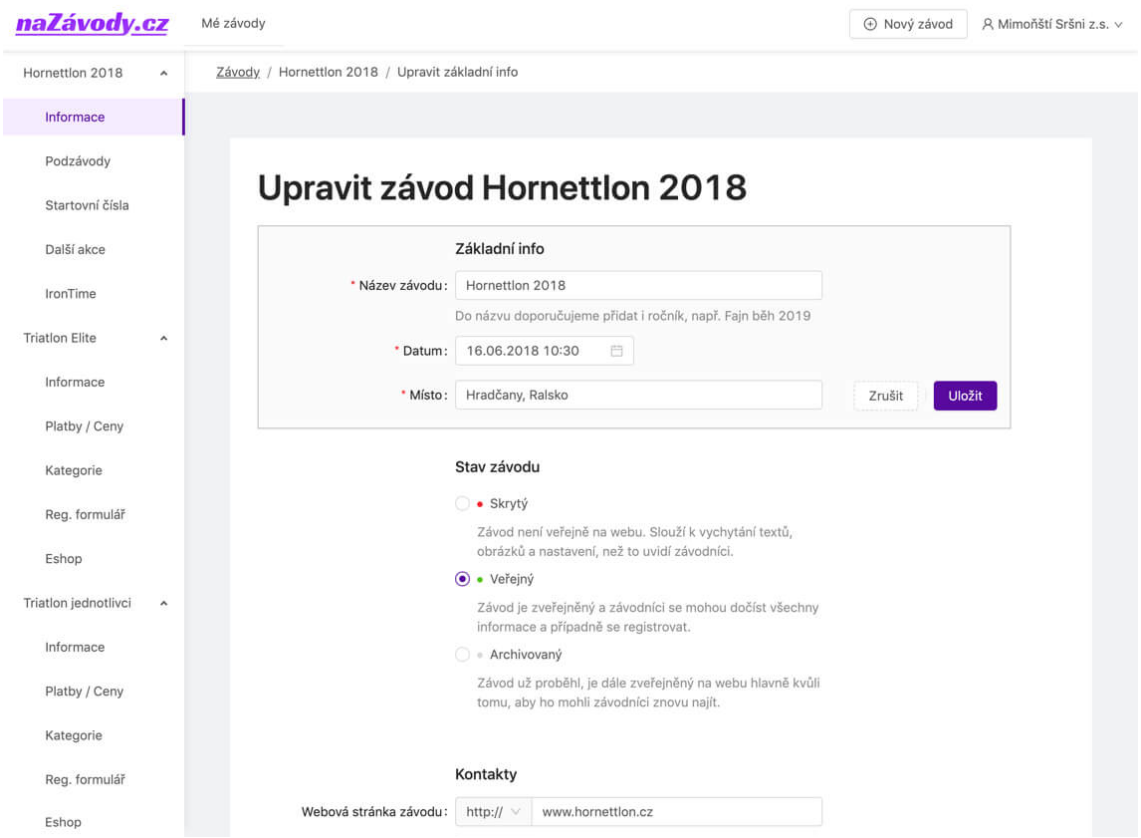
Závodníci	**Cena[EUR]**
100	25
250	50
500	85
1000	130
5000	550

Tabulka 1.1: Cena v závislosti na počtu závodníků

1.2.3 naZavody.cz

Jako poslední srovnání bych uvedl webovou aplikaci *naZavody.cz*. Tato aplikace je nabízena všem organizátorům v cenové hladině od 5 Kč na závodníka po maximální cenu 2495 Kč za závod. Aplikace splňuje většinu požadavků, které jsem uvedl v předešlé kapitole a zároveň nabízí i příjemné a přehledné grafické rozhraní, jak můžeme vidět na obrázku 1.2. Jako bonus nabízí propojení plateb přímo s aplikací a nebo přímo tisk startovních čísel.

Její nevýhodou je to, že je webová. Organizátor tedy musí mít zajištěno stabilní připojení k internetu. A to, že celý závod musí být vytvořen čistě v aplikaci. Tedy možnost online registrace na stránkách závodu a následný import do aplikace není k dispozici. Další velkou nevýhodou je časomíra. Ta je řešena jako u aplikace *RaceResult* přes čipy. V tuto chvíli se musíme obracet na externí firmu, co nám časomíru zajistí.



Obrázek 1.2: Okno webové aplikace naZávody.cz

V závěru kapitoly jsem srovnal klíčové vlastnosti těchto aplikací a vnesl je do tabulky, kterou můžeme nalézt na obrázku číslo 1.3. Můžeme tak vidět, že pro naše požadavky si nejlépe vede aplikace *BikeBase*.

	BikeBase	RaceResult	naZavody.cz
Ovladatelnost	Jednoduchá	Složitá	Jednoduchá
Vhodnost na malé závody	ANO	NE	ANO
Podpora	NE	ANO	ANO
Cena	Žádná	Vysoká	Nízká
Webová	NE	NE	ANO
Nutné čipy	NE	ANO	ANO

Obrázek 1.3: Finální srovnání aplikací

Kapitola 2

Použité technologie

Kvůli požadavku vytvořit desktop aplikaci se mi seznam možných programovacích jazyků zmenšil. Daným kritériím nejvíce vyhovoval programovací jazyk *Python*. Kvůli jeho možnosti lehce manipulovat s daty a hlavně rychlé a flexibilní tvorbě *GUI*. Aplikace naprogramovaná například v *Javě* [13] nebo v *C++* [5] by nejspíš byla rychlejší, avšak vývoj by byl pro mě složitější a možnost se zlepšit v *Pythonu* byla pro mě samotného více lákavá.

2.1 PyQt

PyQt [27] je multiplatformní *GUI toolkit*[1] pro programovací jazyk *Python*. Je to jedna z možností, jak v *Python*u vytvářet *GUI*. Alternativy jsou například *PySide*, *PyGTK*, *wxPython* nebo *Tkinter*, který je dodáván přímo s jazykem *Python*. *PyQt* je stejně jako *Qt* svobodný software. Pomocí této technologie je v mé aplikaci tvořeno celé *GUI*.

Já osobně v mé aplikaci využívám *PyQt* verze 5.12, která běží na *Pythonu* verze 3 a výše. Začátkem roku 2021 vyšla nová verze tohoto *toolkitu* verze 6. Místo *PyQt* jsem si mohl vybrat *PySide* verze 2[2], tyto dvě technologie se od sebe prakticky neliší, ale kvůli bližšímu vztahu k *Qt* softwaru jsem vybral *PyQt*. Další možností návrhu *GUI* byl *Tkinter*. Tento *toolkit* ale nemá žádný software pro návrh, jako tomu je u *PyQt* a zároveň obsahuje méně widgetů.

2.2 SQLite

SQLite [1] je relační databázový systém šířen pod licencí public domain. Není založen na principu *klient-server*, tedy po připojení k aplikaci je k dispozici pomocí rozhraní. Každá databáze v *SQLite* je samostatný soubor typu *.dbm*[3].

[1] Sbírka nástrojů pro tvorbu GUI
[2] Hlavním rozdílem je licencování
[3] Database Manager

Ve své práci jsem se pro *SQLite* rozhodl kvůli nutnosti nezávislosti aplikace na externím serveru. Dalším důvodem je jednoduchost migrace databáze. Uživatelovi stačí přesunout jeden soubor.

2.3 HTML

Hypertext Markup Language [12] je značkovací jazyk používaný pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy. Samotné *HTML* umožňuje tvorbu statických stránek, pro dynamické webové stránky je nutné využít jiných technologií. V mé aplikaci jsem využil *HTML* pouze pro export a tisk startovní listiny nebo výsledků. Aplikace pouze vygeneruje *HTML* syntaxi tabulek s exportovanými daty a exportuje jej.

2.4 QSS

Qt Style Sheets neboli *QSS* [11] je jazyk založený na *CSS* syntaxi pro *HTML*. *QSS* se místo pro webové stránky používá pro stylizaci komponent rozhraní neboli widgetů. Stejného výsledku dosáhneme i pomocí třídy *QStyle*. Šablony stylů jsou textové specifikace, které mohou být určeny pro celou aplikaci. V této aplikaci bylo *QSS* použito pro stylizaci každého komponentu *GUI*.

2.5 GoogleAPI

V této aplikaci se také využívá *Google API* [7] pro synchronizaci databáze s uživatelovým Google diskem a pro hromadné posílání e-mailů. Tyto funkce sice nebyly zmiňovány v požadavcích na aplikaci, ale mně přišly velice užitečné, ať už pro přesun databáze z jednoho zařízení na druhé nebo k informování všech závodníků skrze e-mail. Proto jsem ji implementoval do mé aplikace.

Google API je rozhraní pro programování aplikací vyvinuto společností Google. Samotné *API* umožňuje komunikaci mezi aplikací a službami Google, jako je například Gmail, Google disk a Google mapy. V mé práci byly využity *Gmail API* [6] a *Drive API* [9].

2.5.1 Gmail API

Gmail API je takzvaně *REST API* [22] používané pro přístup k uživatelským Gmail schránkám. Pro většinu webových aplikací je *Gmail API* nejlepší možností, jak spravovat uživatelům e-mail. Nabízí několik funkcí, jako například čtení, odesílání nebo filtraci zpráv. Já osobně jsem tuto technologii použil pro automatické odesílání e-mailů všem závodníkům. Uživatel však nemůže vybrat komu se daný e-mail odešle.

2.5.2 Drive API

Drive API funguje na stejné technologii jako *Gmail API*. Akorát *Drive API* umožňuje vývojáři vytvořit aplikaci, která bude spravovat Google disk. Nabízí programátorovi možnost nahrávat nebo i stahovat dokumenty z uživatelova úložiště. *Drive API* bylo využito pro připojení aplikace k uživatelovu Google účtu a následně k disku pro upload nebo download databáze. Funkce byla implementována pro rychlou migraci databáze.

2.6 JetBrains

JetBrains s.r.o. [14] je firma vyvíjející IDE[4] pro programátory. Od roku 2000 začala nabízet IDE pro programovací jazyk *Java* a v dnešní době nabízí 10 vývojových prostředí pro různé programovací jazyky.

V mé práci jsem využil prostředí, od firmy JetBrains, pro správu databáze DataGrip a pro vývoj v programovacím jazyku *Python* jsem využil PyCharm.

2.7 Adobe XD

Adobe XD (Experience Design) [2] je vektorový editor pro UI[5] designery na návrh aplikací. V aplikaci můžeme převést návrhy z návrhové plochy na prototypy velmi rychle. Já využil tento program pro návrh takzvaně *wireframe*[6] *modelu* pro moji aplikaci.

2.8 QT Designer

Qt Designer [21] je nástroj pro tvorbu grafického rozhraní s *Qwidgety*[7]. Celý program je založen na principu *WISIWYG*[8]. Pomocí takzvaných slotů a signálů v aplikaci můžeme jednoduše napojit tlačítka na jednotlivé metody ještě než se dostaneme do programovací části. Každý widget, který je tvořen v Qt Designeru je možné v pozdější části změnit a nebo úplně zrušit. Qt Designer vytváří soubory s koncovkou *.ui*. Na soubor typu .py, neboli soubor napsaný v jazyce *Python*, se dostaneme pomocí nástroje *Pyuic5* [20]. Nástroj přetvoří soubor z Qt Designer na soubor v programovacím jazyce *Python* pomocí jednoduchého příkazu v terminálu (viz ukázka 2.1).

```
1 pyuic5 -x fileName.ui -o fileName.py
```

Ukázka 2.1: Ukázka převedení formátů

[4]Vývojové prostředí
[5]Uživatelské rozhraní
[6]Drátový model
[7]Základní třída pro každou třídu uživatelského rozhraní
[8]Co vidíš, to dostaneš

2.9 MySQL Workbench

MySQL Workbench [15] je vizuální nástroj pro návrh struktury databáze. Nabízí jednoduché modelování dat pro vývoj v jazyce *SQL* [23]. V programu je také možnost ke konfiguraci *SQL* serveru a správu uživatelů.

V této aplikaci byl MySQL Workbench využit pro prvotní návrh databáze. Vizuální struktura byla dále převedena na formát *SQLite*.

2.10 Visual Paradigm

Visual Paradigm [26] je nástroj pro tvorbu *UML* [25] verze 2. Umožňuje také generování kódů a nebo reverzní generaci *UML* z vašeho kódu.

Aplikace byla využita pro vytvoření diagramu tříd a diagramu užití.

Kapitola 3

Návrh aplikace

V předchozí kapitole jsem podrobně popsal požadované funkce, které má aplikace musí obsahovat. Dále jsem popsal výhody a nevýhody aktuálních systémů na trhu, které splňují dané požadavky. V této kapitole podrobně popíši návrh databáze, strukturu aplikace a návrh grafického uživatelské rozhraní.

