VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ŘÍZENÍ SPORTOVNÍCH AKCÍ

SPORTS EVENT MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ALEŠ TETUR**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Tetur Aleš**
Program: Informační technologie
Název: **Systém pro řízení sportovních akcí**
Sports Event Management System
Kategorie: Informační systémy

Zadání:

1. Prostudujte současné technologie pro vývoj webových klient-server aplikací.
2. Seznamte se s požadavky na systém pro řízení sportovních akcí se zaměřením na šplh na laně.
3. Na základě analýzy požadavků navrhněte architekturu systému. Uvažujte možnost online sledování průběhu závodu, integraci externí elektronické časomíry, zpracování výsledků, tisk diplomů a další funkce na základě konzultací s vedoucím.
4. Implementujte navržený systém pomocí vhodných technologií.
5. Proveďte testování systému v reálném nasazení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 23. října 2020

Abstrakt

Cílem práce bylo vytvořit informační systém s integrovanou elektronickou časomírou pro akce Šplh na laně. U informačního systému jsem využil několik technologií pro tvorbu pokročilých webových systémů (HTML, CSS, PHP, JS, jQuery a MySql). Pro elektronickou časomíru jsem použil volně dostupnou desku Raspberry Pi 4. Na časomíře je dále server Appache 2 a prohlížeč Chromium pro zobrazení grafického rozhraní. Integrace probíhá pomocí komunikace časomíry a serveru informačního systému. Dále časomíra disponuje dvěma čidly. A to spodním a horním čidlem. U obou jsem pro komunikaci s časomírou použil desku ESP8266, která je zasazená do mnou navrženého plošného spoje.

Abstract

The main goal of this thesis was to create an information system with an integrated electronic timer for the events Rope climbing. For the information system, I used several technologies to create advanced web systems (HTML, CSS, PHP, JS, jQuery and MySql). I used a freely available Raspberry Pi 4 board for the electronic timer. The integration takes place using communication between the timer and the information system server. The timer also has two sensors. And that is the lower sensor and the upper sensor. For both, I used the ESP8266 board for communication with the timer, which is embedded in my designed printed circuit board.

Klíčová slova

bakalářská práce, informační systémy, webové stránky, elektronická časomíra, integrovaná časomíra do systému, systém pro šplh na laně, apache 2, GUI, 3D tisk, raspberry pi

Keywords

bachelor's thesis, information system, web page, electronic timer, integrated timer in the system, system for šplh na laně, apache 2, GUI, 3D print, raspberry pi

Citace

TETUR, Aleš. *Systém pro řízení sportovních akcí*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Systém pro řízení sportovních akcí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Aleš Tetur
5. května 2021

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Radkovi Burgetovi, Ph.D. za jeho odborný dohled nad prací a za konzultace, které mi poskytl během vypracovávání této práce. Také bych chtěl poděkovat starostovi Sokola Brno I, panu Ing. Martinu Vlkovi, za poskytnutí možnosti otestování elektronické časomíry v tělocvičně.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

U sportovních akcí Šplh na laně pořadatelé využívají zastaralý systém pro jejich správu. Elektronická časomíra také není aktuální, jelikož neumožňuje zobrazování stopovaného času na zobrazovací zařízení a nemá ani připojení pro audio zařízení. Tyto vlastnosti jsou v dnešní době pro rostoucí základnu tohoto sportu klíčové.

Cílem práce je navrhnout a vytvořit informační systém spolu s integrovanou elektronickou časomírou pro sportovní akce Šplhu na laně. První částí je informační systém, který by měl vycházet z aktuálních požadavků pořadatelů i závodníků. Také by měl co nejvíce zjednodušit pořádání těchto závodů a sjednotit jej. Druhou částí je integrovaná elektronická časomíra, která by měla mít výstup pro připojení monitoru pro zobrazování času a audio výstup pro připojení reproduktorů. Elektronická časomíra by též měla komunikovat s informačním systémem a vkládat do něj dosažené časy závodníků pro co nejefektivnější automatizaci.

Téma bakalářské práce jsem si zvolil, protože jsem jedním z pořadatelů velkých závodů Šplhu na laně a již delší dobu mě iritoval neefektivní systém, který se doposud využíval. Po dohodě s hlavními organizátory jsem se rozhodl vytvořit systém podle současných požadavků.

Práce je rozdělená do pěti částí. První kapitola je zaměřena na dosavadní řešení systému a časomíry spolu s požadavky na nové řešení. Je v ní popsaný sport Šplh na laně a vše s ním spojené. Druhá část práce se zaměřuje na klíčové technologie, které jsem pro tvorbu jak informačního systému, tak pro elektronickou časomíru nastudoval a použil pro jejich vytvoření. Třetí kapitola popisuje podrobný a postupný návrh systému i časomíry. U systému bylo potřeba nejdříve navrhnout přístup vyplývající z požadavků a také funkce, které systém musí obsahovat. Další fází návrhu bylo vytvoření modelů, ze kterých systém vychází. Dále bylo potřeba navrhnout vzhled systému i časomíry. Další část řeší návrh hardwaru časomíry a 3D modelům všech dílů pro časomíru a čidla, která obsahuje. Předposlední kapitola se věnuje konkrétní implementaci a řešení informačního systému a elektronické časomíry. Je zde popsaný způsob implementace čidel časomíry. Poslední kapitola je zaměřena na testování a zhodnocení dosažených výsledků. Největší část testování se zabývá funkčností navrženého hardwaru čidel, odezvy systému časomíry a testování přesnosti měření času. Poslední část testování věnuji informačnímu systému. Zhodnocení shrnuje dosažené výsledky testování a hodnotí celkové řešení.

Kapitola 2

Dosavadní řešení a požadavky

V této kapitole se věnuji dosavadnímu řešení informačního systému a časomíry pro akce Šplh na laně a požadavky na nový systém vyplývající z tohoto řešení.

2.1 Dosavadní řešení

Níže se podrobněji zabývám popisem doposud používaného řešení informačního systému a elektronické časomíry.

2.1.1 Šplh na laně

Ze všeho nejdříve je potřeba přiblížit, co je Šplh na laně. Tento sport sice není příliš známý, má však dlouhou historii, během níž byl i součástí Letních olympijských her. Dnes se v tomto sportu, v každé soutěžní sezóně, koná zhruba 8 Velkých cen. Tato sezóna je ukončena Mistrovstvím ČR. Na každou akci se musí závodníci registrovat. Závody probíhají pravidelně, veřejně a vždy s účastí publika. Zavedená pravidla stanovují způsob šplhu i délku lana, pro každou kategorii. Na laně je dovoleno šplhat pouze bez pomoci nohou a bez přírazu, tzn. závodník startuje ze sedu, ve kterém se nesmí země dotýkat jiná část těla než hýždě a ruka, která mačká startovací spínač elektronické časomíry. Na konci lana, v pravidly určené výšce, je čidlo snímající dokončení šplhu.

2.1.2 Informační systém

Stávající řešení, obsahuje informační systém skládající se z několika dílčích částí. První z nich je registrace závodníku na jednotlivé závody. Doposud se stále používá registrace pomocí e-mailu. Jakmile se závodník dozví o novém závodu, kterého by se chtěl zúčastnit, musí poslat přihlášku se svými údaji pořadateli závodu. Tento způsob opět dává velký prostor pro překlepy a chyby z nepozornosti, které je poté nutné opravit. Další velkou nevýhodou je časová náročnost přepisování informací o závodnících pro pořadatele i pro závodníka.

Další a stěžejní částí je MS Excel. Do něj se zapisují závodníci podle kategorie, která jim přísluší. Každá kategorie má vlastní list v sešitu. Ke každému závodníkovi se postupně ručně dopisují jejich dosažené výsledky v závodě. Tento proces je zdlouhavý a dává prostor pro vytvoření chyby z nepozornosti. Výsledky se po ukončení závodu zpracovávají také v MS Excelu pomocí funkcí, které tento nástroj nabízí.

Po vytvoření kompletních výsledků, je zapotřebí vytisknout diplomy. U všech závodů jsou diplomy předtištěny jako prázdná šablona vytvořená grafiky. U většiny závodů se tyto šablony diplomů obsahující jméno, pořadí i nejlepší dosažený čas závodníka dopisují ručně. To je velmi zdlouhavý a náročný proces. U zbývajících závodů se na tisk diplomů využívá další nástroj od Microsoftu, a to MS Word. Konkrétně funkce hromadné korespondence. Ta se spojí s výsledky uloženými v MS Excelu a vygenerují se diplomy. Úskalí s hromadnou korespondencí je v zapozicování polí s požadovanými informacemi. Tím, že diplomy jsou předtištěné šablony, musí se metodou pokus omyl posouvat pole s informacemi, dokud není výsledek uspokojivý. Tento způsob sice ušetří náročnost práce s ručním psaním, ale vyžaduje znalost MS Wordu a jeho hromadné korespondence.

Poslední součástí systému je distribuce výsledků proběhlého závodu. Během závodění není možnost sledovat průběžné výsledky. Až po každém dokončeném kole se vytiskne listina s dosaženými výsledky. Po zakončení závodu se vytiskne několik kompletních výsledkových listin pro nahlédnutí. Po oficiálním zpracování se výsledky závodníkům pošlou e-mailem a také se vyvěsí na webových stránkách Svět šplhu. Ovšem zde je zase prodleva se zveřejněním výsledků v důsledku administrativy této stránky. Výsledky jsou tedy zveřejněny se značnou prodlevou.

2.1.3 Časomíra

V této části se budu věnovat elektronické časomíře, která se nyní používá na všech oficiálních závodech.

Používaná elektronická časomíra je stará zhruba 15 let a je převážně analogová. Časomíra není se svými rozměry příliš skladná. Je realizována jako kufr o rozměrech 46cm krát 33,5cm krát 15,5cm, viz. obrázek níže.

Obrázek 2.1: **Elektronická časomíra**

Časomíra je rozdělena na tři části. První částí je časomíra jako taková. Obsahuje několik obvodů, dva červené segmentové displeje zobrazující časy, tři mechanická tlačítka na ovládání časomíry, bzučák pro startovací signalizaci a konektory pro připojení periferií a zdroje. Tato část zpracovává signály z periferií dále "čidel"a z ovládacích tlačítek. Jediným vizualizačním výstupem jsou již zmíněné displeje. Také bzučák je jediným audio výstupem.

Tento bzučák při startu vždy třikrát pípne. První dvě pípnutí jsou krátká. Poslední pípnutí je buď krátké nebo dlouhé, a to v závislosti na startu závodníka. Pokud závodník pustí spodní spínač dříve, než určují pravidla, dopustí se tzv. předčasného startu, třetí pípnutí je nepřerušované a dlouhé. Pokud ovšem závodník pustí spodní čidlo v souladu s pravidly, je poslední pípnutí totožné s prvními dvěma. Bzučák je součástí obvodů časomíry a není zde žádná možnost, jak jej připojit k reproduktorům, což je v mnoha situacích potřeba.

Ovládání časomíry probíhá, jak jsem napsal již výše, pomocí tří mechanických tlačítek. Jedno tlačítko má funkci resetu a zbylá dvě tlačítka mají totožnou funkci jako spodní čidla.