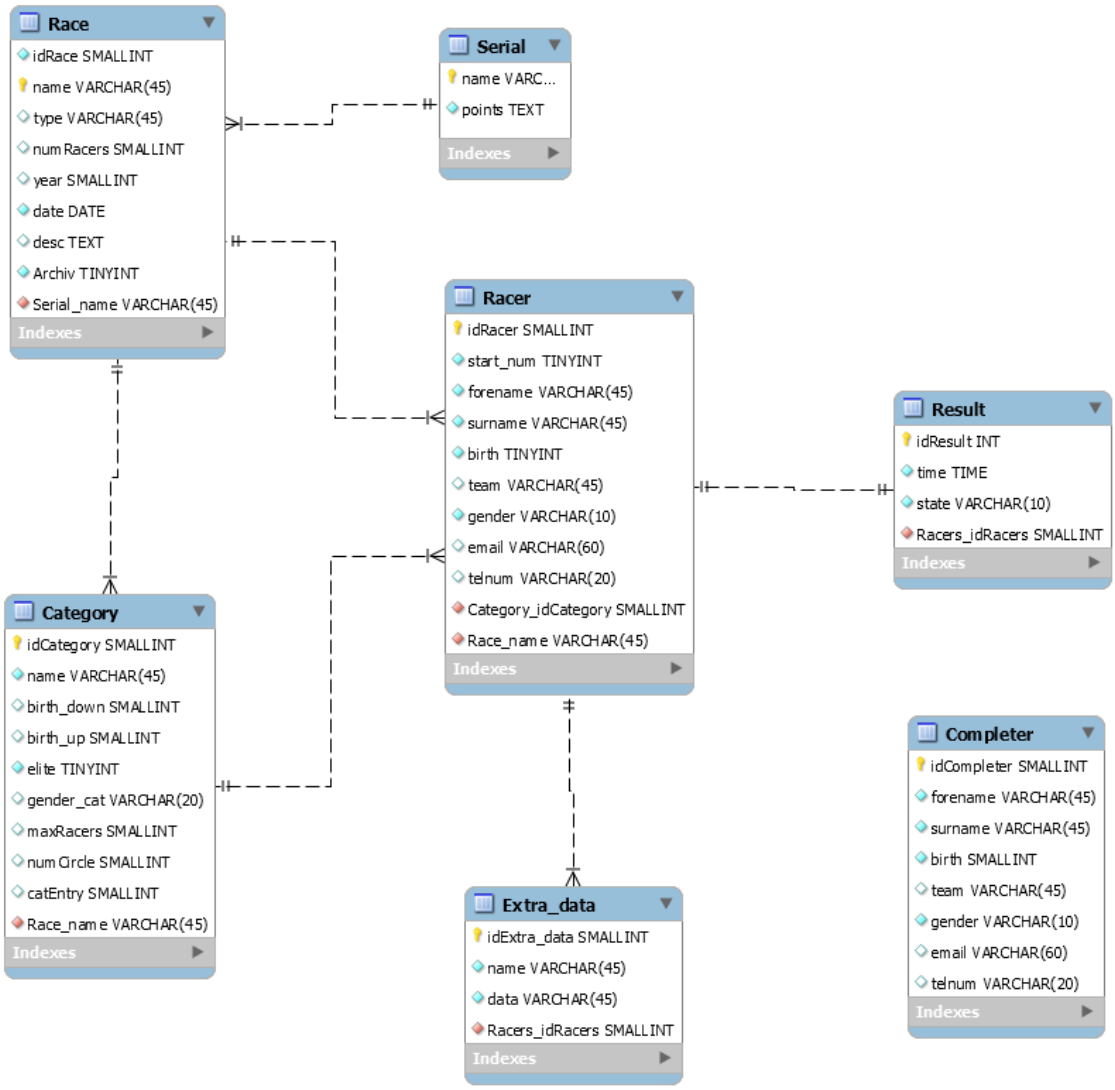
3.1 Databáze

Jedna z nejdůležitějších částí aplikace, která zaznamenává data, je správný návrh datové struktury. Jelikož koncový uživatel bude pracovat s velkými objemy dat najednou, byla pro uložení dat zvolena databáze. Ta poskytuje pozdější velmi jednoduchý a strukturovaný přístup k uchovaným datům. Dále nabízí velmi rychlé třídění a filtraci dat podle daných kritériích.

Při ohledu na mé vlastní zkušenosti a při braní v potaz požadavky na aplikaci, popsané v předešlé kapitole, jsem dospěl k následujícímu návrhu databáze. Prvotní návrh probíhal v programu MySQL WorkBench a následně byl převeden do použitelné formy formátu *SQLite* v aplikaci DataGrip od Jetbrains. V následujícím textu stručně popíši každou entitu databáze a vztahy mezi nimi.

3.1.1 Konceptuální model databáze

Konceptuální model [3, 17] databáze se ve světe informatiky používá pro abstraktní a konceptuální znázornění dat. Jedná se o metodu datového modelování, která tvoří schémata. V tomto případě schéma typu relační databáze. Tento model umožňuje zobrazit a popsat jednotlivé objekty v databázi a vztahy mezi nimi z hlediska jejich významu. Výsledkem modelování je potom nezávislé schéma, které lze aplikovat v souvislosti s jinou aplikací. Nejčastější znázornění bývá v podobě diagramu (viz obr. 3.1).



Obrázek 3.1: Konceptuální model databáze

Racer

Základní entitou celého modelu je entita *Racer*, která představuje všechny zaznamenané závodníky v databázi. Má několik povinných atributů a tím jsou *idRacer, start_num, forename, surname, birth* a *gender*. Tyto atributy specifikují identifikační číslo závodníka, startovní číslo, jméno, příjmení, rok narození a pohlaví. Atribut *idRacer* je zároveň i primárním klíčem celé entity. Nepovinnými atributy jsou potom *team, e-mail*, *tel_num*, které prezentují tým, email a telefonní číslo závodníka. Tato entita má dva cizí klíče a tím jsou *Category_idCategory*, který představuje identifikační číslo kategorie, ve které se závodník nachází, a *Race_name*, což je primární klíč z tabulky *Race*, který určuje v kterém závodu je závodník přihlášen. Oba cizí klíče mají nastavené provázání při updatu dat na *cascade*, tedy změna dat cizího klíče se prolne i do této tabulky. Při akci *delete* se identifikační číslo kategorie nastaví na null, kdežto pokud se smaže závod, v kterém je závodník přiřazen, smaže se i závodník, ale pouze v daném závodě. Vztahy s jinými entitami jsou následující:

- Právě jeden závodník může mít jeden výsledek.
- Více závodníků může být přiřazeno do jednoho závodu.
- Jeden závodník může mít více extra dat.
- Více závodníků může být přiřazeno do jedné kategorie.

Race

Druhou hlavní entitou je *Race*, která slouží pro uchování dat spojené se závody, s primárním klíčem *name* a dalšími povinnými atributy, jako *idRace* a *date*. Zmíněné atributy slouží pro uchování názvu závodu, *idRace* a datum konání. Nepovinných atributů u závodu je více než u závodníka. Jde o atributy *type, num_Racers, year, desc, archiv*, které znamenají typ závodu, maximální počet účastníků, ročník, poznámky a zda je závod v archivu. Tato entita slouží k uchování jednotlivých závodů a informací o něm. Jako cizí klíč je zde *Serial_name* s provázáním *cascade* na akci *delete* i *update*. Vztahy s ostatními entitami jsou:

- Více závodů může být v jednom seriálu.
- Jeden závod může mít více závodníků.
- Jeden závod může mít více kategorií.

Serial

Se závody úzce souvisí entita *Serial*, která zaznamenává informace ohledně seriálů. Primárním klíčem je název seriálu, tedy *name* a dalším povinným atributem je *points*, což znamená bodování závodníků v seriálu. Tabulka pracuje pouze se závody, tedy vztah je pouze, že jeden seriál může mít více závodů.

Category

Třetí entitu databáze reprezentuje *Category* sloužící pro záznamy ohledně kategorií. Primárním klíčem každé kategorie je její *idCategory* a povinně zadané atributy jsou název

kategorie *name* a zda je elitní tedy atribut *elite*. Cizím klíčem je potom *Race_name*, což představuje název závodu, v kterém kategorie existuje. Cizí klíč je nastaven tak, aby akce smazání či změna cizího klíče prolnula i do této tabulky. Dále nepovinné údaje jsou potom *birth_down, birth_up, gender_cat, maxRacers, numCircle, catEntry*, které označují spodní hranici narození, vrchní hranici narození, pohlaví, maximální počet závodníků, počet okruhu a startovné kategorie. Kategorie má dva následující vztahy:

- Jedna kategorie může mít více závodníků.
- Více kategorií může být v jednom závodě.

Result

Hlavní tři entity celé databáze a zároveň tři hlavní celky aplikace máme popsané. Další entitou je tabulka *Result*, kde budou zaznamenány výsledky závodníků. Primárním klíče je stále její *idResult* a cizím klíčem potom identifikační klíč závodníka, tedy *Racer_idRacer*. Zadané atributy musí být potom *time, state* vyjadřující čas v cíli závodníka a stav dojezdu. Vztah u této entity říká pouze, že jeden výsledek může mít jeden závodník.

Extra_data

Další záznamy jsou uchovány v tabulce *Extra_data*, která uchovává ostatní data při importu externích dat s jiným formátem. Jinak řečeno taková data, která nejsou vedena v tabulce *Racer*. Tato entita je prezentovaná primárním klíčem *idExtra_data* a obsahuje pouze jeden cizí klíč *Racer_idRacer*, kvůli určení ke kterému závodníkovi se mají data přiřadit. U cizího klíče se znovu jedná o nastavení cascade, jako v předešlé entitě. Atributy tabulky jsou řešeny povinným názvem *name* a povinnými daty *data*. Vztah u tohoto objektu říká, že více extra dat může mít jeden závodník.

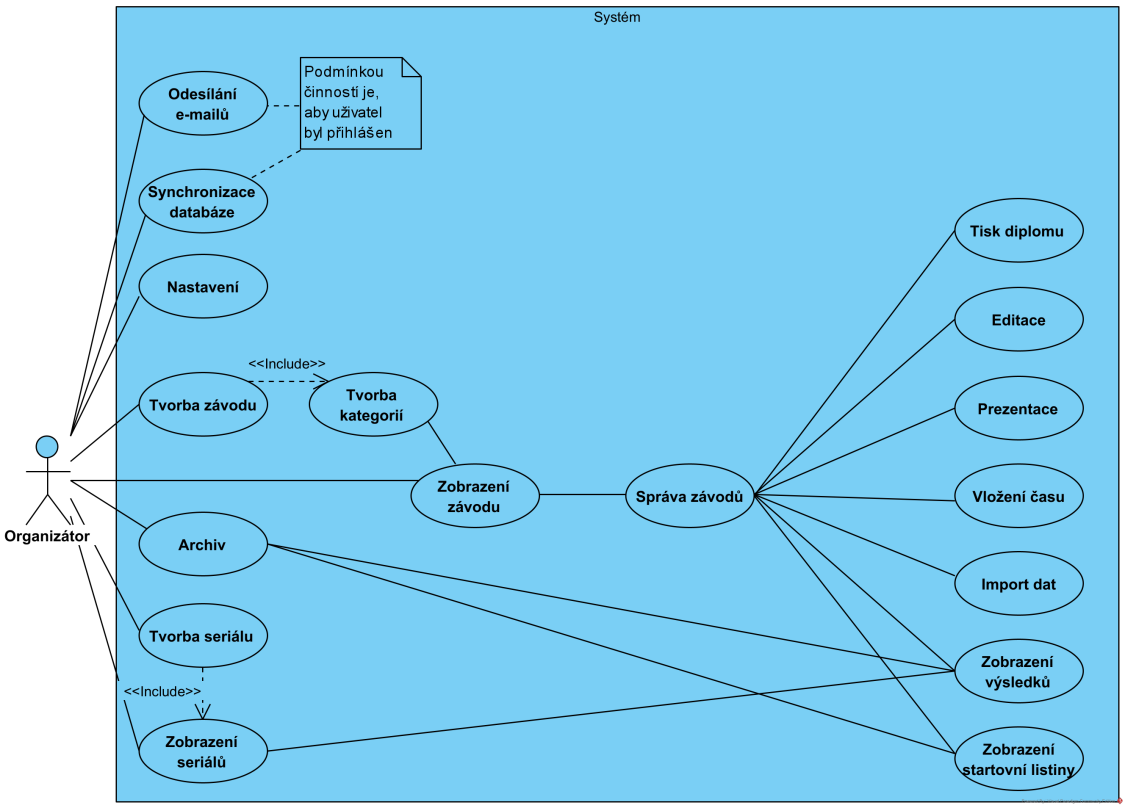
Completer

Tohle byly všechny entity v hlavním celku aplikace, avšak databáze obsahuje ještě jednu tabulku, která napomáhá vyplňování dat při prezentaci na místě. Jedná se o tabulku *Completer*, neboli našeptávač. Funkce tabulky je jednoduchá, pouze uchovává data všech závodníků registrovaných přes moji aplikaci a v budoucnu vám je bude nabízet k doplnění. Primárním klíčem je znovu *idCompleter* i když není nutný, jelikož tabulka se nikde sama nereprezentuje. Cizí klíče tedy nejsou potřeba. Atributy jsou potom stejné jako u entity *Racer*, kvůli zachování konzistentnosti dat. Vztah mezi jinými tabulkami neexistuje.

3.2 Struktura aplikace

Co se týče přístupu uživatele k aplikaci, tak bude rozdělen na přihlášeného a nepřihlášeného. Přihlášený uživatel bude mít přiřazen svůj e-mail v aplikaci a bude tak moct plně využívat funkce aplikace, jako například synchronizace databáze nebo odesílání e-mailů.

Tyto dvě funkce jsou jediné plus, co přihlášený uživatel bude mít. Na diagramu užití můžeme vidět, jak je aplikace rozvržena z pohledu uživatelských rolí a jejich činností (viz obr. 3.2).