Druhou a třetí částí elektronické časomíry jsou čidla. Konkrétně spodní a horní čidlo. Funkce spodního čidla je snímání, zda závodník neodstartoval předčasně. To je zajištěno tak, že závodník nesmí podle pravidel uvolnit spodní čidlo dříve, než zazní poslední třetí pípnutí. Tento spínač je řešen pomocí klasického přepínače na světla se zpětnou pružinou. Velkým problémem je u tohoto spínače jeho spolehlivost. Jelikož se do kontaktů dostává práškové magnesium, kterým si závodníci před šplhem pomažou dlaně. Horní čidlo je řešené pomocí cívky namotané na válci, který se nasune a přišroubuje na lano. Zde se měří některá fyzikální veličina. Obě čidla jsou připojena k časomíře pomocí kabelů. Toto připojení má výhodu minimální latence, ale je zde problematická montáž, a to zejména na osmimetrovém laně. Také rozvod těchto kabelů je náročný protože, nesmí nijak zavazet závodníkům při šplhu.

2.2 Požadavky

V této kapitole se věnuji požadavkům na nový informační systém s integrovanou externí elektronickou časomírou vyplývající z neuspokojivého a zastaralého dosavadního řešení popsaného v kapitole 2.1.

2.2.1 Informační systém

Hlavním požadavkem na vytvoření systému je zjednodušení pořádání a správy závodů Šplhu na laně. Prvním požadavkem na systém je možnost registrace závodníků, a to jak do systému, tak i na jednotlivé závody. Systém by měl také obsahovat jednoduché statistiky pro závodníky. Největší zjednodušení by se mělo týkat zejména pořadatelů. Systém by měl umět vytvářet jednotlivé akce a měl by podporovat jejich následné editování a správu. To zahrnuje možnost upravovat informace o závodnících a její časy. Další funkcí systému by měla být možnost streamování průběhu akce. Dalším požadavkem je automatické generování výsledkových listin. Systém by také měl obsahovat editor pro vytváření a úpravu šablon diplomů. U každé akce by měla být možnost tisku vybrané šablony diplomu. Jedním z posledních požadavků na systém je podpora automatického módu, kdy se časy automaticky vkládají z časomíry do systému. Posledním požadavkem je možnost vytvořit QR kód, odkazující na akci.

2.2.2 Časomíra

Největším požadavkem na časomíru je její integrace do systému a také možnost připojení externích zařízení jako jsou reproduktory či obrazovka. Dalším požadavkem je zmenšení časomíry a její snadnější montáž a příprava. Zjednodušení montáže se týká zejména čidel a to tak, že čidla by měla být kompletně bezdrátová. Také maximální zvýšení bezchybnosti spodního čidla je jedním z důležitých požadavků.

2.2.3 Požadavky v bodech

- Informační systém:
  - Správa a editace akce
  - Přenos časů závodníků
  - Generování výsledkových listin
  - Vytváření a editace diplomů
  - Tisk diplomů
  - Statistiky pro závodníky
  - Automatický režim
  - Lze vytvořit QR code odkazující na stream závodu
- Časomíra:
  - Je integrovaná do systému
  - Měří čas závodníků
  - Odesílá změřené časy do systému, která je zpracuje
  - Má vývod jack 3,5mm
  - Má HDMI
  - Podporuje možnost připojení k monitoru přes HDMI a zobrazovat stream akce
  - Obsahuje jeden bezdrátový spínač k lanu (spodní) a bezdrátový spínač na lano (horní)

Kapitola 3

Použité technologie

V této kapitole se věnuji popisu technologií, které jsem použil při programování informačního systému pro akce Šplhu na laně a pro vytvoření integrované časomíry.

3.1 HTML a CSS

Tyto technologie se využívají pro tvorbu jednoduché prezentace dat na internetu nebo k vytváření webových dokumentů. Já jsem je využil pro vytvoření frontendu.[10]

Html je složeno z jednotlivých elementů, tzv. tagů, a jejich vlastností. Tagy nepárové jsou složeny pouze z jednoho tagu. Každý tag začíná `<` a končí `>`. Mezi těmito závorkami je pak název tagu a popřípadě nastavení jeho vlastností. Existují párové a nepárové tagy. Párové tagy jsou obvykle složeny ze dvou tagů. První je otvírací a druhý uzavírací. Oba mají stejný název, ale u uzavíracího tagu tento název začíná lomítkem. Tagy mohou mít také různé nastavitelné parametry, ovlivňující jejich vzhled či jejich chování. Obsah dokumentu se poté vkládá mezi jednotlivé tagy. Tím se vytváří a upravuje obsah stránky nebo dokumentu do požadované struktury. Html soubor je uložen vždy s příponou `.html`.[9]

Ovšem pro pokročilejší formátování úpravu obsahu se využívá css. Css definuje pomocí selektorů vzhled a může definovat různé animace tagům. Jedním z nejvíce využívaným selektorem je třída. Třída se definuje pomocí jména třídy a deklarací vlastností pro danou třídu. U třídy začíná název tečkou. Deklarace vlastností je obsažena vždy mezi `{` a `}`. Třída se pak v html implementuje pomocí speciálního atributu class. Do něj se vloží jeden nebo více názvů tříd, které se mají na daný tag aplikovat. Selektorem může také být tag jako takový. V tomto případě je selektorem název tagu. Další možností je vytvořit selektor určený pro jeden konkrétní tag s unikátním atributem id. Css také podporuje výběr tagů pomocí kombinace dříve zmíněných způsobů. Css soubor je uložen s příponou `.css`.[8]

3.2 PHP

Php se využívá pro zpracování dat nebo pro vytváření obsahu na straně serveru. Já jsem tuto technologii využil pro backend. Využívám ji jak pro dynamické generování obsahu stránek, ale také na zpracování a uložení dat a pro komunikaci mezi systémem a elektronickou časomírou.[16] Php soubor vždy začíná `<?php` a je ukončen `?>`.[17] Kód je vždy obsažen mezi těmito uvozujícími znaky.[17] Php také umožňuje úpravu souborů uložených na serveru.[16] Umí také pracovat s databázemi. Php je programovací jazyk, obsahující klasické statemanty jako jsou třeba `if`, `if-else` nebo třeba `switch`. Tento jazyk je casesensitive. Proměnné se

definují pomocí `$` za nímž následuje její název. [18] V tomto jazyce se nedeklaruje explicitně typ proměnných. Její typ se deklaruje sám při jejím prvním použitím. Php skript je uložen s příponou `.php`. [16]

3.3 MySQL

Pro ukládání dat do databáze jsem si vybral MySQL. Je to jazyk pro práci s databázemi určený pro webová systémy. Má implementované všechny potřebné příkazy jako jsou `SELECT`, `INSERT INTO`, `WHERE` a další. Příkazy pro práce s databází se vkládají do php skriptů. Ty pro to využívají různá rozšíření php. Já používám rozšíření PDO, které spojuje práci s dvanácti různými databázovými systémy a je tak univerzální.[11]

Příkazy v mysql jsou složené z několika částí. Některé příkazy, jako je `SELECT`, pro výběr dat z databáze, obsahují klíčová slova `SELECT`, poté následuje výběr konkrétních sloupců z tabulky. Dále následuje slovo `FROM` a určení konkrétní tabulky, ze které chceme data vybrat.[20]

Příkaz `INSERT INTO` vkládá nová data to databáze. Začíná slovem `INSERT INTO` následuje název tabulky, které se vkládání týká, a výběr konkrétních sloupců. Výběr je uvozen `(` a `)`. Dále následuje `VALUES` obsahující hodnoty pro vložení. Hodnoty jsou opět vloženy mezi `(`, `)`. Pořadí hodnot je v souladu s pořadím vybraných sloupců, a i jejich počet musí být totožný.[19]

Dále zde máme příkaz `WHERE`. Jedná se o jakýsi filtr, a to nejen pro příkazy `SELECT`, ale i pro `UPDATE` či `DELETE`. Tento filtr se přidává přímo za příkazy. Filtr obsahuje slovo `WHERE` a dále jeho podmínky. Syntaxe u dalších příkazů je velmi podobná.[21]

3.4 JavaScript

Jedná se o programovací jazyk na programování webových rozhraní. Javascript běží implicitně v každém prohlížeči. Díky tomu není potřeba u něj importovat žádné externí knihovny. Javascript pracuje s DOMem. Díky tomu jsme schopní pracovat s jednotlivými elementy html dokumentu. Proměnné v tomto jazyce jsou definovány pomocí typu `var`. Za tímto typem následuje název proměnné. Já využívám tento jazyk spíše na xmlhttprequesty. Pro práci s DOMem a úpravy obsahu stránky využívám JQuery. Soubory s Javascripty jsou uloženy s příponou `.js`.[12]

3.5 JQuery

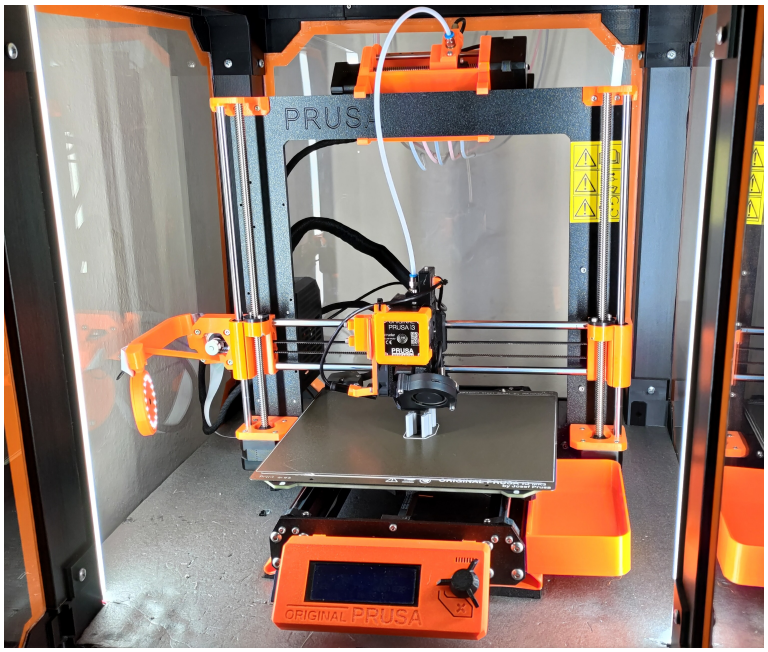
JQuery je knihovna využívající Javascriptu pro usnadnění práce s webovou stránkou a s DOMem.[14] Tuto knihovnu používám pro manipulaci s daty na straně uživatele a také jako rozšíření Javascriptu.[14] Pomocí této knihovny se dá jednoduše získat data z jednotlivých elementů a také je jednoduše upravovat.[14] Dají se i měnit vlastnosti jednotlivých tagů, nebo upravovat css styly. Syntaxe je velmi jednoduchá. Vždy začíná selektore, skládající se ze dvou částí. První je `$`, za ním následuje `(`, `)` obsahující výběr elementu.[15] Knihovna implicitně vybere první shodný element se selektorem. Element můžeme vybrat několika způsoby. Jedním je výběr pomocí třídy, kdy vezmeme její název začínající tečkou. Další možností je výběr pomocí atributu `id`. V tomto případě začíná název identifikátoru `#`.[15] Poslední způsob výběru je pomocí názvu tagu.[15] U této možnosti se vloží do selektoru pouze název tagu. JQuery podporuje selektory zkombinované z těchto variant. Díky tomu

můžeme jednoznačně vybrat konkrétní element, který nás zajímá. Po selektoru následuje tečka a název příkazu co se má provést. Velkou výhodou je možnost řetězení příkazů za sebou pomocí teček. Dá se díky tomu provést delší procedura v jednom řádku kódu. Posloupnost provádění příkazů je z leva doprava.