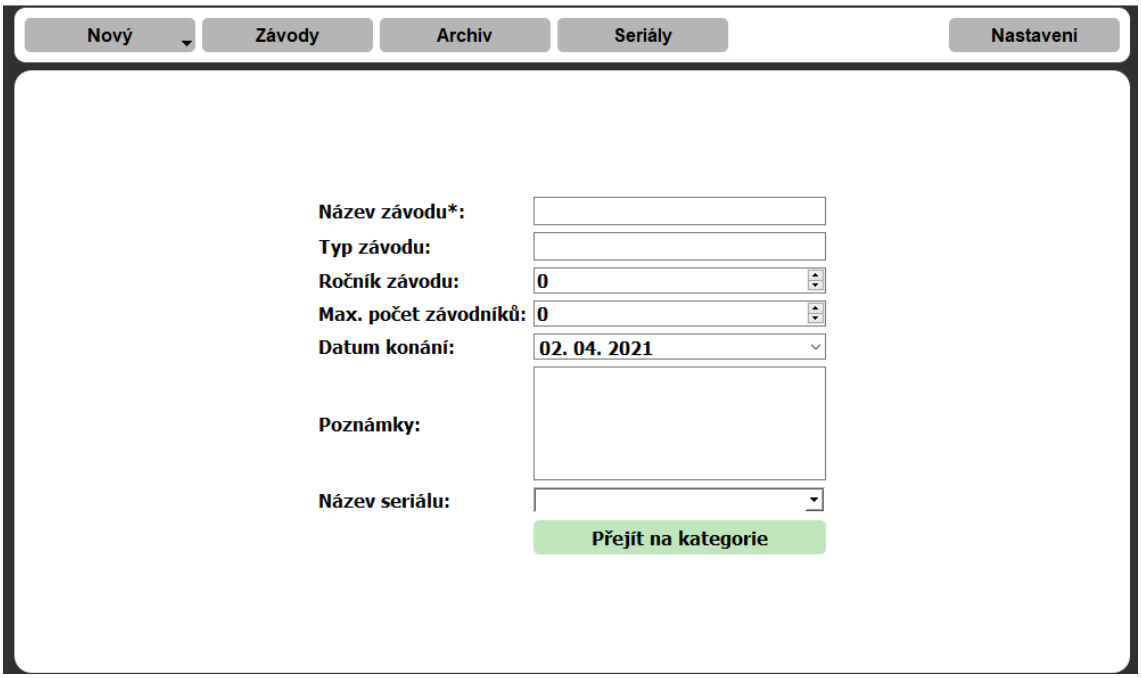


Obrázek 3.2: Use case diagram

3.3 GUI

3.3.1 Základní okno

Samotný vzhled a ovládání aplikace by mělo být co nejvíce intuitivní, aby každý dokázal aplikaci ovládat bez delšího vysvětlování. Uživatel by měl být schopen lehce a rychle přecházet mezi jednotlivými závody a zároveň poznat v jaké sekci se právě nachází. Aplikace dále musí být responzivní. Pro návrh jsem využil program QT Designer a převedl do jazyku *Python* pomocí knihovny *Pyuic5*.



Obrázek 3.3: Základní okno aplikace

Na obrázku 3.3 vidíme návrh hlavního okna aplikace. Okno bude rozděleno do dvou částí. První vrchní část bude pouze lišta s menu, které se bude měnit podle sekce, ve které se uživatel bude nacházet. Druhá část bude zobrazovací sekce a bude sloužit pro zobrazení obsahu. Po spuštění aplikace bude zobrazené pouze horní menu s pěti tlačítky. Bude se jednat o tlačítka *nový, závody, archiv, seriály a nastavení.*

- *Nový* – vysune menu s tvorbou závodu a seriálu. Po kliknutí na jedno z uvedených se do obsahové části vykreslí vstupní boxy pro vepsání jednotlivých dat. Po potvrzení se u závodu přejde rovnou do přidání kategorií se stejným způsobem zadání.
- *Závody* – pouze zobrazí jednotlivé závody jako tlačítka. Po najetí na jednotlivé závody se vysunou dodatečné informace o závodu, aby uživatel nemusel pro základní informace vstupovat do správy závodu. Po kliknutí na závod se otevře spravování. Pravý klik na závod otevře kontextové menu s funkcemi pro rychlejší pohyb v aplikaci.
- *Archiv* – archiv slouží pro zobrazení starších závodů. Pravý klik na jméno závodu otevře kontextové menu, kde bude možnost zobrazení startovní listiny a výsledků. Pravý klik dále zpřístupní obnovu a smazání závodu v archivu.
- *Seriály* – zobrazí tlačítka s aktuálními seriály. Po kliknutí je uživatel převeden do zobrazení všech závodů v seriálu a výsledků všech závodníku v seriálu. Uživatel si bude moct přepínat mezi kategoriemi a dále dané výsledky tisknout nebo exportovat.

- *Nastavení* – nastavení nebude sloužit pro správu závodů, ale pro správu aplikace. Bude se zde nacházet synchronizace s uživatelovým Google diskem a nebo také promazání našeptávače. Uživatel si také bude moct promazávat databázi.

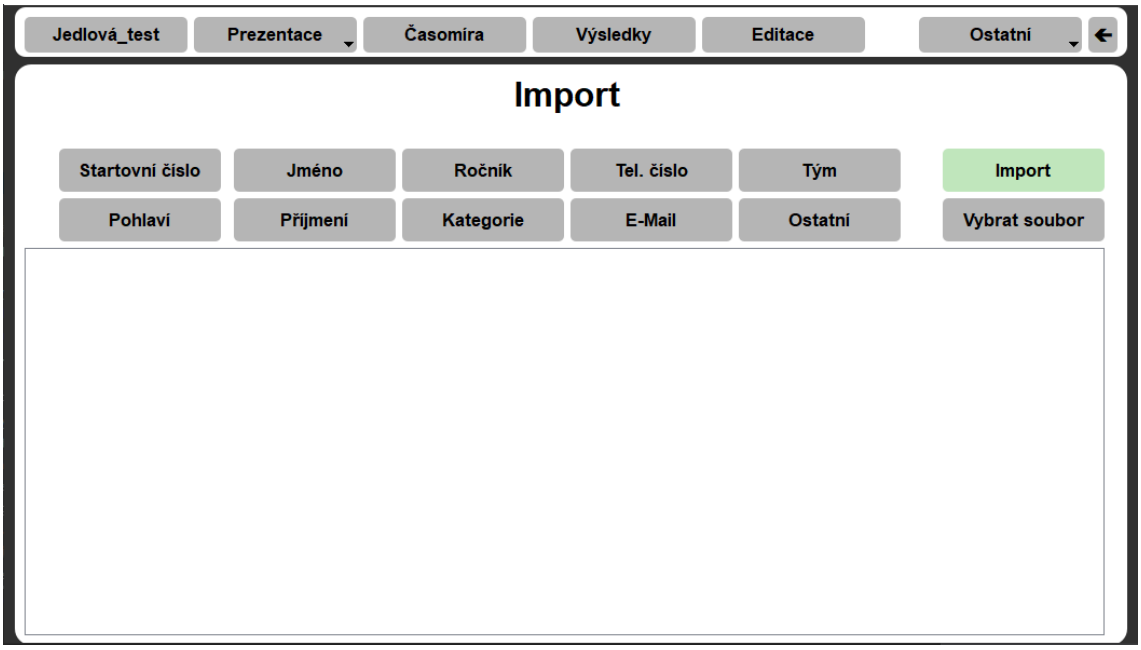
3.3.2 Správa závodu

Po kliknutí na tlačítko závodu se uživatel přenese do správy celé akce. Nejdřív se mu zobrazí základní informace o závodě a jeho kategorie. Menu se změní na podobu *název závodu, prezentaci, časomíru, výsledky, editaci, ostatní* a šipku pro vrácení do zobrazení závodu. Po kliknutí na jméno závodu v menu se znovu přeneseme do sekce základní informace.

- *Prezentace* – tlačítko *prezentace* bude navrženo jako vysouvací menu s prvky *startovní listina, prezentace na místě a import.*
  - *Startovní listina* – jednoduše zobrazí zaregistrované závodníky a jejich základní informace v přehledné tabulce. Nad tabulkou budou dvě tlačítka *tisk* a *export* tabulky do formátu *PDF.*
  - *Prezentace na místě* – sekce pro vkládání závodníků do databáze. U této části se obsahová část rozdělí na další dvě části. A to do části pro vepsání dat o závodníkovi a log. Log bude prezentován jako tabulka s daty.
  - *Import* – část pro vkládání externích dat s jiným formátem do aplikace a následně do databáze. V horní části obsahové části najdeme tlačítka, ke kterým se budou přiřazovat sloupce externích dat. Pokud daný sloupec externích dat aplikace nezná tak použijeme tlačítko *ostatní.* Pro potvrzení importu bude nutno mít přiřazeny základní informace jako jméno, příjmení, pohlaví, rok narození. Vše se potvrdí tlačítkem *import.* Na obrázku 3.4 můžeme vidět základní podobu sekce *import.*

- *Časomíra* – tlačítko *časomíra* nás převede do zobrazení startovní listiny s funkcí vložení času. Vložení času se bude dát zapsat dvěma způsoby. Můžeme přiřadit čas libovolnému startovnímu číslu, které budeme vkládat nad tabulkou a zároveň stav závodníka. Stavem je myšleno, zda závodník dojel, tak jak se má nebo byl diskvalifikován. Čas půjde vkládat i skrze tabulku pomocí zapsaní času do sloupce *čas.*

- *Výsledky* – zobrazení výsledků bude řešeno stejně jako startovní listiny tedy tabulkou. Všechna data bude potřeba tisknout a exportovat pomocí dvou tlačítek, kde tlačítka budou nabízet i náhled finální podoby dokumentu. Bude možnost si zobrazit celkové výsledky a výsledky pro každou kategorii zvlášť pomocí rolovacího menu. Dva poslední sloupce tabulky bude odstup od předešlého závodníka a pořadí v závodu. Pokud bude závod zapsán v seriálu tak v tabulce bude zapsáno i kolik závodník získal bodů. Pokud budou zobrazovány celkové výsledky tak bude uvedeno ještě pořadí v kategorii.



Obrázek 3.4: Design navržen pro import dat

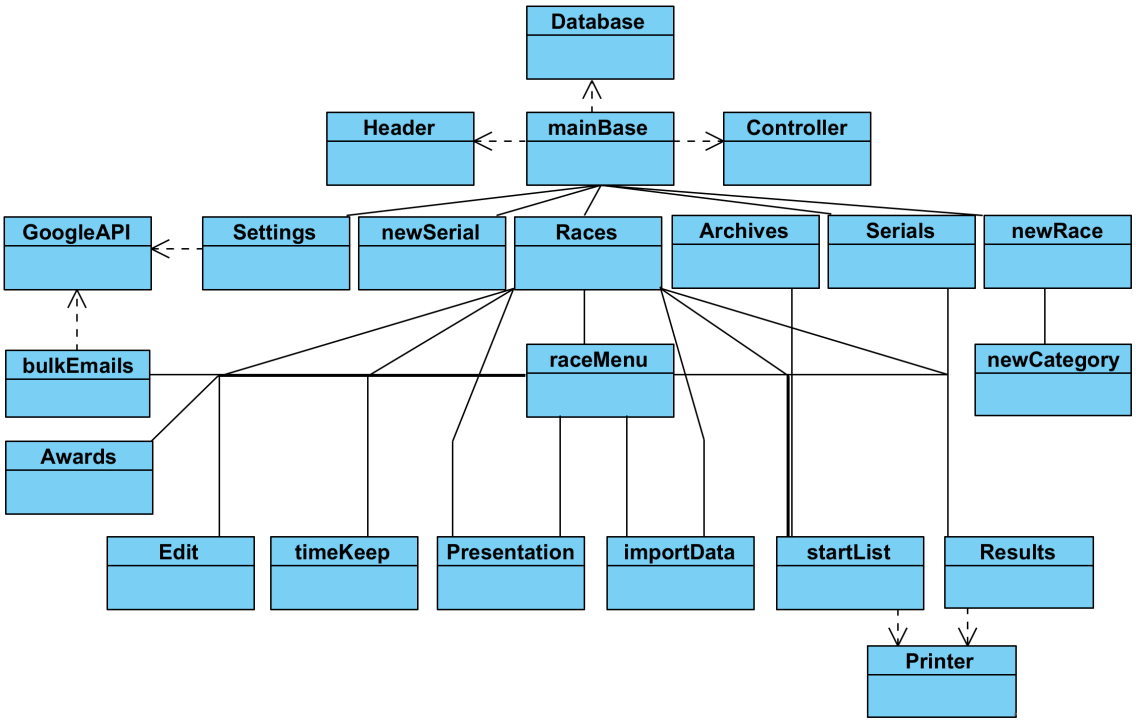
- *Editace* – sekce editace bude situována podobně jako sekce importu. V horní části bude na výběr z tlačítek pro editaci závodu, kategorií a závodníků. Poslední tlačítko bude sloužit pro potvrzení. Ve zbytku obsahové části budou vyobrazena data v tabulce. Editace bude probíhat jednoduchou úpravou dat v tabulce. U kategorie přibude tlačítko pro přidání a smazání kategorie. U závodníků pouze tlačítko pro smazání.
- *Ostatní* – tlačítko ostatní bude chápáno jako menu, kde si uživatel bude moci vybrat z tisku diplomu nebo posílání hromadných e-mailů. Tisk diplomů bude řešen několika tlačítky, kterými si uživatel nejprve navolí šablonu a následně pozice určitých dat. V posílání hromadných e-mailů bude uživateli nabídnuta oblast pro vepsání zprávy a možnost přiložit soubor.

Kapitola 4

Implementace

Implementaci jsem provedl v programu PyCharm od firmy JetBrains. PyCharm je vývojový nástroj pro programovací jazyk *Python*, který nabízí velmi přehledné a rychlé prostředí pro vývoj. V předešlé kapitole byl také zmíněn Qt Designer pro tvorbu modelů *GUI*. Dále jsem využíval databázový systém DataGrip, také od firmy JetBrains, pro velice přívětivé zobrazení databáze ve formátu *SQLite*.

Celkový vzhled aplikace nám více přiblíží diagram tříd (viz obr. 4.1). Jedná se o *UML* popisující strukturu celé aplikace z pohledu implementace.



Obrázek 4.1: Diagram tříd

4.1 Popis jednotlivých celků aplikace

V této sekci stručně popíši jednotlivé prvky aplikace a jejich funkce popřípadě i důležité proměnné. Dost často se jednotlivé proměnné využívají znovu skrze celou aplikaci. Popíši je tedy pouze v místě prvního výskytu. Dílčí funkce programu popíši podrobněji. Programem budu postupovat hierarchicky tak, jak by uživatel měl postupovat při prvotním vstupu do aplikace.

4.1.1 Definice okna aplikace

Třída *mainBase* nadefinovává hlavní okno aplikace. Po spuštění aplikace se zobrazí vše, co v tomto souboru nadefinujeme. Je zde určeno, jaké má být rozlišení aplikace a celkový styl. Také je zde zavolána funkce na výpis menu, pomocí kterého je uživatel dále přesměrován do dalšího celku. Hlavními proměnnými jsou potom:

- *MainWindow* – hlavní okno aplikace.
- *mainWidget* – hlavní widget, do kterého se vypisuje to, co aplikace vygeneruje.
- *gridLayout* – rozvržení, do kterého budou umísťovány další widgety.

4.1.2 Generování menu

Elementy v sekci menu se mění pouze podle aktivity uživatele. To zajišťuje třída *header*. Spravuje-li uživatel závod, menu se mění na podobu správy závodu. Celkové fungování této třídy je založeno na tom, že třída dostane parametry *mainWidget* a *gridLayout*. Do těchto dvou widgetů následně vykreslí požadovaný obsah. Dále se vygenerované widgety vrátí zpátky do třídy, odkud byla funkce pro generování menu zavolána.

Základem této třídy jsou dvě metody *printBaseHeader* a *printRaceHeader*. Tyto dvě metody mají v základu fungování stejné, jenom vykreslují jiný obsah. Třída se také stará o vysouvací menu, které tato sekce může nabízet. Redukuje tak zbytečnou redundanci metod v různých třídách.

Tato třída se také stará o přesun v aplikaci. Obsahuje metody, které jsou napojeny na kliknutí daného tlačítka, sloužící pro zavolání potřebné třídy a následného vygenerování nového obsahu. Je to kvůli zbytečnému opakování volání metod v jednotlivých třídách. Obě metody tedy dostanou widgety *mainWidget*, *gridLayout* a *gridLayout_2*, metoda smaže starý obsah v druhém layoutu a následně vygeneruje nový obsah volané třídy. Typicky tato metoda vypadá jako na ukážce číslo 4.1

```
1 def serials(self, mainWidget, gridLayout, gridLayout_2):
2     from serials import Ui_serials
3     controller().deleteLayout(gridLayout_2)
4     Ui_serials().setupUi(mainWidget, gridLayout)
```

Ukázka 4.1: Metoda pro předání fungování jiné třídě

4.1.3 Tvorba závodu

Tento požadavek je řešen v třídě *newRace*. Jedná se o první sekci, kde uživatel může nahrát data do databáze skrze aplikaci. Jelikož jde o sběr dat o závodu, mění se tu pouze obsahová část aplikace. Pomocí několika widgetů typu *lineEdit* dostaneme zadaná data do paměti aplikace. Třída má jen dvě metody:

- *toCat_clicked* – metoda napojená na stisknutí tlačítka nejdříve zavolá *writeRace* a potom přenese uživatele do dalšího souboru.
- *writeRace* – nejdříve zkontroluje platnosti vstupů a následně zapíše vložený závod do databáze.

4.1.4 Přiřazení kategorie

Dalším bodem je přiřazení kategorií k jednotlivému závodu. To je řešeno v třídě *newCategory*. Po přesunu z předešlého celku se přegeneruje obsahová část na několik dalších widgetů typu *lineEdit* a tabulku sloužící pro log zadaných kategorií. Zde jsou pouze dvě metody. Jedna je připojena na tlačítko *Přidat kategorii* a druhá na tlačítko *Dokončit*

- *addCat_clicked* – metoda přidá vloženou kategorii do databáze a zároveň zkontroluje platnost vstupů. Metoda také zapíše vložená data do tabulky logu. Zároveň tu probíhá kontrola duplicity s už vytvořenými kategoriemi.
- *complete_clicked* – nahraje zadaná data a přesune uživatele do sekce zobrazení závodů.

4.1.5 Tvorba seriálu

Tato funkce je naprogramována v *newSerial* a je dost podobná sekci *Tvorba závodu*. Data od uživatele jsou uložena do paměti aplikace pomocí několika widgetů. Po stlačení tlačítka *Potvrdit* se odešle dotaz typu *insert* do databáze s vloženými daty. Třída využívá pouze jediné metody a tím je:

- *WriteSerial* – zkontroluje a zapíše zadané hodnoty do databáze.

4.1.6 Zobrazení závodů

Races řeší zobrazení všech aktuálních závodů. Dále nabízí kontextové menu, které se zobrazí kliknutím pravým tlačítkem myši, pro urychlený přesun v aplikaci. Kontextové menu je řešeno pomocí widgetu *QMenu*, ke kterému se přiřadí jednotlivé *QAction*. Maximální počet zobrazených závodů je padesát, kvůli přehlednosti. Po najetí na každé tlačítko závodu se zobrazí základní informace o závodu, což je řešeno pomocí widgetu *QToolTip* a generováno při výpisu. Tyto funkce vyžadují několik jednotlivých metod:

- *printButtons* – načte všechny závody z databáze a následně je vypíše do widgetu s kontextovým menu a s informacemi po najetí na tlačítko. Problém je řešen jednoduchým cyklem *for*.

- *actionClicked* – metoda, která zjišťuje který prvek z kontextového menu byl zvolen. Dále pouze vyhotoví funkci vybraného prvku. Při zvolení prezentace, startovní listiny, časomíry, výsledků a editace se obsahová část přegeneruje novou třídou. U smazání či přidání závodu do archivu se pošle dotaz na databázi a zároveň se aplikace pomocí *pop-up* okna zeptá, zda uživatel danou operaci chce opravdu uskutečnit, pro prevenci uživatelské nepozornosti.

- *context_menu* – funkce pro určení, u kterého závodu bylo vybrané kontextové menu.

4.1.7 Správa závodu

Tato sekce slouží uživateli jako hlavní rozcestník ve správě určitého závodu. Uživateli se tu zobrazí základní data zadaná při tvorbě závodu a zadané kategorie s informacemi. Ke každé kategorii je dále spočteno kolik závodníků každá kategorie obsahuje. Všechny informace jsou zobrazeny ve widgetech typu *QtTableWidget*. Menu bylo přegenerováno tlačítky pro správu. Poprvé se celé menu přegenerovalo tedy celý *gridLayout* se musel smazat.

4.1.8 Startovní listina

Jedná se o jednoduché zobrazení všech zaregistrovaných závodníků v aktuálním závodu. Všichni závodníci jsou vypsáni do tabulky s jejich informacemi. Všechna data lze tisknout nebo exportovat pomocí tlačítek. Obě tlačítka jsou napojena na zavolání funkcí *print* a *export* ze třídy *Printer*, kterou si také rozebereme. Export a tisk probíhá pouze pro prvních 7 sloupců, jelikož na startovní listině nesmí být e-mail, telefonní číslo nebo jiné osobní údaje. Všechna data v tabulce jdou řadit podle velikosti a nebo podle abecedy po kliknutí na daný sloupec. Za základní informace, které má k dispozici aplikace, se vypíšou ostatní data z importu. Metody, zajišťující zobrazení dat jsou:

- *writeRacers* – načte a vypíše data o závodnících do tabulek.

- *writeOthers* – vypíše ostatní informace o závodnících, pokud jsou k dispozici. Jedná se o data, která nejsou implementovaná v základu aplikace. Může se jednat o přídavné informace od uživatele například datum registrace. Tato data jsou selektována z tabulky *Extra_data*

- *find_indices* – metoda pro analýzu datového typu *list*. Funkce vrací jednotlivé indexy na kterém se vstupní element nachází.

4.1.9 Prezentace na místě

Obsahová část se přegeneruje na několik *lineEditů* a *QTableWidget* pro historii vložených závodníků. Prezentace na místě je první část implementace, kde jsem se potkal s problémy. Bylo totiž zapotřebí zakomponovat do třídy několik nezbytných metod pro následné přívětivější fungování. Jednalo se například o automatické přiřazení startovního čísla, našeptávač na základě už vložených dat o závodnicích nebo také automatické přiřazení kategorie k nejbližší možné. Tyto důležité funkce bych popsal v jednotlivých odstavcích. Tato třída má pouze jednu funkci, která není spojena se zmíněnými funkcemi a tou je:

- *add_racer* – validuje a zapíše závodníka do databáze.

1. **Automatické přiřazení startovního čísla**

   Uživateli je nabídnuta možnost vepsat startovní číslo ručně, kde bude kontrolována duplicita a nebo to aplikace udělá za něj. Automatický způsob ušetří dost času při registraci na místě. Toto pole tedy obvykle uživatel vůbec nemusí řešit. Kód můžeme vidět na ukázce 4.2

   - *set_startNumber* – nejdříve se funkce dotáže databáze na startovní čísla, která už závodníci mají. Pokud závod už má nějaké závodníky, tak najde chybějící čísla v posloupnosti a začne je doplňovat, jinak vezme poslední číslo a přičte k němu 1. Pokud závod ještě nemá závodníky tak se začne startovním číslem 1.
   - *find_missing* – nalezne chybějící čísla v datovém typu list.

```
    def set_startNumber(self):
        start_nums = self.db.query("SELECT start_num FROM Racers "
                                   "WHERE Racers.Race_name = '{0}' ORDER BY start_num".format(self.name))

        # Aby aplikace nespadla pri startovnim cislu typu string
        start_nums = [num for num in start_nums if isinstance(num, (int,float))]

        if (start_nums != []):
            miss = Controller().find_missing(start_nums)
            start_nums = sorted(start_nums)
            if (miss == [] and start_nums != []):
                start_num = start_nums[-1] + 1
            else:
                start_num = miss[0]
        else:
            start_num = 1

        self.starNumber.setValue(start_num)
```

Ukázka 4.2: Metoda pro automatické přiřazení startovního čísla

2. **Našeptávač**

   Automatické doplňování neboli našeptávač slouží uživateli pro rychlejší zadání častých účastníků závodů. Pokud účastník už někdy byl registrován stačí začít psát jeho příjmení a po potvrzení doplnění se vše vyplní za uživatele. Velmi to urychlí zejména registraci na místě na menších závodech, kde se závodníci dost často opakují. Pokud se zadané příjmení shoduje se záznamem z databáze, aplikace vám ho nabídne. Vypíše první příjmení, křestní jméno a rok narození, kvůli totožnosti příjmení a jména. V nastavení je i uživateli nabídnuto tuto historii vložených závodníků vymazat a začít tak znovu.

   - *surname_changed* – našeptávač je připojen na začátek psaní příjmení závodníka, tedy na změnu *lineEditu surnameRacer* funkce si nejdříve ověří, zda příjmení bylo zadáno a následně ho ověří s daty z databáze. Pokud se shodují tak se po kliknutí doplní data do jednotlivých *lineEditů*.
   - *saveCompleter* – uloží vloženého závodníka do tabulky *Completer*, pokud tam daný záznam už není.
   - *loadCompleter* – načte data z tabulky *Completer* a připraví je pro doplnění do *lineEditů*.

3. **Automatické přiřazení kategorie**

   Cílem této funkce bylo uživateli nabídnout možnost, aby aplikace rozhodla o kategorii za něj pouze na základě roku narození vkládaného závodníka. Měla by ulehčit vkládání závodníků do závodu, který má více kategorií. Pokud aplikace zjistí, že vložený rok narození se neshoduje s žádným rozsahem kategorií, automaticky se zeptá, zda uživatel chce přiřadit nejbližší kategorii, avšak už nehledí na pohlaví kategorie. Celý algoritmus se potom skrývá ve funkci:

   - *determineCat* – nejdříve se zjistí, že není zaškrtnut *checkBox* pro elitní kategorii. Pokud není, funkce posílá dotaz na databázi. Nejdříve se zjišťuje, zda existuje kategorie s rozsahem, který odpovídá roku narození vloženého závodníka a zároveň jeho pohlaví. Následuje podmínka, která rozhodne zda předešlý dotaz něco vrací. Pokud ne, aplikace se dotáže uživatele, zda chce kategorii přiřadit. Při souhlasu se posílá další dotaz na databázi stejného typu, ale už bez ohledu na pohlaví závodníka.

4.1.10 Import dat

Dalším důležitým sektorem pro funkčnost aplikace je import dat. Jedná se o import dat s formátem typu *XLSX*, *CSV* a *ODS*, což považuji za nejvíce používané formáty pro uložení dat tohoto typu. Popřípadě rozšíření tohoto seznamu není v pozdějším vývoji problémem.