Jelikož se jedná o knihovnu, je potřeba ji importovat. Kód se dá kombinovat s Javascriptem a je uložen v souborech s příponou `.js`.

3.6 FDM 3D tisk

FDM je zkratka pro Fused Deposition Modeling a jedná se o aditivní výrobní proces, patřící do skupiny vytlačování materiálů. Objekt je vytvářen postupným nanášením roztaveného materiálu do předem určených cest, vrstvu po vrstvě. Používanými materiály jsou termoplastické polymery ve formě strun. Konkrétně se používá celá řada materiálu. Počínaje PLA, PETG a ABS, které jsou ty nejzákladnější, až po pružné nebo průhledné filamenty. Také se používají filamenty s různými příměsemi. Těmi můžou být karbonové částice pro velmi pevné vlastnosti objektu, hoblinky dřeva pro dřevěně vypadající vzhled a mnoho dalších.[7]



Obrázek 3.1: **3D tiskárna – Prusa MK3S+ MMU2S**

FDM je nejrozšířenější technologie pro 3D tisk. Proces začíná vygenerováním cest tisku z vymodelovaného objektu. Tento proces se nazývá slicování a získá se za jeho pomoci soubor s příponou `.gcode`. Po získání a nahrání tohoto souboru na 3d tiskárnu se zapne proces tisku. Ze všeho nejdříve se na požadovanou teplotu, určenou typem materiálu, vyhřeje tisková podložka a extruder. Po dosažení těchto teplot se provede kalibrace tiskárny, zavedení filamentu do extruderu a započne tisk. Tiskárna navede extruder a podložku na počáteční pozice. Extruder začne vytlačovat roztavený materiál na podložku v určených cestách. Po dokončení vrstvy se extruder zvedne na výšku vrstvy další. Tento proces pokračuje až do konce tisku.

Kapitola 4

Návrh

Tato kapitola je zaměřena na návrh informačního systému a na návrh architektury elektronické časomíry.

4.1 Přístup do systému

Ze všeho nejdříve jsem navrhl počet úrovní přístupů do systému. Vzhledem k požadavkům, viz. část 2.2, jsem se rozhodl u svého řešení pro čtyři úrovně přístupu do systému. První a nejnižší úroveň přístupu bude pro neregistrované uživatele. Těmito uživateli jsou myšleni návštěvníci akcí Šplhu na laně nebo prozatím neregistrovaní závodníci. Další úroveň bude určena pro registrované závodníky a třetí úroveň bude pro pořadatele akcí. Poslední, tedy nejvyšší úroveň přístupu do systému, bude pro administrátora systému. Systém bude umožňovat registrovat se a zažádat o přidělení dané úrovně přístupu, vyjma první úrovně.

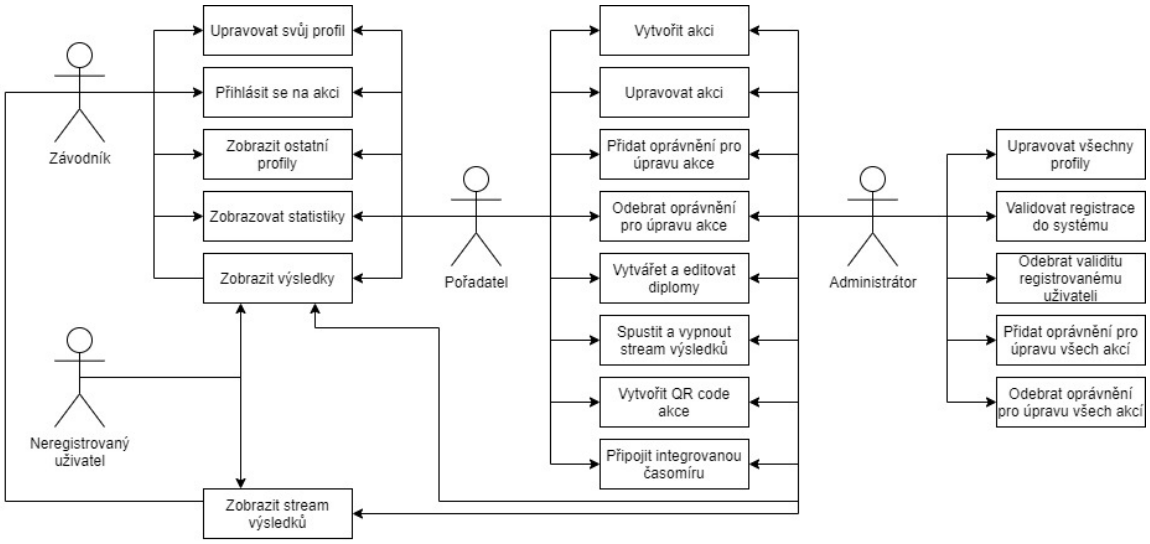
4.2 Funkce systému

Každá úroveň přístupu do systému bude umožňovat používání jiných funkcí a prostředků tohoto systému. Popis vychází z diagramu 4.1 níže. Na nejnižší úrovni určené pro neregistrované uživatele, budou mít návštěvníci právo na zobrazování finálních výsledků všech proběhlých závodů. Také budou mít možnost zapnout sledování streamu výsledků, aktuálně probíhajících akcí.

Další, tedy druhá úroveň přístupu učená pro závodníky, bude umožňovat registraci závodníka na vytvořenou akci, která ještě neproběhla. Závodník bude také mít v rámci svého účtu profil, který bude moci upravovat spolu se svými přístupovými údaji. Každý závodník si také bude moci prohlédnout profily ostatních uživatelů. Další funkcí bude vytváření jednoduchých statistik pro závodníka, obsahující základní data.

Třetí úroveň bude obsahovat funkce pro pořadatele. Jednou z těchto funkcí bude vytvoření nové akce nebo úprava již vytvořené akce. Pořadatel bude mít možnost přidělit nebo odebrat práva pro úpravu akce. Ovšem jen u akcí, kde má toto právo on sám. Pořadatelé budou mít k dispozici editor diplomů, pro jejich vytváření a úpravu, v rámci systému. U jednotlivých akcí budou moci spouštět a vypínat stream dané akce nebo v rámci akce vytvářet QR code odkazující na ni. Pořadatel také bude mít práva pro připojení integrované časomíry k akci. Všechny funkce určené pro závodníky bude obsahovat i úroveň pro pořadatele.

Poslední úroveň bude obsahovat všechny funkce určené pro závodníky i pro pořadatele. Navíc bude mít administrátor právo upravovat profily všech uživatelů. Další funkcí pro tuto úroveň bude validování nových účtů. Administrátor bude mít také právo upravovat veškeré vzniklé akce a odebírat či přidělovat práva na jejich úpravu.



Obrázek 4.1: **Use Case Diagram**

4.3 Datový model

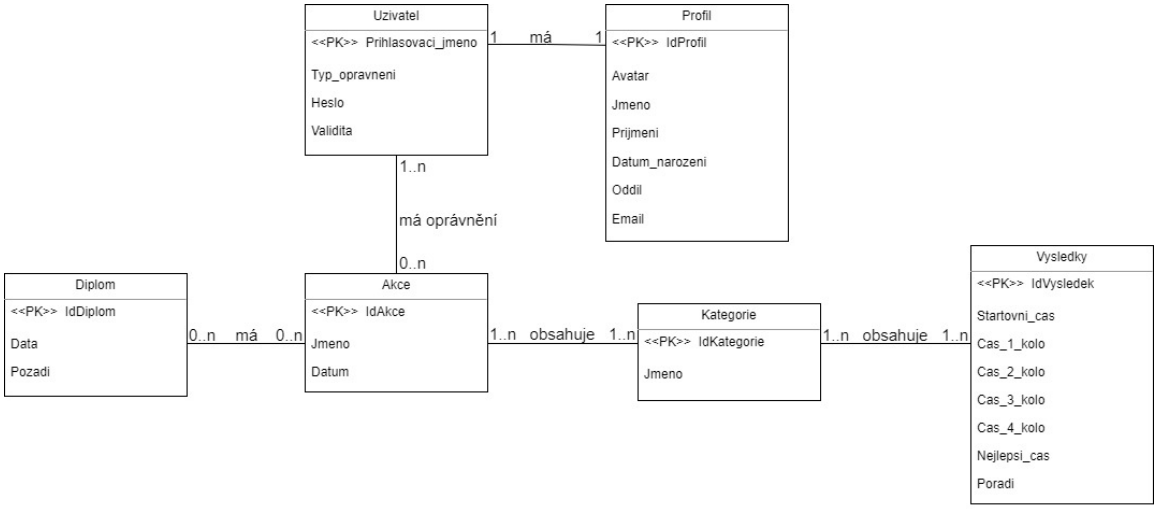
Dalším krokem návrhu je vytvoření datového modelu a vztahů mezi daty. Toto popisuje diagram 4.2 níže. Navrhl jsem celkem šest objektů. Prvním objektem je `Uzivatel`. Ten využívá uživatelské jméno jako primární klíč `Prihlasovaci_jmeno`. Dále objekt uchovává heslo uživatele `Heslo`, jestli má uživatel validní účet `Validita`. Uživatel může mít oprávnění k žádné nebo k více akcím `Akce`.

Každý uživatel má právě jeden profil popsaný objektem `Profil`. Ten má primární klíč `IdProfile`. V profilu jsou obsaženy detailní informace o uživateli. Jako je profilový obrázek `Avatar`, křestní jméno `Jmeno` a příjmení `Prijmeni` dále také datum narození `Datum_narozeni`, oddíl, který reprezentuje `Oddil` a kontaktní emailovou adresu `Email`.

Dalším objektem je `Akce`. Primárním klíčem pro akci je `IdAkce`. Každá akce má název akce `Jmeno`, datum konání, ve kterém akce probíhá `Datum`. Akce může mít žádný nebo více diplomů `Diplom`. Dále každá akce obsahuje jednu nebo více kategorií popsané objektem `Kategorie`.

Předposlední objekt `Kategorie` se zaměřuje na kategorie, do kterých jsou závodníci rozděleni. Kategorie obsahuje primární klíč `IdKategorie` a název dané katehorie `Jmeno`. Také každá kategorie obsahuje jeden či více výsledků `Výsledky`, které jsou také posledním objektem.

Výsledky jsou složeny z primárního klíče `IdVysledek`, z času `Startovni_cas`, podle kterého se závodní zařadí v prvním kole závodu. Dalšími atributy jsou časy jednotlivých kol `Cas_1_kolo`, `Cas_2_kolo`, `Cas_3_kolo`, `Cas_4_kolo`, nejlepší dosažný čas závodníka `Nejlepsi_cas` a poslední atribut obsahující dosažené pořadí na akci `Poradi`.