Po výběru importovaného souboru se zobrazí jeho obsah v tabulce a uživatel jednoduše začne přiřazovat, které sloupce odpovídají jednotlivým datům v aplikaci. Po přiřazení sloupec z tabulky zmizí a uživatel může pokračovat v importu. Nutná data jsou jméno, příjmení, pohlaví a rok narození závodníka. O základní fungování se starají funkce:

- *button_checked* – funkce, která zjistí který *QPushButton* byl zvolen a uloží jeho název do proměnné. Zároveň nastaví *boolean* proměnnou *checked* na *true* pro zamezení výběru dat stejného typu.
- *assign* – uloží importovaný sloupec do proměnné a skryje už importovaný sloupec. Zároveň vypne už vybrané tlačítko pro zamezení znovu vybrání už importovaných dat.
- *openFile* – vygeneruje *pop-up* okno pro získání absolutní cesty k importovanému souboru. Soubor se uloží do proměnné a podle koncovky souboru se data rozparsují do proměnné pro pozdější zpracování.

Uživateli je nabídnutá možnost importu „ostatních" dat. To jsou data, která aplikace nezná a nepotřebuje je ke svému fungování. Slouží tedy pouze uživateli pro vložení dodatečných informací a dále nejsou editovatelná. Tato data jsou nahrána do databázové entity *Extra_data* a zobrazena v sekci startovní listina. Může se jednat o data typu datum registrace, země, město atp. Tuto problematiku řeší metoda:

- *writeOthers* – metoda, která dostane vložená data a zjistí identifikátory vložených závodníků. Dále tyto identifikátory vloží společně s jejich daty do tabulky *Extra_data*.

Pro vložení dat do databáze tato třída používá dvě základní metody:

- *writeCats* – funkce na vložení kategorií do databáze. Ty se přiřadí k aktuálnímu závodu, avšak ne k závodníkům. Je to kvůli možnému rozdílu v ročnících kategorií v aplikaci a importovaných datech. Přiřazení kategorií k závodníkům proběhne až po určení jejich rozmezí, které nelze pouze z názvu jednoznačně zjistit.
- *writeToRacers* – tato sekce třídy slouží pro vložení závodníků do závodu. Je zde také řešeno automatické přiřazení kategorie, pokud má závodník vyplněný rok narození a pohlaví. Na základě těchto dvou údajů se aplikace pokusí určit jeho kategorii pomocí dotazu na databázi.

4.1.11 Časomíra

Časomíra je řešena pomocí tabulky, v které jsou zobrazeni závodníci a možnosti zapsání času přímo do tabulky. Uživatel také může vložit startovní číslo, čas a stav mimo tabulku, pokud nechce hledat startovní číslo v tabulce. Při vyšším počtu závodníků to je vhodnější způsob.

- *writeTime* – po stlačení tlačítka *Potvrdit* se podle startovního čísla zjistí id závodníka a k němu se přidá čas se stavem dojezdu a následně je celá tato informace odeslána do databáze. Také se zde kontroluje zda závodník už nemá přiřazený čas.
- *writeTable* – vypíše vkládaný čas a stav k vhodnému závodníkovi podle jeho identifikačního čísla.

- *cellChanged* – funkce napojena na změnu buňky s časem. Pokud uživatel napíše čas do tabulky zavolá se tato metoda. Ta zkontroluje, zda je vložený čas a stav závodníka, následně pošle dotaz typu *insert* do databáze, kde se do entity *Result* vloží daný čas a stav k identifikačnímu číslu závodníka.

4.1.12 Výsledky

Tato třída řeší finální produkt této aplikace. Tím jsou výsledky jednotlivých závodů s pořadím a časy závodníků. Po vstupu do sekce výsledků pro jednotlivý závod se v aplikaci zobrazí celkové výsledky. Celkové výsledky jsou řešeny na straně databáze pomocí *SQLite* dotazu v metodě:

- *complete_results* – zavolá se po kliknutí v menu na celkové výsledky a nebo po vstupu do této sekce. Dotáže se databáze pomocí *SQLite* dotazu na celkové výsledky a seřadí je podle času a stavu. Dále následuje zavolání funkce na výpis dat do tabulky. Dotaz, který je využíván touto metodou můžeme vidět na ukázce 4.3

```
SELECT start_num,forename,surname,birth,
C.name,team,gender,time,state
FROM Results
LEFT JOIN Racers R on R.idRacers = Results.Racers_idRacers
LEFT JOIN Category C on R.Category_idCategory = C.idCategory
WHERE R.Race_name = '{0}' ORDER BY CASE
WHEN state = '' THEN 1
WHEN state = '-1' THEN 2
WHEN state = '-2' THEN 3
WHEN state = 'DNF' THEN 4
END,time
```

Ukázka 4.3: Dotaz pro vrácení pořadí z výsledků

Po kliknutí na element v rolovacím menu se zavolá následující metoda, která rozhodne zda se jedná o celkové výsledky nebo kategorii. Podle toho se bude odvíjet skládání dotazu na databázi.

- *combobox_changed* – metoda, která je napojena na změnu elementů v rolovacím menu. Metoda rozhodne zda se jedná o celkové výsledky a nebo kategorii. Podle toho připraví data pro výstup a nastaví tabulku podle daných dat.

Po vyhotovení dotazu ze strany databáze už máme srovnané výsledky a zbývá je jenom vepsat do tabulky. To je jednoduše řešeno v metodě *writeRacers*, která k výpisu přidá ještě pořadí závodníka a pokud se jedná o celkové výsledky přidá se ještě pořadí v kategorii. Pořadí v kategorii je získáno pomocí metody *getCatOrder*. Výsledky pro seriál jsou řešeny samostatně ve své třídě.

- *getCatOrder* – pro vložené startovní číslo závodníka vrátí pořadí ve vložené kategorii daného závodu.

4.1.13 Editace

Třída *edit* je rozhodně nejobsáhlejší z celé aplikace. Je v ní řešená celková pozdější editace závodu, kategorií nebo závodníků. Hlavní okno se tedy generuje podle toho, na jaké tlačítko uživatel klikne. Podle toho se vypíší aktuální data, která se dají lehce editovat v tabulce. Všechna data jsou validována pomocí regulárních výrazů, aby uživatel nezadal něco co aplikace neočekává. Pro potvrzení a odeslání dotazu do databáze se používá tlačítko *Potvrdit*. Popis této třídy bych rozdělil do dalších tří skupin.

1. **Editace závodů**

   U úpravy závodů se vypíše pouze jeden řádek, který lze upravit. Pro úpravu je nutné dodržet několik základních pravidel jako například, že název závodu už neexistuje nebo je nutné měnit název seriálu na už vytvořený seriál. Celá úprava je řešena dvěma metodami:

   - *loadRace* – načte data z databáze přiřazené k závodu s aktuálním názvem. Dále se nastaví tabulka na aktuální data.
   - *writeRace* – zkontroluje vstupní data a zkontroluje duplicitu jmen v databázi. Následně upraví závod v databázi.

2. **Editace kategorií**

   Samotná editace kategorií už je složitější. Přidával jsem totiž do aplikace funkci pro rekalkulaci kategorií po změně rozmezí ročníků kategorie. Tato metoda se nazývá *recalcCats* a bude popsána později. Co se týče základní manipulace tak uživatel může kategorie mazat a nebo přidávat. K tomu slouží tlačítka, která požadovaný úkon vykonají. Pro smazání kategorií je samozřejmě nutno označit, kterou kategorii chceme smazat. Obsluhu této sekce zařizuje několik metod:

   - *delete_clicked* – multifunkční metoda pro smazání závodníka a nebo kategorie
   - *addCat_clicked* – přidá řádek k tabulce pro novou kategorii.
   - *writeCat* – validuje a zapíše kategorie do databáze.
   - *loadCat* – načte kategorie z databáze.
   - *recalcCats* – metoda sloužící pro přepočet kategorií. Načte všechny závodníky daného závodu a zkontroluje například zda jejich kategorie odpovídá jejich ročníku. Pokud ne tak jim přiřadí jinou a nebo žádnou. Kontrola probíhá na straně databáze pomocí dotazu. Nabídka, z které dotaz vybírá je samozřejmě pouze z normálních kategorií, tedy ne elitních. Dotaz, který metoda využívá je na ukázce 4.4

```
SELECT CASE
WHEN
Race_name= '{0}' AND birth_down <= '{1}'
AND birth_up >= '{1}'
AND gender_cat = '{2}' AND elite = '0'
THEN Category.idCategory
WHEN
Race_name= '{0}' AND birth_down <= '{1}'
AND birth_up >= '{1}'
AND gender_cat = '' AND elite = '0'
THEN Category.idCategory
ELSE ''
END AS d
FROM Category WHERE d is not '
```

Ukázka 4.4: Dotaz na určení kategorie závodníka

3. **Editace závodníků**

   Velice podobná sekce jako editace kategorií. Lze zde také závodníky mazat a upravovat. Po každé editaci závodníků se také volá metoda *recalcCats* kvůli kontrole kategorií u každého závodníka. Jelikož pokud jsme změnili rok narození u závodníka může nastat možnost změny kategorie.

   - *writeRacers* – validuje vstupní informace a následně je upraví v databázi.
   - *loadRacers* – načte všechny závodníky k danému závodu a vypíše je do tabulky.

4.1.14 Seriály

Celková manipulace se seriály je definována v třídě *serials*. Třída řeší zobrazení závodů v seriálu a nebo také výsledků. Jediný problém v této třídě bylo sjednocení výsledků z jednotlivých závodů pro jeden seriál. Většina metod je pouze implementována na problematiku seriálu jinak už byly použity v předešlých třídách. Metoda, kterou bych popsal právě řeší sjednocení výsledků.

- *concatCats* – pro zvolenou kategorii sjednotí všechny shodné závodníky a sečte jejich body. Tyto výsledky se potom zapíšou do tabulky srovnané vzhledem k bodům. Kód můžeme vidět na ukázce 4.5

```
def concatCats(self,data,cat):
    results = []
    res = list(filter(lambda x: cat in x, data))
    tmp = [l[:5] for l in res]
    dupl = []
    for elem in tmp:
        if elem not in dupl:
            dupl.append(elem)
        if(len(res)>=1):
            for x in dupl:
                sum=0
                indices = [index for index, value in enumerate(tmp) if
    value == x]
                for index in indices:
                    sum = int(sum) + int(res[index][5])
                x.insert(len(x),sum)
                results.append(x)
    return results
```

Ukázka 4.5: Kód na sjednocení výsledků závodníků

4.1.15 Controller

Třída kontroléru obstarává funkce, které se prolínají celou aplikací například mazání starého obsahu z layoutu. To je řešeno metodou *deleteLayout* (viz obr. 4.6), zde je využívaná logika potomků v aplikaci.

Dále tato třída řeší validaci vstupů v celé aplikaci. Validace je řešena pomocí regulárních výrazů a pomocí validátoru z třídy *QRegExpValidator*. Třída mimo jiné obsahuje metody, které se využívají skrze celou aplikaci, jako například metoda *writeIntoTable* pro vypsání dat do tabulky a její nastavení.

```
    def deleteLayout(self, layout):
        if layout != None:
            while layout.count():
                child = layout.takeAt(0)
                if child.widget() is not None:
                    child.widget().deleteLater()
                elif child.layout() is not None:
                    self.deleteLayout(child.layout())
```

Ukázka 4.6: Metoda pro smazání obsahu z layoutu

4.1.16 Printer

Třída řešící export a tisk startovní listiny nebo výsledků. Třída nabízí export dat do *PDF, XLSX* nebo tisk.

Při exportu dat se vygeneruje dialogové okno s kategoriemi pro zaškrtnutí. Uživatel si tedy může vybrat, které kategorie chce exportovat. Tato logika je uplatněna nejen u výsledků závodu, ale i u,výsledků seriálu a startovní listiny.

Po zaškrtnutí požadovaných kategorií třída vygeneruje *HTML* kód, který je následně předán metodám pro export nebo tisk. Metoda pro vygenerování *HTML* je znázorněna na obrázku 4.7

Pro export dat z tabulky jsem použil třídu *QPrinter*, která slouží pro výstup dat z aplikace. Třída je založená na stejném principu jako třídy *QWidget* nebo *QPixmap*.

```
model = table.model()
        html = """<html>
               <head>
               <style>
               table, th, td {
               border: 1px solid black;
               border-collapse: collapse;
               }
               th, td {
               padding:5px;
               text-align:center;
               }
               h1{
               text-align:center;
               }
               </style>
               </head>"""
        html += "<h1>" + type + "</h1>"
        html += "<h1>" + catName + "</h1><table width=100%><thead>"
        html += "<tr>"
        for c in range(model.columnCount()):
            if (c <= 6):
                html += "<th>{}</th>".format(model.headerData(c, QtCore
    .Qt.Horizontal))

        html += "</tr></thead>"
        html += "<tbody>"
        for r in range(model.rowCount()):
            html += "<tr>"
            for c in range(model.columnCount()):
                if (c <= 6):
                    html += "<td>{}</td>".format(model.index(r, c).data
    () or "")
            html += "</tr>"
        html += "</tbody></table>"
        html+= "</html>"

        return html
```

Ukázka 4.7: Kód pro vytvoření HTML kódu pro export

- *print* - zajišťuje tisk dat. Metoda dostane tabulku s daty, nadpis a vygenerované *HTML*. Po vytvoření objektu z třídy *QPrinter* aplikace vytvoří dialogové okno pomocí *QPrintDialog*. Tento objekt se následně předá printeru a započne tisk.
- *preview* - funkce pro náhled. Sestavení náhledu je podobné jako u tisku. Funkce dostane už sestavené *HTML*, které se vloží do objektu třídy *QTextDocument* a to se potom předá dialogovému oknu, ve kterém se náhle zobrazí.

- *export* - export je řešen podobným procesem jako u ostatních metod. Nejdříve se nastaví základní data pro *QPrinter* s úložištěm. *HTML* se potom předá printeru pro export.

4.1.17 Tisk diplomu

Přidaná funkce do aplikace je tisk diplomu podle uživatelovi šablony. Šablona musí být formátu JPG nebo PNG. Po vybrání cesty k šabloně se otevře a uživatel vybere pozici pro umístění daných dat. Aplikace potom vygeneruje diplomy do formátu *PDF*.

Celou problematiku řeší třída *Award*, která obsahuje další tři podtřídy:

- *AwardImage* – vyvolá okno, kde bude zobrazena šablona pomocí třídy *Tkinter*. Dále nastaví plátno a připojí k němu další podtřídy.
- *SelectionObject* – třída řešící zobrazení vybraného pole v okně.
- *MousePositionTracker* – sleduje pozici myši.

Sama třída *Award* má několik metod:

- *imageOpen* – otevře okno pro výběr cesty k šabloně.
- *open* – vytvoří objekt z třídy *AwardImage* a vloží do ní zvolenou šablonu.
- *export_clicked* – uloží diplom do formátu *PDF*.
- *getData* – získá výsledky ze závodu pro vypsání do šablony.
- *generateImages* – vypíše všechna zvolená data do šablony.
- *setFont* – nastaví velikost písma podle velikosti šablony. Kód můžeme vidět na ukázce 4.8.

```
1     def setFont(self):
2         """
3         Metoda pro urceni optimalni velikosti fontu podle velikosti
      okna a delky slova
4         Vraci:
5             Font s optimalni velikosti
6         """
7         img_fraction = 0.095
8         fontsize = 1
9         font = ImageFont.truetype("/arial.ttf", fontsize)
10        while font.getsize(self.coords[0][0])[0] < img_fraction * self.
      app.img.width():
11            fontsize += 1
12            font = ImageFont.truetype("/arial.ttf", fontsize)
13        return font
```

Ukázka 4.8: Kód pro nastavení velikosti písma

4.2 Synchronizace databáze

Prvotní využití *GoogleAPI* je pro synchronizaci databáze. Pokud uživatel přejde do nastavení aplikace, nabídne se mu možnost nahrání databáze na svůj Google disk a nebo stažení databáze ze svého Google disku. Po kliknutí na jedno z uvedených tlačítek, vytvoří aplikace objekt z třídy *GoogleAPI* a otevře browser pro přihlášení k vašemu Google účtu. Následně vše probíhá automaticky pomocí *API*.

Pro ověření je nutno založit aplikaci Google projekt přímo na stránkách *GoogleAPI* [8] a přiřadit k ní uživatele, kteří ji budou moci používat. Dále je nutno vytvořit takzvané *credentials* pro získání přístupu do projektu. Po vytvoření *credentials* se vygeneruje *Client ID*, které je nutno stáhnout ve formátu *JSON* a nahrát do adresáře aplikace. Tento *JSON* je potom načítán při přihlašování uživatele ke svému disku. Po přihlášení je vytvořen takzvaný *token*, který umožňuje uživateli opětovnou synchronizaci bez ověření. *Token* je platný dokud se uživatel neodhlásí a nebo vypne aplikaci.

- *init* – konstruktor sloužící pro navázání spojení s uživatelským účtem a vytvoření adresáře.
- *getID* – vrátí *ID* požadovaného adresáře podle názvu.
- *createFolder* – vytvoří adresář na uživateli Google disku.
- *uploadDB* – nahraje databázi do daného adresáře na Google disku. Metoda je na ukázce 4.9
- *downloadDB* – stáhne databázi z úložiště a uloží ji do adresáře aplikace.

```
1     def uploadDB(self):
2         """
3         Nahraje do adresare BP_app databazi typu sqlite
4         """
5         folder_id = self.getID("BP_app")
6         id = self.getID("database.sqlite")
7         if(id is not None):
8             self.service.files().delete(fileId=self.getID("database.
    sqlite")).execute()
9         file_metadata = {
10            'name' : "database.sqlite",
11            'parents' : [folder_id],
12        }
13        media = MediaFileUpload("./DB/database.sqlite",mimetype = "
    application/x-sqlite3")
14        self.service.files().create(
15            body = file_metadata,
16            media_body = media,
17            fields = 'id'
18        ).execute()
```

Ukázka 4.9: Metoda pro nahrání databáze na Google disk

4.3 Odesílání E-mailů

Druhé využití *Google API* v aplikaci je hromadné odesílání e-mailů. Způsob přihlášení je stejný jako u synchronizace databáze. *Token* pro synchronizaci a e-maily je propojen.

Pokud uživatel vloží text zprávy do *QTextEditu* a klikne na odeslat. Aplikace zjistí všechny e-maily závodníků a pošle jim e-mail. Uživateli je také nabídnuta možnost přiložit ke zprávě soubor.

- *init* – konstruktor sloužící pro navázání spojení s uživatelovým účtem a vytvoření adresáře.
- *createMessage* – vytvoří zprávu, přiloží soubor ke zprávě, pokud je k dispozici.
- *sendMessage* – odešle zprávu.
- *getUserEmail* – vrací uživatelův e-mail
- *getEmails* – vrátí všechny dostupné e-maily pro daný závod.

Závěr

Cílem práce bylo vyvinout desktopovou aplikaci schopnou tvorby a následné správy závodů či seriálů. Použití aplikace není závislé na připojení k internetu. Připojení pouze nabízí synchronizaci přes uživatelův Google disk pro jednodušší přesun databáze nebo odesílání e-mailů přes Gmail.

Na základě rešerše byl pro realizaci využit programovací jazyk *Python* propojený s databázovým systémem *SQLite*. Pomocí *HTML* aplikace nabízí export vybraných dat a zároveň možnost tisku. Za celým *GUI* stojí *toolkit PyQt* verze 5 se stylizací pomocí *QSS*. Pro připojení ke službám firmy Google bylo využito *Google API*. Návrh a úprava prvních modelů *GUI* probíhala v programu Qt Designer. Pro tvorbu databáze jsem využil program od firmy Oracle MySQL WorkBench a následně upravil na standardy *SQLite*. Celá aplikace se sama přizpůsobuje rozlišení monitoru, na kterém je spuštěna. Aplikace je tedy plně responzivní.

Všechny, už dříve, zmíněné požadavky byly implementovány podle pokynů a aplikace nabízí i několik funkcí navíc. Čistě pro ulehčení práce organizátora. Systém je schopen samostatně vyhodnotit výsledky pouze podle času dojezdu závodníka a jeho stavu při dojezdu. Všechna vyhodnocená data aplikací je možno exportovat na papír a nebo do formátu *PDF*. U každého exportu je dále možnost náhledu na výsledek.

Minimální velikost aplikace je *1000x600* pixelů. Při zvyšování velikosti rozlišení je aplikace automaticky responzivní a přizpůsobí se tak na displeje větší velikosti. Minimální velikost aplikace je určena kvůli množství zobrazovaných dat.

K aplikaci je dodána dokumentace celého kódu pomocí modulu *Pydoc* [18], který zajišťuje vygenerování dokumentace přímo ze souboru, v programovacím jazyce *Python*, do *HTML* kódu. Pomocí *Pydoc* si také uživatel může vygenerovat nápovědu související se souborem a nebo přímo s metodou některé třídy.

Uživateli je také nabídnuta možnost zobrazení aplikace pomocí takzvaného *wireframe modelu*. Jedná se o jednoduché a rychlé znázornění konceptu aplikace. Základní myšlenkou modelu je ukázat uživatelovi nejjednodušší možnou stránku aplikace, kvůli jejímu maximálnímu pochopení. Uživatel se na model může podívat v jednotlivých *PDF* souborech a nebo v krátkém videu, které se nachází na přiloženém CD

Aplikace má zatím několik nedostatků, které se především týkají jednoduchosti ovládání. Například v sekci časomíra by pro vložení stavu dojezdu v tabulce mohlo být menu s možnostmi. Tato funkce, ale při vyšším počtu závodníků zpomaluje načtení sekce. Proto jsem tuto možnost zatím neimplementoval.

Pro budoucí vývoj bych uvedl možnost propojení aplikace s platbami startovného. Uživateli by tak aplikace evidovala počet zaplacených a nezaplacených závodníků. Musela by se tak implementovat úplně nová sekce do správy závodu a to sekce platby. K tomu by se musela přidat funkcionalita odesílání emailů závodníkům s potvrzení o platbě.

S tímto systémem by se aplikace pomalu stávala nejen aplikací situovanou převážně na místo konání akce, ale byla schopna celkové organizace závodu. Popřípadě se tu naskytuje možnost propojení aplikace přímo s webovým serverem řešící online registrace na webu daného závodu.

Jako další rozšíření se tu nabízí odesílání SMS závodníkům s výsledky ihned po závodě. Pro tuto funkci *Python* nabízí několik skvěle zdokumentovaných *API*. Avšak všechny tyto brány, které tyto služby nabízí, jsou zpoplatněny, ať už na měsíční paušál a nebo na počet SMS. Kvůli tomuto problému jsem se rozhodl neimplementovat toto *API* do mé aplikace.

Jako ulehčení jsem chtěl nabídnout organizátorům možnost spustit aplikaci na systému *Android*. Toto ulehčení by se vyplatilo například při hlášení startovní listiny moderátorem. Nemusela by se stále tisknout nová listina, pokud stále probíhá registrace. Moderátor by jednoduše měl v ruce tablet s aplikací a vždy by jen synchronizoval databázi a měl obnovenou tak i startovní listinu. Tento export aplikace na systém *Android* se mi ale nepodařilo uskutečnit. Po exportu a následnému spuštění aplikace selže bez jakékoliv chybové hlášky. Export jsem prováděl přes *buildozer* [4] na systému *Linux*. Zatím jsem nepřišel na úspěšnější řešení.

Tato aplikace nebyla vytvořena pouze za účelem splnění bakalářské práce, ale také za účelem ulehčit organizátorům v mém okolí práci při organizování sportovních akcí. Ze zpětné vazby od organizátorů akcí jsem se dozvěděl, že aplikace splnila účel a ulehčuje tak organizaci menších závodů. Aplikaci zatím využili dva organizátoři v mém okolí, avšak nabízím ji už i dalším. Kvůli zrušení závodů, ve smyslu zamezení pandemické situace, vše nabralo neočekávané zpoždění.

Při plnění práce jsem se zdokonalil v mnoha odvětvích. Od programování v jazyce *Python* po tvorbu databáze pomocí *SQLite*. Několik užitečných zkušeností jsem také získal při psaní této práce. Největší zkušeností je však fakt, že jsem si zkusil vytvořit obsáhlejší práci. V průběhu několika měsíců, kdy jsem tvořil tuto práci, jsem se potkal s několika problémy a kdybych na začátku práce lépe navrhl databázi nebo funkčnost aplikace, ušetřil bych mnoho času.

Literatura

[1] About SQLite. *SQLite* [online]. 2020 [cit. 2021-2-26]. Dostupné z: https://www.sqlite.org/about.html

[2] *Adobe* [online]. Adobe, 2021 [cit. 2021-6-7]. Dostupné z: https://www.adobe.com/cz/products/xd.html

[3] BEZVODOVA, Sara. Popis prvků konceptuálních modelů, smysl konceptuálního modelu a příklad modelu Knihovny. *WikiSofia* [online]. Praha, 2013 [cit. 2020-1-25]. Dostupné z: https://wikisofia.cz/wiki/Popis_prvk%C5%AF_konceptu%C3%A1ln%C3%ADch_model%C5%AF,_smysl_konceptu%C3%A1ln%C3%ADho_modelu_a_p%C5%99%C3%ADklad_modelu_Knihovny

[4] *Buildozer-s documentation* [online]. Kivy's Developers Revision, 2014 [cit. 2021-6-5]. Dostupné z: https://buildozer.readthedocs.io

[5] *C++* [online]. cplusplus.com, 2000 [cit. 2021-6-7]. Dostupné z: https://www.cplusplus.com/

[6] *Gmail for Developers* [online]. Google, 2021 [cit. 2021-6-8]. Dostupné z: https://developers.google.com/gmail/api/guides

[7] Google APIs. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2021 [cit. 2021-5-25]. Dostupné z: https://en.wikipedia.org/wiki/Google_APIs

[8] *Google Docs for Developers* [online]. Google [cit. 2021-6-5]. Dostupné z: https://developers.google.com/docs/api

[9] *Google Drive for Developers* [online]. Google, 2020 [cit. 2021-6-8]. Dostupné z: https://developers.google.com/drive/api/v3/about-sdk

[10] HENLEY, A.J a D. WOLF. *Learn Data Analysis with Python.* New York: Apress Media, 2018. ISBN 978-1484234853

[11] How To Use Qss In Qt Project. *QSS stock* [online]. Montpellier Centre: DevSec Studio, 2020 [cit. 2021-1-8]. Dostupné z: https://qss-stock.devsecstudio.com/documentation.php

[12] HTML Tutorial. *W3Schools* [online]. Norway: Refsnes Data, 2016, 8-9-2016 [cit. 2021-2-26]. Dostupné z: https://www.w3schools.com/html/

[13] *Java portál* [online]. Praha: MathAn Praha, 2002 [cit. 2021-6-7]. Dostupné z: http://www.java.cz/

[14] *Jetbrains* [online]. Praha: JetBrains s.r.o, 2000 [cit. 2021-6-7]. Dostupné z: https://www.jetbrains.com/company/

[15] *MySQL* [online]. Oracle Corporation, 2020 [cit. 2021-6-7]. Dostupné z: https://www.mysql.com/products/workbench/

[16] PILGRIM, M. *Ponořme se do Python(u) 3.* 3. vydání. Praha: CZ.NIC, 2010. ISBN 978-80-904248-2-1.