Obrázek 4.2: **ER Diagram**

4.4 Uložení dat

Po rozvržení funkcí systému a rozložení dat je nutné navrhnout uložení těchto dat. Já jsem vybral uložení dat do databáze. A to z důvodu jejího jednoduchého použití jak pro malé, tak i pro větší aplikace nebo pro její rychlé a snadné vyhledávání dat. Dále budu popisovat schéma databáze, viz. schéma 4.3, kterou jsem pro můj systém vytvořil.

Návrh obsahuje devět tabulek. První je tabulka `users`. Tato tabulka obsahuje informace o účtu uživatele. Primárním klíčem je zde uživatelské jméno, které musí být unikátní. Má typ `varchar` s maximální délkou třiceti znaků. Dalším polem tabulky je heslo. Do něj se ukládá hashovaný obraz hesla typu `varchar` s fixní délkou třiceti dvou znaků. Pole `Type` obsahuje číselnou informaci o typu přístupu do systéme. Toto číslo je typu `int` a má délku jedna. Poslední informací je validita účtu `Valid` s datovým typem `tinyint` o délce jedné cifry, jedná se tedy o příznak validity.

Další je tabulka `prifiles` v níž jsou uloženy detailní informace o uživateli. Pole `IdProfile` obsahuje automaticky generovaný primární klíč s typem `int` maximální délce čtyř číslic. Další pole `Avatar` obsahuje profilový obrázek uživatele s type `mediumblob` obsahující pouze data. Přípona obrázku je uložena v poli `AvatarType` s typem `varchar` a maximální možnou délkou patnácti znaků. Jméno a příjmení je uložené v polích `Name` a `Surname` obě s typem `varchar` a maximální délkou třiceti znaků. Datum narození je uložené jako `Birthday` s typem `date`, které je určené pro ukládání dat. Dalšími informacemi jsou oddíl závodníka `Troop` a jeho kontaktní email `Email`. Obě tyto informace mají typ `varchar` s maximální délkou třiceti znaků. Poslední pole obsahuje cizí klíč `Uname` ukazující na uživatele, kterému profil patří.

Třetí tabulka s názvem `events` obsahuje informace o jednotlivých akcích. Má primární klíč pojmenovaný `IdEvent` s typem `int` a délkou maximálně čtyř číslic. Tento klíč je opět automaticky generovaný. Dále je zde obsažen název akce `Event` a datum konání akce `Date`. Název má datový typ `varchar` s maximální délkou třicet znaků a datum má typ `date`.

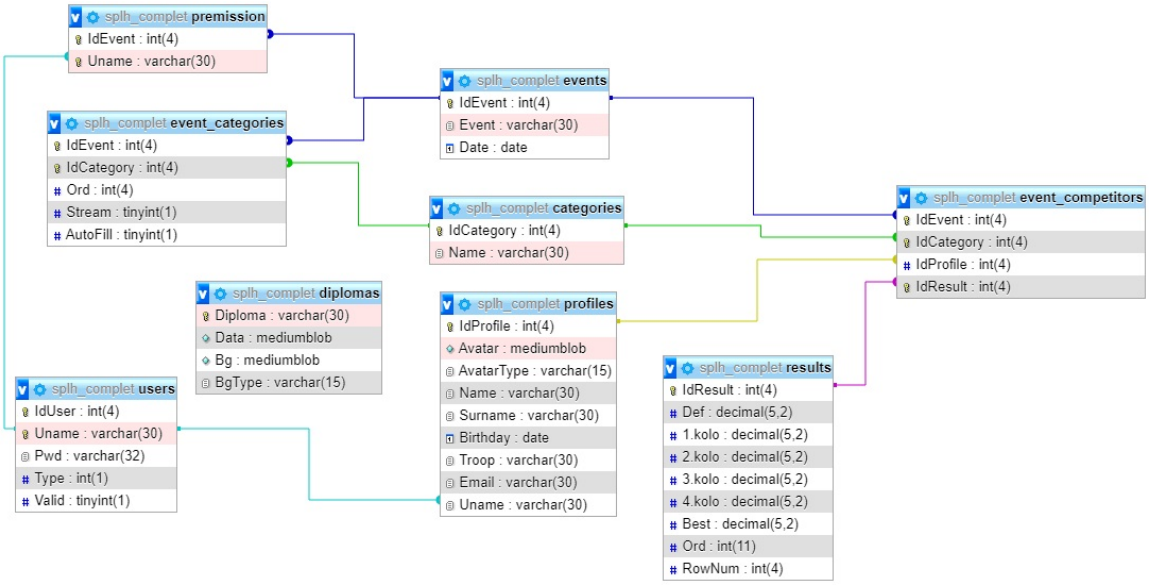
Tabulka `categories` uchovává jméno kategorie. Tato tabulka obsahuje pouze automaticky generovaný primární klíč `IdCategory`, typu `int` a maximální délky čtyř číslic a jméno kategorie `Name`. Jméno má typ `varchar` a má maximální délku třicet znaků. Tuto tabulku

jsem navrhl pro uchování a znovupoužití jedné kategorie z důvodu úspory datového prostoru.

Výsledky jsou uloženy v tabulce `results` s primárním klíčem `IdResult`. Tento klíč je opět generován automaticky a má typ `int` o maximální délce čtyř číslic. Tabulka dále obsahuje veškeré časy závodníků. Startovní čas závodníka `Def`, čas dosažený v prvním kole `1.kolo`, v druhém kole `2.kolo`, ve třetím kole `3.kolo`, ve čtvrtém kole `4.kolo` a nejlepší dosažný čas v závodě `Best`, jsou všechny stejného typu `decima`, tedy čísla s pevnými dvěmi ciframi za desetinou čárkou a maximálně pěti číslicemi před čárkou. Další pole obsahuje dosažené pořadí závodníka `Ord` s typem `int` a maximální délkou jedenácti číslic. Poslední pole `RowNum` obsahuje číslo určující pořadové číslo záznamu v tabulce zobrazované uživateli. Má typ `int` s délkou čtyř číslic.

Pro uchování informací o tom, která akce má jaké kategorie slouží tabulka `event_categories`. Ta má složený primární klíč ze dvou cizích klíčů, a to z `IdEvent` a z `IdCategory`. Dále je u kategorií zastoupených v jednotlivých akcí potřeba uchovat jejich pořadí. To zajišťuje informace `Ord` s datovým typem `int` o maximální délce čtyř číslic, v níž je uložené pořadí kategorie. Další pole `Stream` a `AutoFill` osahují příznaky stavu streamu výsledků a připojení integrované časomíry. Obě pole mají stejný datový typ `tinyint` i stejnou délku jedné číslice.

Pro uchování informací, ke kterému závodníkovi v dané kategorii na dané akci patří konkrétní výsledek slouží tabulka `event_competitors`. Primární klíč je zde složen ze tří cizích klíčů. A to z `IdEvent` odkazující na konkrétní akci, `IdCategory` odkazující na konkrétní kategorii a `IdResult` odkazující na konkrétní výsledky. Poslední informací v této tabulce je cizí klíč `IdProfile`, který odkazuje na profil s informacemi závodníka, a tedy i na konkrétního uživatele.

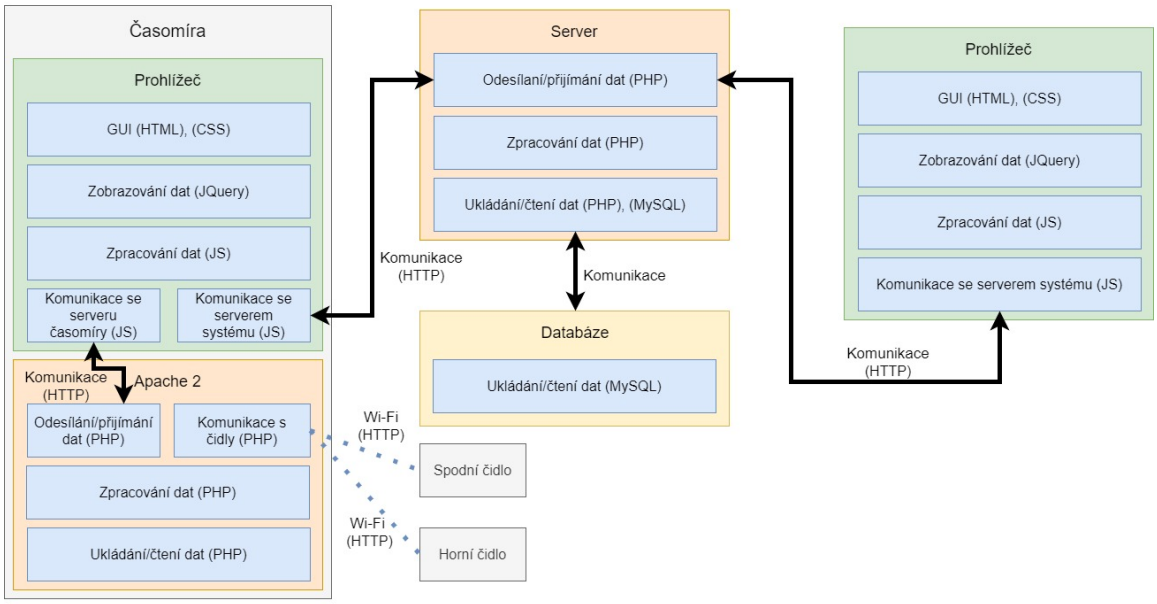


Obrázek 4.3: **Schéma databáze**

Tabulka `premission` slouží k uchování informací o tom, jaký uživatel má oprávnění k upravování jednotlivých akcí. Obsahuje pouze dva cizí klíče `IdEvent` a `Uname`, které jsou zároveň i složeným primárním klíčem.

Poslední tabulka `diplomas` obsahuje uložené diplomy. V E–R diagramu, viz. 4.2, je mezi objektem `Diplom` a objektem `Akce` oboustranná vazba `0..n`. V systému ovšem není třeba uchovávat informace o tom, jaká akce má který diplom, jelikož si diplom k akci si uživatel vždy vybere z nabídky diplomů. Primárním klíčem je zde unikátní název diplomu `Diploma` s typem `varchar` a s maximální délkou třiceti znaků. Dále jsou zde v poli `Data` uložená data obsahující strukturu a text diplomu. Toto pole je typu `mediumblob`. Data pozadí diplomu jsou uložena v poli `Bg` a je taktéž typu `mediumblol`. Přípona pozadí je uložena v `BgType`, je typu `varchar` s maximální délkou patnácti znaků.

4.5 Blokové schéma systému



Obrázek 4.4: **Blokové schéma systému**

Blokové schéma 4.4 ukazuje jednotlivé vrstvy systému, komunikaci mezi nimi a komunikaci mezi systémem a integrovanou časomírou. Na serveru systému jsou klíčové vrstvy pro ukládání dat a pro dynamické generování webového rozhraní. Jsou zde také vrstvy pro posílání dat mezi webovým rozhraním systému a serverem a mezi integrovanou časomírou a serverem. Komunikace probíhá pomocí http protokolu. Další vrstva se stará o zpracovávání dat a jejich transformaci na požadovaný tvar pro možnou další práci, jako je třeba uložení do databáze nebo úprava dat v databázi. Server má také přístup k serveru s databázemi.

Ve webovém prohlížeči se generuje uživatelské rozhraní, pod kterým se skrývá několik vrstev. První slouží pro zobrazování zpracovaných dat do grafického rozhraní. Následující vrstva zpracovává data, která posílá serveru nebo která naopak od serveru dostává. Poslední vrstva komunikuje se systémovým serverem a odesílá a přijímá od něj data.

Integrovaná časomíra obsahuje dvě části. První je frontend. Ten zajišťuje webový prohlížeč, v němž se zobrazuje, stejně jako u webového systému, uživatelské rozhraní. Pod ním je

opět několik vrstev zajišťující správnou interpretaci dat. Stejně jako u webového rozhraní je zde vrstva zajišťující správné zobrazování dat, získaných nejen od systémového serveru, ale i od serveru časomíry. Další vrstva zpracovává data pro odeslání či pro zobrazení. Poslední vrstva je rozdělena na dvě části. První část komunikuje se serverem časomíry a přijímá od něj data nebo je naopak na něj odesílá. Druhá část se stará o integraci časomíry do systému a to tak, že komunikuje se serverem systému a předává mu potřebná data nebo je od něj získává.

Server na časomíře obsahuje vrstvu pro komunikaci s uživatelským rozhraním časomíry. Na stejné úrovni je i komunikace s čidly, která probíhá výhradně přes Wi–Fi. Další vrstva zpracovává data do požadované formy pro další použití. Poslední vrstva ukládá nebo čte data na serveru.

4.6 Desky čidel

Další fází je návrh desek pro horní i dolní čidlo. U navrhování jsem se rozhodl použít pro obě čidla snímání stavů pomocí měření elektrické veličiny kapacity. Při změně kapacity se pošle signál do časomíry a ta informaci zpracuje a interpretuje. Implementaci komunikace a zpracování dat se blíže věnuji v kapitole 5.4.

Pro měření jsem zvolil kapacitní senzor TonTouch TTP223, SOT-23-6L. Důvodem jeho výběru je dostupnost, cena a jeho vestavěná funkce autokalibrace. Tento integrovaný obvod má vstup, k němuž je připojena veličina, která nás zajímá. Další vstup nastavuje výstup na LOW nebo na HIGH, také má vstup na nastavení chování čipu. Čip se může chovat jako spínač nebo jako tlačítko. Obvod má jen jeden výstup, který udává stav stisknutí.

Dále jsem navrhl obvod pro výše zmíněný senzor vycházející z jeho obvyklého zapojení. Obě plošné desky mají připojení pro desku ESP8266 pomocí pinů. Dále bylo nutné plošnou desku napájet. To je zajištěno Li–Pol baterií s kapacitou 3000mAh. Baterie je zároveň připojená k nabíjecímu obvodu pro tento typ baterií. Spodní čidlo má v rámci plošné desky spirálu připojenou k integrovanému obvodu na snímání kapacity. Horní čidlo tuto spirálu nemá, protože se jedná o čidlo visící na laně a snímaná plocha musí být obtočená kolem něj. Proto je v tomto případě k plošné desce připojena měděná fólie. Hlavním důvodem k výběru měděné fólie byla její pružnost. Dá se ohnout bez poškození. Toto je důležitá vlastnost, protože horní čidlo musí být rozevíratelné a to kvůli usnadnění montáže na lano. Dosavadní čidlo se musí na lano nasouvat, a to je zbytečně náročné. Horní čidlo se stejně jako původní čidlo na lano fixuje pomocí čtyř šroubů.

4.7 Grafické rozhraní

V této kapitole se věnuji návrhu vlastního uživatelského rozhraní pro informační systém, ale také pro integrovanou elektronickou časomíru.

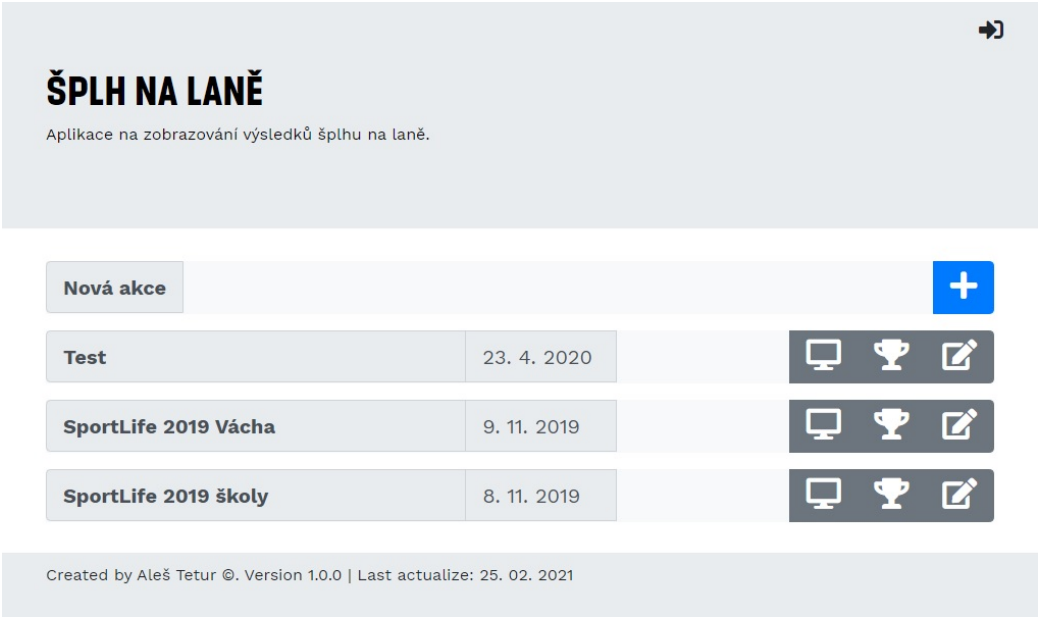
4.7.1 Informační systém

Při návrhu grafického rozhraní jsem použil již přednastavených stylů Bootstrap 4. Převzal jsem pouze tyto styly avšak finální vzhled grafické rozhraní jsem navrhl sám. U rozhraní jsem několikrát konzultoval jeho vzhled s budoucími uživateli. Barva rozhraní je laděná do bílošedé barvy stejně jako barva plastových dílů elektronické časomíry. Pro styl písma jsem použil Sokolské fonty `Tyrs` a `Fügner` v souladu s grafickým manuálem vydaným ČOS. Výsledný design je vidět na obrázku 4.5 níže. U systému se pouze mění obsah stránky v

prostřední části. Záhlaví a zápatí zůstává všude stejné. Odlišný design je pouze na stránce obsahující stream výsledků. Zde jsem navrhl bílé pozadí s informacemi o aktuálním závodníkovi a ve spodní části je tabulka se závodníky, kteří jsou dále na řadě. Pokud není stream výsledků u akce zapnutý, je zde napsaný pouze text "Stream výsledků závodu ještě nezačal.".

Další částí rozhraní jsou výsledky akcí. Zde jsou obsažené tabulky s výsledky jednotlivých kategorií. Je zde možnost vyhledávání v tabulkách konkrétní informace. V horní části stránky jsou tlačítka pro stažení výsledků ve formátu `pdf` nebo pro jejich tisk.

Stěžejní částí rozhraní je editor akcí. Ten je rozvržen jako tabulka. V horní části je seznam záložek, které lze přejmenovat nebo odstranit. Záložky také zobrazují informaci, jestli je konkrétní list streamování nebo jestli je připojen k integrované časomíře. Tabulka obsahuje sloupce s pevnými názvy. Dále tabulka obsahuje jednotlivé řádky obsahující informace. Ty se dají upravovat a mazat podle potřeby. Pod tabulkou jsou umístěny tlačítka pro ovládání a úpravu tabulek. I na této stránce je tlačítko pro vyhledávání v tabulkách nebo pro tisk tabulek a jejich exportování do pdf nebo formátu xmlx.



Obrázek 4.5: **GUI – Homepage**

Další částí rozhraní je rozbalovací menu. To je umístěno v pravém horním rohu. Před přihlášením uživatele odkazuje menu na přihlášení, po kterém se změní na navigaci v rámci správy systému. Menu obsahuje odkazy na všechny části správy, tj. na úpravu profilu, nastavení, editor diplomů a statistiky. Upravování profilu je rozvržené jako jednoduchý formulář s možností úpravy uložených informací. V nastavení je pomocí karet rozdělení několika částí. První jsou profily jednotlivých uživatelů, které jsou rozděleny podle úrovně práv v systému. Dále pak následují karta se seznamem všech akcí s možností přihlásit se na ně. Další karta není určena pro závodníky, a proto se jim nezobrazuje. Obsahuje totiž akce, ke kterým má uživatel práva úpravy a může zde práva přidělit dalšímu uživateli. Poslední karty jsou určené pro administrátory. První zobrazuje seznam všech validních účtů a umožňuje zrušení jejich validity. Druhá pak ukazuje seznam všech nevalidních účtů, a naopak umožňuje jejich validaci. Editor diplomů dává nejdříve na výběr, jaký diplom chce uživatel

upravovat. Jsou zde tři možnosti. Prví je úprava již existujícího diplomu. Další možností je nahrání diplomu, který musí být ve specifickém souboru. Poslední možností je vytvoření nového diplomu. Po výběru jedné z těchto možností se zobrazí lišta pro editaci a pod ní se zobrazí stránka s diplomem ve formátu A4. Poslední částí pro uživatele jsou statistiky. Ty jsou opět rozděleny do jednotlivých záložek, které obsahují vždy jeden graf. Každá záložka obsahuje jiný typ grafu a zobrazuje jiná data. První chronologicky zobrazuje všechny nejlepší dosažené časy na závodech v lineárním grafu. Druhá záložka ukazuje umístění na jednotlivých závodech na sloupcovém grafu. Další záložka zobrazuje donutový graf se sumarizací počtů všech umístění. Poslední graf ukazuje, kolik závodníků bylo na jednotlivých závodech lepších než uživatel, horších než uživatel a kolik závodníků na tom bylo podobně. Opět je to zobrazeno ve sloupcovém grafu. Poslední tlačítko v menu slouží pro odhlášení.

Pro přihlášení jsem navrhl jednoduchý formulář. Na konci tohoto formuláře je možnost kliknout na registraci pro neregistrované uživatele. Po kliknutí na tuto možnost se formulář změní na registraci.

4.7.2 Časomíra

Uživatelské rozhraní u integrované časomíry jsem navrhoval sám. Je inspirované vzhledem rozhraní moderních zařízení a telefonů. Je laděné do bílošedé barvy stejně jako webové rozhraní. Během návrhu jsem design několikrát konzultoval s přáteli a budoucími uživateli. Obrázek 4.6 níže je finální vzhled grafického rozhraní pro integrovanou časomíru. V záhlaví jsou zobrazené informace týkající se stavů časomíry. První status zobrazuje zda-li je časomíra připojena do systému nebo není. Uprostřed je název časomíry. Vlevo jsou indikátory vybraného módu a indikátory stavů čidel, hlasitosti zvuku a připojení k Wi–Fi. Toto záhlaví je na všech stránkách stejné. V těle je šest tlačítek. První odkazuje na připojení časomíry do systému. Zde se musí uživatel přihlásit a zvolit si k jaké události časomíru připojí. Druhé tlačítko přesouvá uživatele na stránku s kontrolou časomíry. Zde je stopovaný čas a tlačítka k ovládání stopek. Třetím tlačítkem je nastavení, ve kterém jsou vypsány všechny stavy v přehledné tabulce. Je zde i možnost nastavit zvuk, mód a nastavení spojené s konkrétními módy. První tlačítko na druhém řádku přesune uživatele na stránku se spouštěním vysílání přes hdmi výstup. Předposlední tlačítko odkazuje na stránku s manuálním ovládáním časomíry. Jedná se o ovládání bez nutného použití čidel. Poslední tlačítko vypíná časomíru. Podrobnější implementaci se věnuji v kapitole 5.3.