[17] POKORNÝ. Konceptuální model databáze. *KTD - Česká terminologická databáze knihovnictví a informační vědy* [online]. Praha: ExLibris, 2012 [cit. 2021-1-12]. Dostupné z: http://aleph.nkp.cz/publ/ktd/00000/01/000000110.htm

[18] *Pydoc: Documentation generator and online help system* [online]. Python Software Foundation, 2001 [cit. 2021-6-5]. Dostupné z: https://docs.python.org/3/library/pydoc.html

[19] *Python 3.9.5 documentation* [online]. Python Software Foundation, 2001 [cit. 2021-6-5]. Dostupné z: https://docs.python.org/3/

[20] *Pyuic5 Tool* [online]. Python Software Foundation, 2021 [cit. 2021-6-5]. Dostupné z: https://pypi.org/project/pyuic5-tool/

[21] Qt Designer Manual. *Qt Documentation* [online]. Espoo, Finsko: The Qt Company, 2008, 2014 [cit. 2021-1-12]. Dostupné z: https://doc.qt.io/qt-5/qtdesigner-manual.html

[22] *Red Hat* [online]. Red Hat, 2021 [cit. 2021-6-8]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[23] *SQLcourse* [online]. TechnologyAdvice [cit. 2021-6-7]. Dostupné z: http://www.sqlcourse.com/intro.html

[24] SUMMERFIELD, M. *Python Výukový kurz 3.* Brno: Computer Press Brno, 2013. ISBN 978-80-251-2737-7.

[25] *UML diagrams* [online]. uml-diagrams.org., 2009 [cit. 2021-6-7]. Dostupné z: https://www.uml-diagrams.org/

[26] *Visual paradigm* [online]. Visual Paradigm, 2020 [cit. 2021-6-7]. Dostupné z: https://www.visual-paradigm.com/

[27] What is PyQt? *Riverbank Computing* [online]. Dorcheste, 2020 [cit. 2021-2-26]. Dostupné z: https://www.riverbankcomputing.com/software/pyqt/intro

Příloha A

Základní ovládání aplikace

V této kapitole jsou stručně popsány pokyny pro všechny základní funkce aplikace. Uživatel by po přečtení kapitoly měl být schopen ovládat aplikaci se všemi jejími přednostmi.

Pro větší představu o aplikaci slouží video přiložené na CD. Video je ukázkou průchodu aplikací pomocí *wireframe modelu*.

A.1 Postup instalace

Samotný postup, jak spustit moji aplikaci je velice jednoduchý. Celá aplikace je *portable*[1]. Tedy stačí mít k dispozici adresář s aplikací a pouze spustit spustitelný soubor. Zda je adresář na vašem zařízení nebo na externím nehraje žádnou roli.

V adresáři s programem je připojené *readme*[2], které uvádí, jak program spustit i pro uživatele, který nečetl práci.

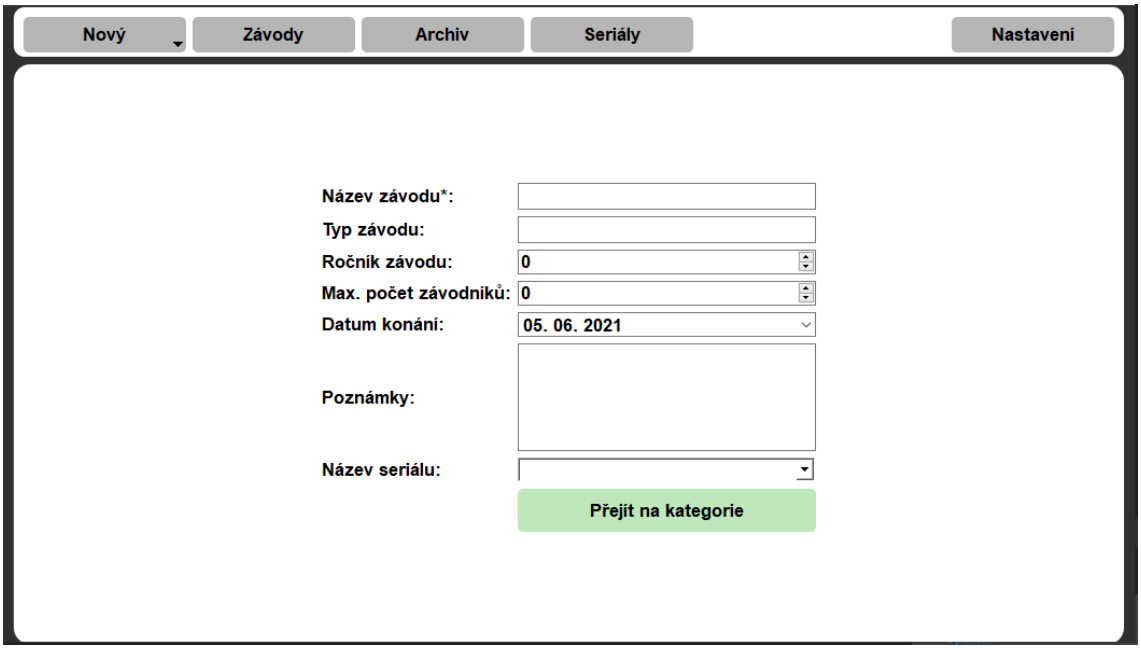
A.2 Vytvoření závodu

Po otevření okna aplikace přejdeme na tlačítko *Nový*. Po vysunutí podmenu klikneme na závod. Po vygenerování obsahu (viz obr. A.1) vyplníme požadované textové pole název závodu a klikneme na tlačítko *Přejít na kategorie*.

Aplikace nás přesměruje do tvorby kategorií. Vyplníme název kategorie a rozmezí ročníků, pokud kategorie není určena pro přiřazování. Pokud u kategorie není potřeba zadávat pohlaví tak klikneme na tlačítko *Přidat* a pokračujeme s dalšími kategoriemi.

[1] Lze spustit z přenosného zařízení

[2] Textový soubor se základními informacemi



Obrázek A.1: Tvorba závodu

A.3 Vytvoření seriálu

Postupujeme stejně jako u tvorby závodu. Klikneme na tlačítko *Nový* a *Seriál*. Vyplníme všechny textové pole a klikneme na tlačítko *Potvrdit*. Obrázek pro lepší představu nalezneme zde A.2

A.4 Zobrazení startovní listiny

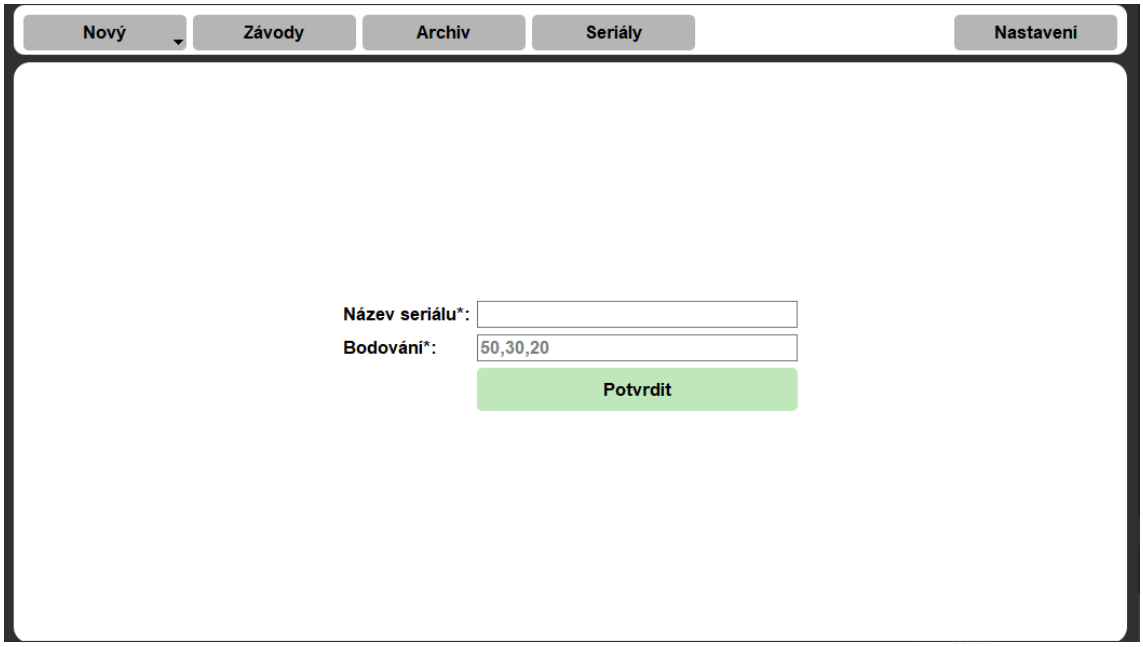
První krok je vybrat si závod u kterého chceme zobrazit startovní listinu. Z menu vybereme *Závody* a klikneme na vybraný závod. Tento krok nás přenesl do správy závodu. V novém menu se přeneseme na tlačítko *Prezentace* a dále na *Startovní listina* (viz obr. A.3).

Tento krok se dá zrychlit kliknutím pravým tlačítkem na závod a následným kliknutím na *Startovní listina* už v zobrazení závodů.

A.5 Zobrazení výsledků

Vybereme si závod, jako u zobrazení startovní listiny. Přejdeme na tlačítko *Výsledky* (viz obr. A.4). Pro vybrání jednotlivých kategorií klikneme na rolovací menu.

Tento krok se dá zrychlit kliknutím pravým tlačítkem na závod a následným kliknutím na *Výsledky* už v zobrazení závodů.



Obrázek A.2: Tvorba seriálu

A.6 Import dat

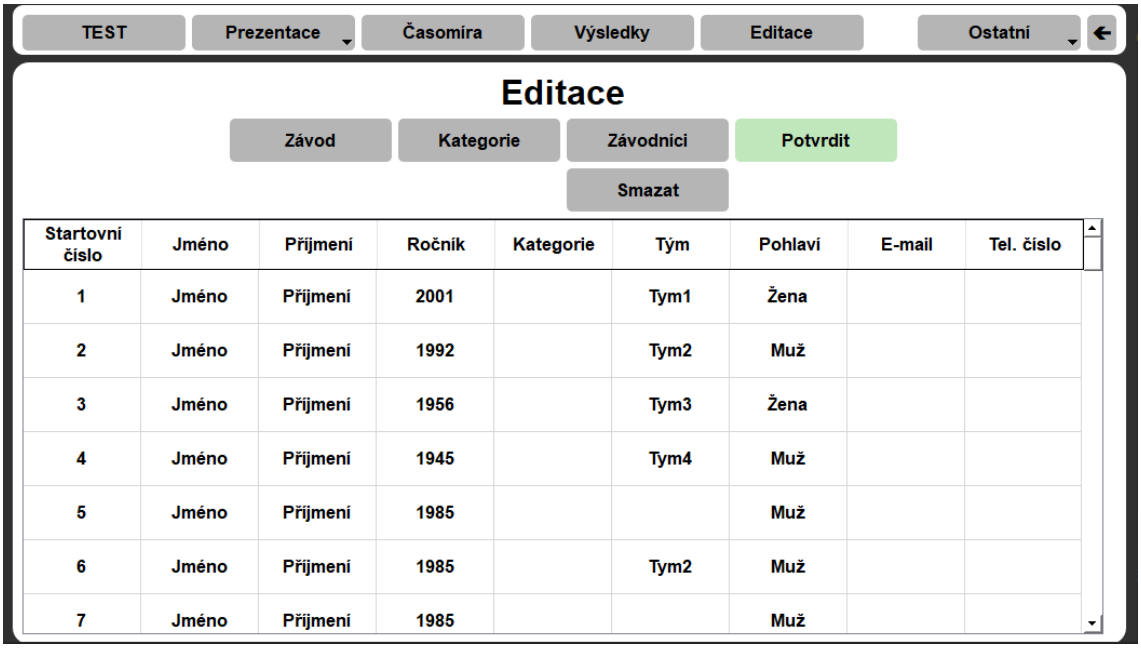
Po vybrání závodu, jako v předešlých pokynech klikneme znovu na tlačítko *Prezentace* a vyberem *Import*. Přejdeme na tlačítko *Vybrat soubor* a vybereme požadovaný soubor ve formátu *CSV, XLSX, ODS*. Po zobrazení dat začneme s výběrem. Klikneme na tlačítko pro jméno a přiřadíme sloupec dat, který danému záznamu odpovídá. Po vybrání všech požadovaných dat klikneme na tlačítko *Import*

A.7 Zaznamenání času

Ve správě závodu vybereme tlačítko *Časomíra* (viz obr. A.5). Čas můžeme přiřadit dvěma způsoby. Buď vyplníme startovní číslo, čas dojezdu, stav a klikneme na tlačítko *Zapsat*. Druhým způsobem je vepsat čas přímo do zobrazené tabulky do sloupce čas.

A.8 Editace dat

Po kliknutí na závod přejdeme na tlačítko *Editace* a vybereme dále sekci, kterou chceme upravovat. Na výběr máme z sekcí závod, kategorie, závodníci. Pro vybrání sekce klikneme na dané tlačítko. Vypíší se nám data o dané sekci. Pro editaci pouze klikneme na požadovanou buňku a změníme hodnotu. Vše se potvrdí tlačítkem *Potvrdit*. Obrázek s editací můžeme vidět na obrázku A.6.