Obrázek 4.6: **GUI – Časomíra homepage**

4.8 Hardware

Kapitola se zaměřuje na výběr hardwaru pro moje řešení elektronické časomíry a čidla.

4.8.1 Časomíra

Jako hardware pro elektronickou časomíru jsem zvolil desku Raspberry Pi 4 Model B - 4GB RAM a to z několika důvodů. Desky raspberry jsou cenově dostupné a výkonné. Běží na operačním systému Raspbian, který je na ně přímo stavěný. Raspberry pi 4 má také již implementované konektory pro periferní zařízení, včetně integrovaného Wi–Fi modulu, a má v systému na vše ovladače. Jeho použití je tedy velmi snadné. Také velikost je velmi příjemná. Integrace do systému tedy probíhá pomocí komunikace serveru na raspberry pi a webového serveru. Této integraci jsem se částečně věnoval v návrhu výše, viz. 4.5 a implementaci časomíry se podrobně věnuji v kapitole 5.3. Při implementaci desky jsem využíval její dokumentaci [6].

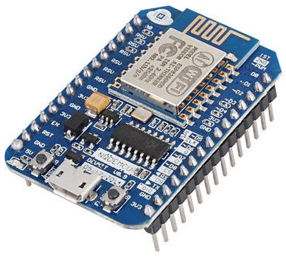
Obrázek 4.7: **Raspberry Pi 4 Model B - 4GB RAM**[1]

Dalším hardwarem je displej časomíry. Zde jsem zvolil displej Waveshare 5"dotykový LCD (H) displej, TFT, 800x480, kapacitní, HDMI, USB. Důvodem je připojení obrazu přes hdmi kabel, přenos dotykových dat, a to přes usb a nikoliv přes patici pinů. Dalšími důvody jsou integrovaná patice pro připojení reproduktorů, audio jack 3,5 pro připojení externích audio zařízení nebo možnost externího napájení. Připojení napájená je v tomto případě velmi žádoucí. Sice má deska Raspberry Pi 4 integrovaný usb hud, přes který lze displej napájet, ale při vyšší zátěži a poklesu napájecího proudu, deska jako první vypne napájení právě tohoto hudu. To zapříčiňuje občasné vypnutí displeje. Sice toto vypnutí trvá v řádu necelé sekundy ovšem displej se i po tak krátkém výpadku vypne a opětovné zapnutí trvá dalších několik sekund. To je u elektronická časomíry velmi nežádoucí chování. Také rozměry displeje jsou zcela vyhovující pro moje řešení.

[1]Obrázek je převzat z webové stránky:, https://www.rpishop.cz/raspberry-pi/1598-raspberry-pi-4-model-b-4gb-ram-765756931182.html

4.8.2 Čidla

Pro čidla jsem zvolil NodeMCU LUA ESP8266. Tato deska je malá a cenově dostupná. Důvodem jejího výběru je také její funkcionalita. Deska obsahuje integrovaný Wi–Fi modul a má digitální GPIO piny pro připojení externích zařízení. Pro snímání fyzikálních veličin jsem navrhl vlastní desku, do které je ESP8266 zasazeno. Vlastní desce se blíže věnuji v kapitole 4.6.



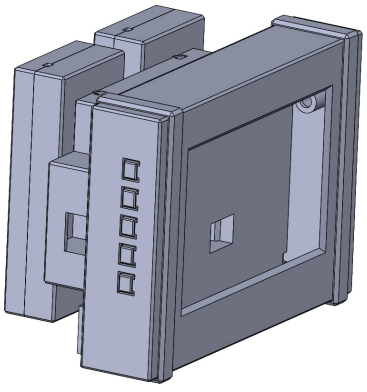
Obrázek 4.8: **NodeMCU LUA ESP8266**[2]

4.9 Modely

V poslední fázi návrhu, bylo zapotřebí vymodelovat veškeré díly na elektronickou časomíru a na čidla. Veškeré modelování jsem prováděl v programu SolidWorks.

4.9.1 Časomíra

Prvním je model pro samotnou časomíru. Ten se skládá z několika částí, a to z krytu na desku Raspberry Pi 4 obsahující otvory pro připojení externích zařízení a dva otvory pro zasunutí reproduktorů. Dalšími částmi jsou spodní kryt pro displej, do kterého je posazena deska a přední kryt na displej, který obsahuje pět tlačítek pro ovládání displeje a otvor pro 3,5 jack. Dále jsou součástí časomíry reproduktory, na které jsem také vymodeloval kryty, které ve spodní části obsahují koleje pro jejich zasunutí do časomíry.

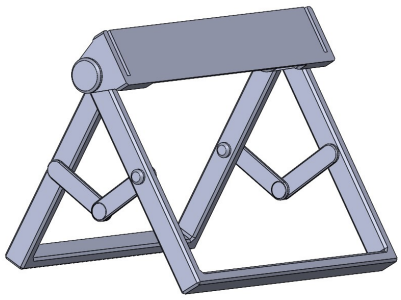


Obrázek 4.9: **Model sestavy časomíry**

[2]Obrázek je převzat z webové stránky:, https://www.gme.cz/nodemcu-lua-esp8266-ch340-wi-fi

4.9.2 Stojan na časomíru

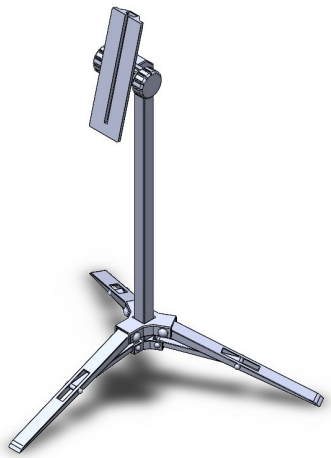
Dalším modelem je stojan pro časomíru. Ten je nutný pro lepší stabilitu časomíry na stole. Stojan je vytvořen dvěma nohami spojenými pomocí rozevíratelnými sponami a dosedací plochy s otvory pro kolejnice na časomíře. Tato plocha je k nohám připevněna pomocí kolíků.



Obrázek 4.10: **Model sestavy stojanu časomíry**

4.9.3 Stojan na reproduktory

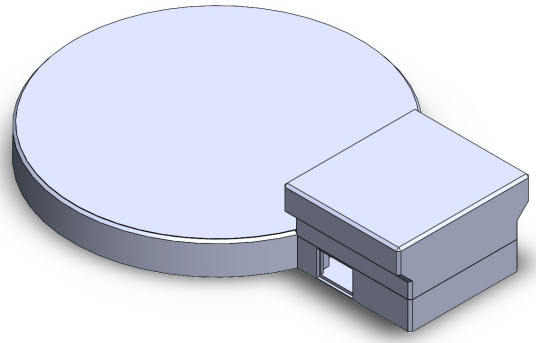
Třetím modelem je stojan na reproduktory. Uprostřed něj je noha, na kterou je připevněna dosedací plocha s otvorem na kolej pro reproduktor za pomocí odtahovacího šroubu s plastovými kryty. Na noze je dále jezdec, do kterého jsou připevněny tři nohy. Nohy jsou ke stojanu dále připevněny vzpěrami pro udržení stability. Stojan lze částečně rozebírat a také má skládací nohy pro úsporu místa při uschováváni časomíry. V tomto případě je většina dílu drobná což zapříčiňuje, že se nedají vytisknout kolíky pro jejich spojení. Proto jsou díly spojeny pomocí kolíků vytvořených z drátu kancelářských svorek. Tyto svorky jsem vybral, protože jejich průměr drátu vyhovuje mým požadavkům a jsou velmi levné.



Obrázek 4.11: **Model sestavy stojanu pro reproduktory**

4.9.4 Spodní čidlo

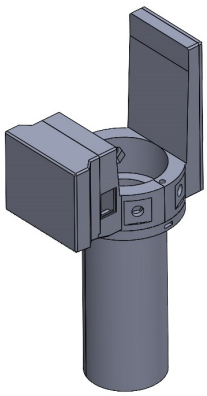
Model pro spodní čidlo je velmi jednoduchý. Je složen ze tří částí. Ze spodního dílu, ve kterém je je prostor pro uložení baterie a otvor pro zasazení desky spodního čidla. Dalším dílem je kryt na ochranu ESP8266, které je vložené v desce, před prachem a poškozením. Poslední součástí modelu je krytka na integrovanou spirálu čidla. Tato krytka není funkčně nutná, slouží pouze jako zakrytí spirály a jako designový prvek. Zajímavostí je, že tento díl je tvořen pouze jednou vrstvou plastu, která je tenká 0,2mm.



Obrázek 4.12: **Model sestavy dolního čidla**

4.9.5 Horní čidlo

Předposledním modelem je horní čidlo. Ten je oproti spodnímu čidlu velmi složitý. Je složen ze sedmi dílů. Tělo časomíry je tvořeno rozpůlenou trubkou, na kterou patří měděná fólie. Obě poloviny jsou spojeny pomocí vytištěného proužku tenkého plastu, který přesně zapadá do výřezů v půlválcích. Díky tomu je zajištěna rozevíratelnost horního čidla a zároveň i jeho nerozebíratelnost. V obou dílech jsou vloženy dva neodymové magnety pro jednodušší montáž a dvě čtvercové matice pro fixaci horní části čidla.



Obrázek 4.13: **Model sestavy horního čidla**

Horní část se skládá opět ze dvou polovin a obě také obsahují dva neodymové magnety. Každá z horní části má ke spojení na jedné straně výřez a na druhé straně výstupek, každý s dírou na fixační šroub. Dále obě části obsahují dvě díry pro aretační šrouby a výřezy pro matice, pomocí kterých se šrouby pohybují a zajišťují. Pro dosažení celkově vyšší pevnosti jsou spoje horní a dolní části čidla pootočeny o šedesát stupňů. Posledními částmi těchto dílů je u jednoho prostor pro baterii a u druhého prostor pro desku horního čidla spolu s vloženým ESP8266. Baterie je odpojovatelná z důvodu možnosti demontáže. Model obsahuje i krytky jak na ESP8266, tak i na baterii. Spojení spínací plochy s deskou jsem vyřešil pomocí pinu. V těle čidla vyčnívá jeden pin, který přesně zapadá do dutinky v horní části čidla.

4.9.6 Krabice

Posledním modelem je krabice na elektronickou časomíru. Model je velmi jednoduchý. Skládá se ze dvou částí. Z těla a z víka. U tohoto modelu jsem se inspiroval jedním modelem [5] ze stránky `prusaprinters.org`. Pouze jsem u něj změnil rozměry tak, aby vyhovovaly mému řešení. Dále bude v krabici pěna s otvory přímo na jednotlivé části elektronické časomíry. Důvodem je bezpečný převoz a nárazuvzdorost.

Obrázek 4.14: **Sestava krabice**

Kapitola 5

Implementace

V této části práce se zaměřuji na implementační detaily informačního systému, elektronické časomíry a její integrace. Ze začátku se věnuji popisu rozložení a umístění souborů. Poté přichází na řadu podrobnější popis jednotlivých částí systému.

5.1 Rozdělení systému

Řešení je rozděleno na klientskou a serverovou část. Klientská část zajišťuje interaktivní grafické rozhraní informačního systému a běží v prohlížeči na straně uživatele. Serverová část má na starosti zpracovávání požadavků na systém, ukládání dat a práci s nimi i celou logiku informačního systému.

5.1.1 Klientská část

Pro webové skripty jsem využil Javascript v kombinaci JQuery. Všechny jsou uloženy v adresáři `js`. Podobně jako u stylových souborů má každá stránka vlastní skripty pojmenované podle názvu stránky a s příponou `-script.js`. Skripty zajišťují zobrazování a zpracování dat na straně uživatele. Dále je také posílají na další zpracování na server.

5.1.2 Serverová část

Serverové skripty slouží pro zpracování dat na straně serveru a jejich následné uložení nebo odeslání. Skripty jsou psány pomocí `php` a jsou uloženy v adresáři `php`. Název vždy začíná názvem stránky, ke které skript patří a dále pokračuje názvem funkce skriptu s příponou `.php`.

5.2 Informační systém

Systém je rozdělen do několika částí. Každá část je uložena ve svém vlastním adresáři. V kořenovém adresáři se nacházejí soubory jednotlivých stránek, které jsou zobrazovány uživatelům. Níže budu popisovat jednotlivé části systému.

5.2.1 Kostra stránek systému

U všech stránek jsem využil dynamického generování obsahu. Veškeré části, které jsou totožné pro všechny stránky jsou uloženy ve vlastních souborech ve složce `templates` a jsou

psány jako php skripty. Jedná se konkrétně o hlavičkovou část html souboru `head.php` obsahující informace tagu `<head>`, tj. všechna metadata, import potřebných knihoven i Javascriptů. Dále jde o záhlaví stránky `header.php`. To obsahuje šedý kontejner s názvem systému odkazující na domovskou stránku. Dalším souborem ve složce `templates` je `menu.php`. Ten vygeneruje menu v závislosti na stavu přihlášení uživatele. Pokud je vytvořené `session` uživatele, je obsahem menu pro pohyb správou systému. Pokud ovšem není místo menu je vytvořeno tlačítko pro přihlášení. Posledním souborem je `footer.php` obsahující zápatí stránky s informacemi o autorovi, verzi systému a datum poslední aktualizace. Na stránce se pak tyto části importují pomocí php příkazu `include`.

5.2.2 Vzhled stránek systému

Grafický vzhled jsem řešil pomocí Bootstrap 4, ovšem pro úpravu některých prvků jsem použil vlastní styly. Tyto styly jsou všechny uloženy v adresáři `css`, v němž se nachází devět souborů. Každá stránka má vlastní soubor se styly pojmenovaný stejně jako stránka s příponou `-sheets.css`. Ty obsahují pouze styly, které jsou pro danou stránku upravené. Jsou–li některé upravené pro více nebo pro všechny stránky jsou uloženy v souboru `sheets.css`. Ten je importován ve všech stránkách.

Další součástí vzhledu jsou fonty. Jelikož nepoužívám standardní sadu fontů, je nutné použité fonty uložit. Proto jsou fonty uloženy ve složce `fonts`. Pro systém používám fonty, které určuje grafický manuál ČOS.

5.2.3 Domovské stránka

Domovská stránka slouží jako základní rozcestí pro uživatele. V obsahové části jsou vypsané jednotlivé akce, ke kterým může uživatel přistupovat. Každá akce má název, datum a menu obsahující nabídku, kam se může uživatel v rámci akce dostat. Menu vždy obsahuje tři možnosti, a to stream výsledků akce, výsledky akce a upravování akce.

Jednotlivé akce se zobrazují postupně ve formátu viz. obrázek 5.1. Po načtení stránky se spustí Javascriptový příkaz `setInterval`, který na server odesílá `XMLHttpRequesty` s intervalem dvou sekund. Požadavky jsou poslány na php skript `event-list.php`. Ten se připojí na databázi a provede příkaz pro zjištění všech akcí a seřadí je podle data od nejnovější akce po nejstarší. Výsledek dotazu odešle jako řetězec ve formátu JSON zpět Javascriptu. Po obdržení odpovědi je řetězec ve formátu JSON rozparsován a předán funkci pro generování jednotlivých štítků akcí.

Obrázek 5.1: **Štítek akce**

Generování štítků probíhá za pomocí JQuery, které jednoduše vytváří a vkládá elementy do DOMu. Pokud jde o první generování štítků tak se nejdříve vygeneruje štítek pro vytvoření nové akce a až poté se generují jednotlivé akce. Všechny štítky se vygenerují se stylem `display: none;`, který je skrývá. Po každém generování akce se zapne funkce `fade` na zobrazení štítků. Ta je s prodlevou zobrazuje, a to pomocí JQuery příkazu `fadeIn`. Při všech dalších generováních se nejdříve zkontroluje počet akcí. Pokud jsou všechny akce zobrazené nic se neprovede, jinak se přidá nový štítek akce.

5.2.4 Stránka s výsledky

Stránka s výsledky obsahuje tabulku pro každou kategorii dané akce. Při načtení stránky se volá funkce `loadData`, která odešle `XMLHttpRequest` s parametrem, obsahující název akce, na serverový php skript `results-data.php`.

Ten požadavek zpracuje a to tak, že se nejprve připojí na databázi a pomocí série tří komplexních sql dotazů získá veškeré informace o akci. První sql dotaz zjistí datum konání akce a uloží ho do pole `result` jako první prvek. Další sql dotaz zjistí názvy jednotlivých kategorií a počet řádků výsledků a také je vloží do pole `result` na pozici druhého prvku. Poslední sql dotaz zjistí všechna data o výsledcích akce, které také uloží do pole `result` a to na jako třetí prvek. Poté se pole odešle zpět jako řetězec ve formátu JSON. Jakmile Javascript tato data obdrží, vygeneruje tabulku pro každou kategorii. Některé hodnoty časů mají speciální interpretaci, např. čas `999.00` se zobrazí jako `Neplatný pokus` a čas `888.00` jako `Vynechaný pokus`. Tabulky a názvy kategorií jsou opět vygenerovány se stylem `display: none;`. Po vygenerování obsahu stránky se spustí pro každou tabulku a název kategorie funkce `timer`, která nastaví timeout zobrazení jednotlivých částí. Postupně se zobrazuje vždy kategorie a poté tabulka s rozmezím půl sekundy.

V tabulkách je možné vyhledávat. Toto zajišťuje funkce `search`, která se spustí vždy při změně vyhledávaného výrazu. Tato funkce je převzata z následujícího zdroje [13].

Systém dále umožňuje stažení výsledků ve formátu pdf nebo jejich tisk. Pro obě tyto funkce využívám knihovnu `pdfmake` [2]. Při volbě tisku výsledků se pdf se volá funkce `print`, která dokument otevře na nové stránce. Při stažení se volá funkce `downloadPDF`, která výsledky stáhne s názvem složeným ze jména akce a přípony `-výsledky.pdf`.

5.2.5 Přihlašování

Přihlašování do systému jsem řešil pomocí html formuláře. Všechny soubory přihlašování jsou uloženy v adresáři `login`, který má stejnou strukturu jako systém, tzn. Javascripty jsou umístěny ve složce `js` a php skripty jsou zase ve složce `php`. Jelikož se pro přihlašování využívají informace z databáze, je skript pro navázání spojení uložen v adresáři `db-work`.

Při načítání stránky zkontroluje Javascript get atributy poslané stránce a zobrazí příslušnou hlášku uživateli.

Při přihlašování uživatel zadá svoje uživatelské jméno a heslo. Po kliknutí na tlačítko přihlásit se aktivuje php skript `login.php`. Ten se připojí k databázi a zkontroluje zadané údaje. Pokud uživatel zadal správné heslo a uživatelské jméno, vytvoří se `session` proměnná s uživatelským jménem a s typem přístupu do systému. Potom je uživatel přesměrován buď na domovskou stránku nebo na stránku editoru akcí. To záleží odkud byl přihlašovací formulář spuštěn. Pokud byl spuštěn po kliknutí na tlačítko přihlásit, je uživatel přesměrován na domovskou stránku. Pokud byl ovšem uživatel vyzván k přihlášení systémem po kliknutí na úpravu akce nebo na přidání akce, je uživatel přesměrován na danou akci. Toto přesměrování je zajištěno pomocí get atributu `event`, který obsahuje buď název akce nebo v případě vytvoření nové akce znak `~`. Typ přístupu je potřebný pro dělení přístupu do systému. Pokud uživatel zadal špatné heslo nebo přihlašovací jméno, je přesměrován spolu s get atributem `val=bad` zpět na přihlášení. Tentokrát je ve formuláři nad přihlašovacím tlačítkem hláška `Wrong username or password.`, která informuje uživatele o jeho chybě. Pokud uživatel zadal správné heslo i uživatelské jméno, ale nemá validní účet, je opět přesměrován na přihlášení tentokrát však s hláškou `You don't have a Validated account.`.

5.2.6 Registrace

Pod tlačítkem pro přihlášení je možnost kliknout na registraci. Při výběru této možnosti skryje Javascript přihlašovací formulář a zobrazí registrační formulář. Ten se skládá ze dvou částí. První je na vyplnění uživatelských informací a druhá na vyplnění informací do profilu. Z registračního formuláře se dá vždy vrátit zpět na přihlášení. První informací, kterou musí uživatel vyplnit je přihlašovací jméno. To se po každém změněném znaku zkontroluje, jestli není již použito za pomocí odeslání `XMLHttpRequestu`. Tento požadavek se pošle na php skript `check-uname.php`, který se připojí na databázi a zkontroluje se, jestli dané uživatelské jméno již neexistuje. Pokud ano, odešle se v odpověď `e`, jinak zůstává odpověď prázdná. Jakmile obdrží Javascript tuto odpověď oznamující duplicitní jméno, zobrazí uživateli hned pod polem pro uživatelské jméno hlášku `This username already exists. Pleas change your username.` a znepřístupní tlačítko pro pokračování. Další položkou pro vyplnění je heslo. To musí uživatel vyplnit dvakrát kvůli kontrole možných překlepů. Javascript po dopsání druhého hesla zkontroluje, jestli se hesla shodují. Neshodují–li se hesla, je vypsána pod hesly hláška `This password is diferent. Please insert same password.` a tlačítko na pokračování je znepřístupněno. Pokud se shodují, může uživatel pokračovat v registraci kliknutím na tlačítko `Next`. Po kliknutí na toto tlačítko, Javascript chová první část registračního formuláře a zpřístupní část druhou.