Obrázek A.3: Startovní listina

U sekce kategorie můžeme přidat kategorii tlačítkem. Objeví se tak nový řádek a my můžeme zadat patřičné informace o nové kategorii. Nebo ji druhým tlačítkem smazat.

K přiřazení kategorie závodníkovi a nebo přiřazení seriálu k závodu je nutné, aby dané záznamy existovaly.

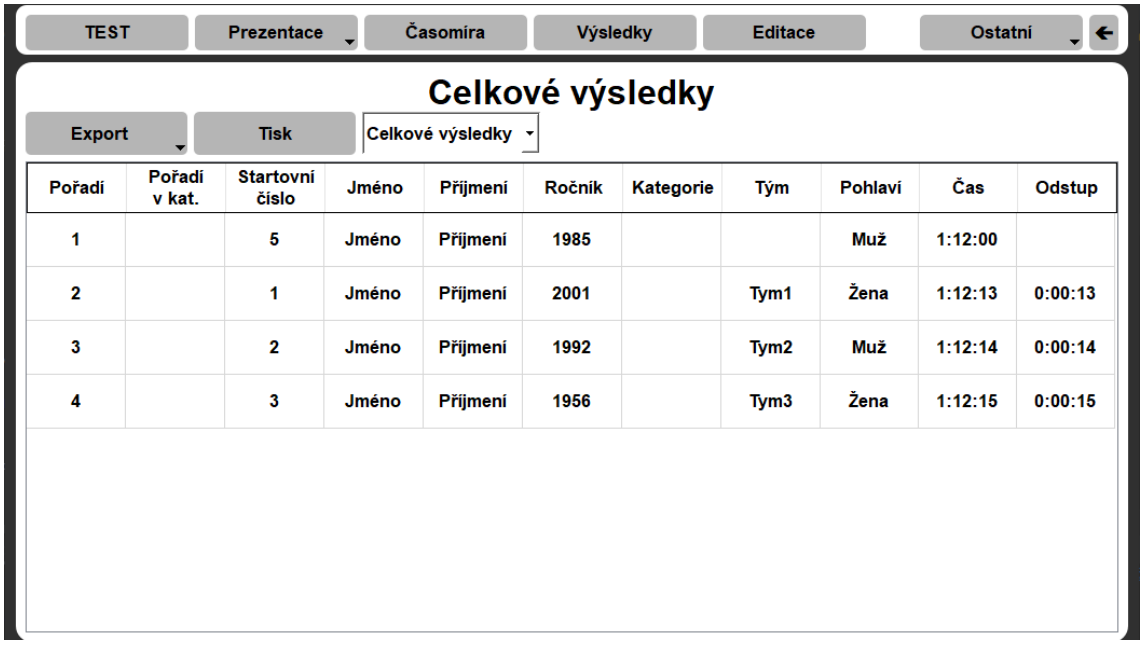
A.9 Tisk diplomů

Ve správě závodu klikneme na tlačítko *Ostatní* a ve výsuvném menu klikneme na *Tisk diplomu*. Když klikneme na jakékoliv šedé tlačítko otevře se nám dialogové okno pro výběr šablony na diplom. Šablona musí být formátu *PNG* nebo *JPG*. Po vybrání šablony určíme pozici hodnoty. Je nutné vybrat oblast z levé do pravé strany. Pro potvrzení vybereme tlačítko *Export*

Zaškrtnutím políčka *Jednotlivé* určíme, že aplikace vygeneruje diplom pro každého závodníka zvlášť do formátu *PDF*. Hotový diplom z této šablony může vypadat jako na obrázku A.7

A.10 Odeslání hromadných emailů

Ve stejném menu, jako se nachází tisk diplomů vybereme sekci *Hromadné E-maily*. Do velkého textového pole vypíšeme naši zprávu pro závodníky. Pokud chceme ke zprávě



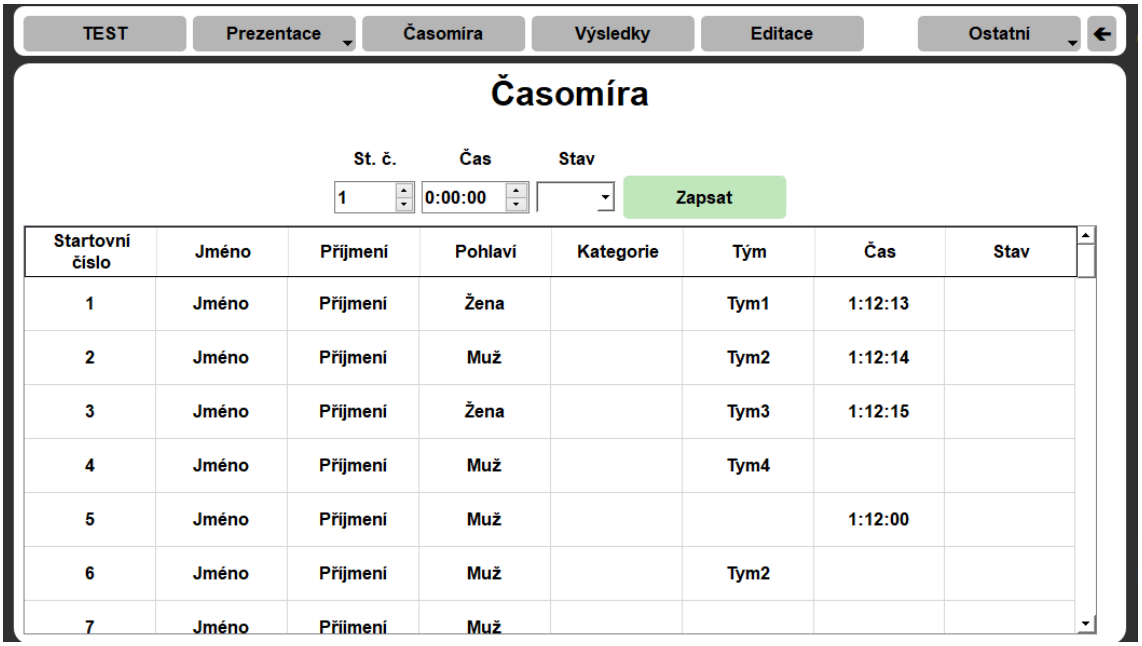
Obrázek A.4: Výsledky

připojit soubor, musíme ho vybrat pomocí tlačítka *Připojit soubor*. E-mail odešleme tlačítkem *Odeslat*. E-mail se odešle všem závodníkům s vyplněným e-mailem.

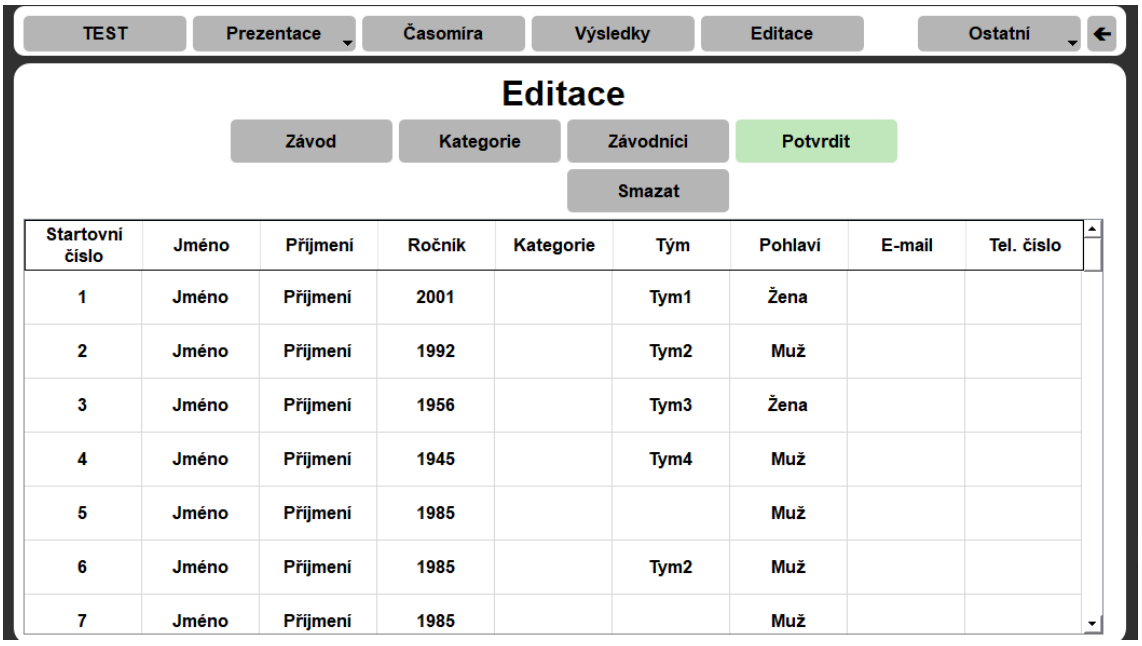
A.11 Synchronizace databáze

Pro synchronizaci databáze přejdeme na sekce *Nastavení* po otevření okna aplikace. Pro nahrání databáze na Google disk stiskneme *Upload* a naopak pro stažení nejnovější databáze stiskneme *Download*. Stisknutí na jedno z uvedených tlačítek nám otevře internetový prohlížeč pro autorizaci. Do zavření aplikace není nutné se znovu přihlašovat, pokud neklikneme na tlačítko *Odhlásit se*.

Můžeme zde také vymazat funkci našeptávač a nebo smazat celý obsah databáze.



Obrázek A.5: Časomíra



Obrázek A.6: Editace

Obrázek A.7: Ukázka diplomu

Příloha B

Testování

B.1 Běh do vrchu – Jedlová

První otestování aplikace proběhlo na závodě v Jiřetíně pod Jedlovou 18. 4. 2021. Jednalo se o běžecký závod s omezením, který byl omezen pandemickou situací. Start závodníků byl tedy po vlnách a měřený čas byl vypočítán z času doběhnutí do cíle a času startu. Výsledky byly, kvůli tomuto omezení, vloženy do aplikace až zpětně a vyhlašování výsledků proběhlo online.

Aplikace byla nejdříve využita na místě registrace pro zaznamenávání závodníků. Každý závodník zapsal požadované údaje na papír a následně přišel k registraci, kde byl zaregistrován do závodu. Jedná se o běžný postup.

Při využívání aplikace se objevily chyby:

1. Chyba při registraci závodníka, který měl v e-mailové adrese „-“. Tento znak mi validace, pomocí regulárních výrazů, nepropustila.
2. Startovní listina závodníků nebyla řazena podle startovního čísla.
3. Následný export výsledků do *PDF* nepočítal správně časové odstupy od vítěze.

Organizátor závodu, pan Karel Valenta, se vyjádřil k aplikaci takto:

„Pořadatelé menších závodů postrádají nenákladnou a na obsluhu jednoduchou správu startujících i následné zpracování výsledků. Na zkoušené aplikaci od pana Zahradníka oceňuji přehlednost, i bez manuálu je jasné co se stane po stisknutí nějakého tlačítka, v aplikaci jsem „nezabloudil.“ Co vidím jako další přínos je velikost fontů a tlačítek. V praxi je to velkým ulehčením, neboť se při prezentacích většinou používají „cizí stroje“, kde například citlivost touchpadu, bývá noční můrou pořadatelů. Ze svého pohledu bych ocenil možnost potvrdit souhlas s pravidly závodu a GDPR při zadávání údajů do startovní listiny. Dále pak při generování výsledků záhlaví a zápatí (název závodu, datum, zpracovatele výsledků, stránkování). Věřím ale, že další rozšiřování a cílené úpravy aplikaci neminou. I teď je ale zdatným pomocníkem pro pořadatele.“

B.2 Závod horských kol v Podluží

Druhé testování proběhlo na cyklistickém závodu horských kol dne 22. 5. 2021. Jednalo se o menší závod s třiceti závodníky, tedy závod na který je má aplikace přesně určena. Ta byla využita na místě prezentace k registraci závodníků a pak k zápisu výsledků přímo v cíli. Závod už měl hromadný start, takže jsem vyzkoušel i rychlý zápis časů dojezdu přímo na místě.

Po dojetí posledního závodníka jsem nechal zobrazit výsledky jednotlivých kategorií, které bez problému aplikace vyhodnotila a předal organizátorovi. Žádná chyba z pozice funkčnosti už nebyla nalezena. Vyhlášení proběhlo ihned po dojetí a závod byl ukončen. Výsledky následně byly exportovány a předány organizátorovi ještě ve formátu *PDF*.

Organizátor závodu, pan David Stárek, se vyjádřil k aplikaci takto:

„Snaha Vojty pomoct nám „menším“ organizátorům se mi velice zalíbila. Atmosféru menších závodu si velice užívám a když ještě nemusím řešit výsledky a bát se, že nebudou, je to perfektní. Dost často se totiž stává, že pomocníci nestihnou zapsat na papír všechny výsledky, které potom chybí. Zpětné reklamace kvůli špatným výsledkům se také stávají. Pokud to vše řeší aplikace, tak je vše bez starosti a můžu si v klidu zazávodit.“

Příloha C

Obsah CD

V kořenovém adresáři CD nalezneme:

- **BP_Zahradník.pdf** – text bakalářské práce.
- **Aplikace** – zde je uložena spustitelná verze aplikace. Pro spuštění je nutno spustit soubor *aplikace.exe*.
- **Dokumentace** – dokumentace aplikace ve formátu *HTML*.
- **Kód** – obsahuje soubory s kódem aplikace v programovacím jazyce *Python*.
- **Modely, diagramy** – zde jsou uloženy modely, jako *wireframe model* nebo také *E-R diagram* databáze, *Use case diagram* nebo i *Class diagram*.