Druhá část obsahuje informace pro vytvoření profilu. Uživatel zadá svoje jméno, příjmení, datum narození, oddíl a dvakrát svůj email. Javascript kontroluje shodu emailu stejně jako u hesel a také kontroluje správný formát emaiul. V případě špatného formátu emailu se uživateli ukáže hláška `This is not email. Please insert email address.`. Formát se kontroluje pouze u prvního emailu, protože druhý musí být totožný a pokud není objeví se hláška `This email is diferent. Please insert same email.` a také se znepřístupní tlačítko pro registraci. Poslední, co si musí uživatel vybrat je úroveň přístupu. Pokud uživatel zapomene vyplnit některou informaci, tak se po kliknutí na registrační tlačítko objeví u každé nevyplněné informace text `Please fill out this field.` a k registraci nedojde. Po správném vyplnění všech informací a po kliknutí na tlačítko registrace se aktivuje php skript `sing-up.php`. Ten uloží všechny informace do databáze a přesměruje uživatele na přihlašovací stránku.

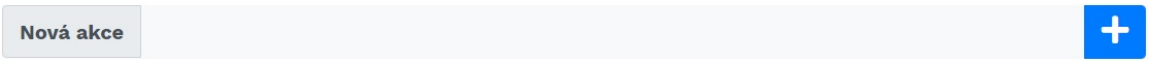
Poslední součástí je odhlášení. To probíhá po kliknutí na tlačítko odhlásit, které aktivuje php skript `logout.php`. To spustí funkci `session_destroy`, která zruší `seeeion` proměnnou a přesměruje uživatele na domovskou stránku.

5.2.7 Editor akcí

V editoru akcí lze akce editovat i vytvářet. Níže je podrobnější popis všech funkcí, které editor akcí má.

Vytvoření nové akce

Vytvoření akce začíná po kliknutí na tlačítko pro přidání nové akce, viz. obrázek 5.2.



Obrázek 5.2: **Tlačítko pro přidání nové akce**

Po kliknutí na toto tlačítko je uživatel přesunut do editoru akcí, ve kterém je zobrazen štítek se vstupy na informace o nové akci, viz. obrázek 5.3.

Obrázek 5.3: **Přidání nové akce**

Těmito informacemi jsou jméno akce a datum konání. Při změně znaku názvu akce kontroluje Javascriptová funkce `checkEvent` duplicitu názvu akcí pomocí `XMLHttpRequestu`. Stejně jako u kontroly duplicity přihlašovacích jmen je dotaz poslán php skriptu `event-check.php`, který zkontroluje existenci názvu v databázi a odešle `e` jako odpověď v případě duplicity. Jinak odešle odpověď prázdnou. Při duplicitě funkce zobrazí hlášku `This event already exists. Please change event name.` pod příslušným polem a znepřístupní tlačítko na potvrzení vytvoření akce. Pokud je vše vyplněné správně, uživatel může kliknout na potvrzovací tlačítko. Také je zde mazací tlačítko `Cancel` volající funkci `evnCancel`, které odstraní vyplněné informace. Po odkliknutí na potvrzovací tlačítko se spustí funkce `evnCreate`. Ta odešle požadavek na vytvoření nové akce php skriptu `event-create-new.php` s vepsanými informacemi jako jeho parametry.

Po korektním vytvoření nové akce se animovaně zobrazí název akce, datum konání a také prázdný obsah akce. Ten obsahuje pouze tlačítko na přidání kategorie, viz obrázek 5.4.

Obrázek 5.4: **Prázdná akce**

Uložení změn

Každá úprava v editoru se vkládá do struktury `changesBuff`, která obsahuje informace o všech provedených změnách od posledního uložení. Jedná se o řetězec ve formátu JSON. Úpravy se uloží do zásobníku vždy s názvem a parametry. Před uložením nového záznamu se také prochází celý zásobník pro případ, že v něm již záznam o dané úpravě je. Pokud tento případ nastane, starý záznam se smaže a vytvoří se nový záznam. Zamezí se tak zbytečnému plnění zásobníku neužitečnými nebo duplicitními záznamy. Pokud jsou v zásobníku dvě opačné úpravy, které se navzájem anulují, jsou obě smazány.

Přidání listu

Funkce `tableAdd` je spuštěna po kliknutí na tlačítko přidání nového listu kategorie. Při přidání nového listu, funkce nejdříve zkontroluje, jestli se jedná o první list. Jediným rozdílem u vytváření dalších listů je to, že se u nich nastavuje hodnota aktuálně používaného listu na hodnotu nového. Dále se vytvoří pomocí JQuery nová záložka se všemi potřebnými tlačítky a vytvoří se tabulka s jedním prázdným řádkem. Posledním rozdílem je, že u prvního listu se vytvoří menu, viz. obrázek 5.5, na ovládání editoru akcí. Všechny vygenerované části nejsou vidět, protože mají styl `display: none;`. Vše se zobrazuje až po dokončení vygenerování s rozestupy půl sekundy. Přidání nového listu je uloženo do zásobníku změn s názvem `addTab` a s parametry: název akce `Event` a název listu `Name`.

Obrázek 5.5: **Menu editoru akcí**

Úprava akce

V případě, že uživatel chce upravovat již vytvořenou akci, klikne na domovské stránce u akce na editovací tlačítko. Při načítání editoru se pomocí php skriptu `session.php` zkontroluje, jestli je uživatel přihlášený. Pokud ano, zkontrolují se jeho práva na úpravu dané akce. V případě, že práva na úpravu má, systém ho pustí dál. Pokud však na úpravu oprávnění nemá, je přesměrován na stránku `denied.php`, která uživatele na tuto zkutečnost upozorní. Nepřihlášený uživatel je přesměrován na stránku pro přihlášení, které se v argumentu `event` pošle název akce k úpravě. Pokud je vše v pořádku načte se funkce `loadTable`. Ta odešle požadavek na php skript `event-data.php`, který se připojí na databázi a za pomocí sql dotazů získá potřebná data, která vrátí v podobě řetězce ve formátu JSON. Ten je ve funkci rozparsován a dále je z něj vytvořena tabulka s listy kategorií. Po vytvoření tabulky se vygeneruje menu pro ovládání editoru. Stejně jako u vytváření nové akce nejsou všechny vygenerovaná části vidět do konce generování. Až po jeho dokončení se vše zobrazuje s rozestupy půl sekundy.

V tabulkách je možné vyhledávat pomocí stejné funkce, která je popsaná v kapitole 5.2.4.

Úprava listu

Každý list akce, viz. obrázek 5.6 má název, který lze upravovat. Po kliknutí na editovací tlačítko se spustí funkce `renameList`. Ta vezme název a na místo záložky vloží pole s tímto textem, za které vloží potvrzovací tlačítko. Po potvrzení změny názvu karty, se vše vrátí do původního stavu spolu s novým názvem. Dále uloží změnu názvu do zásobníku úprav. U přejmenování listu je název úpravy `renameTab` s parametry: název akce `Event`, originální název listu `Orig` a nový název listu `New`.

Obrázek 5.6: **List akce – změna názvu**

Přidávání řádků

Každý list obsahuje neměnné záhlaví a poslední needitovatelný a nesmazatelný řádek, který umožňuje přidání řádku nového, viz. obrázek 5.7. Při kliknutí na přidání nového řádku se aktivuje funkce `newRow`. Ta odstraní z posledního řádku přidávací tlačítko, nahradí ho tlačítkem pro smazání řádku a spustí ho v módu pro editaci. Dále vytvoří nový poslední řádek. Nakonec vloží záznam do zásobníku úprav s názvem `addRow` a s parametry: jméno akce `Event`, jméno listu `Category` a číslo přidaného řádku `RowNum`.

Obrázek 5.7: **Tabulka editoru akcí**

Mazání řádků

Při smazání řádku se aktivuje funkce `dataDelete`, která řádek odstraní a změní číslo řádku u všech ostatních tak, aby šly přesně po sobě. Dále se vloží do zásobníku změn informace o smazaném řádku. Název záznamu je `deleteRow`. Parametry jsou zde: název akce `Event`, jméno listu `Category`, identifikátor výsledku `IdResult` a identifikátor profilu `IdProfile`. Identifikátor výsledku jednoznačně identifikuje výsledek v databázi a identifikátor profilu určuje profil závodníka, ke kterému daný řádek patří. Pokud se jedná o neregistrovaného závodníka identifikátor profilu určuje záznam profilu v databázi, patřící danému řádku.

Seřazení řádků

Další funkcí je seřazení listu podle sloupce. Seřazení může být sestupné nebo vzestupné. Při kliknutí na tlačítko setřídit se volá funkce `sortTable`. Tato funkce nejdříve zavolá funkci `finish` a poté řádky seřadí. Po dokončení seřazování se odešle dotaz na php skript `event-sort.php` s upravenými daty. Skript tato data vezme a v databázové tabulce `results` změní atribut pořadím řádku `RowNum` u každého záznamu, u kterého bylo změněno. Nakonec se odešle odpověď s úspěchem akce Javascriptu, který změní ikonu indikace seřazení u všech sloupců na výchozí. U sloupce, podle kterého se seřazení provádělo, nastaví ikonu buď na vzestupné seřazení nebo na sestupné seřazení. Nakonec je zavolána funkce `save`, která vše uloží a zapne mód editace.

Editovací mód

Zapínání módu editace je zajištěno tlačítkem `edit`, které spouští funkci `edit`. Tato funkce změní obsah všech buněk všech tabulek akce. Výjimkou jsou sloupce `Nejlepší čas` a `Pořadí`. Jejich hodnoty jsou generovány automaticky. U každé buňky vezme funkce text a vloží jej do tagu `<input>`. Dále za něj vloží `<datalist>`, který dává uživateli nápovědu. Ta je ovšem jen u sloupců `Jméno`, `Příjmení`, `Oddíl`, `Rok narození` a `Výchozí čas`. Při každé změně znaků informace v těchto sloupcích se volá funkce `datalist`, která odešle dotaz na php skript `datalist-data.php` s dosavadním textem a typem jako parametry. Tento skript načte databázový pohled konkrétního typu informace. Poté vybere jen ty řádky, které k danému vstupu patří. Výsledek odešle zpět jako řetězec ve formátu JSON. Jakmile Javascript dostane odpověď, vloží všechny informace do datalistu, který se zobrazí uživateli.

Při opuštění buňky se spustí funkce `updateRow`. Ta nejdříve volá funkce `cas` aktualizující sloupec `Nejlepší čas`. Hned po ní je zavolána funkce `poradi`, která přepočítá pořadí závodníků podle nejlepšího času. U každého řádku pak uloží změnu času i pořadí do zásobníku změn pomocí funkce `updateBestOrd`. Ta do zásobníku vloží úpravy s názvem `updateBestOrd` obsahující parametry: název akce`Event`, jméno listu `Category`, číslo řádku `RowNum`, identifikátoru výsledku `IdResult`, nejlepší čas `Best` a pořadí `Ord`. Poté funkce `updateRow` uloží do zásobníku změn upravenou informaci buňky. Název úpravy je `updateRow` a má parametry obsahující všechny informace řádku, kromě nejlepšího času a pořadí. Ty jsou již v zásobníku vloženy.

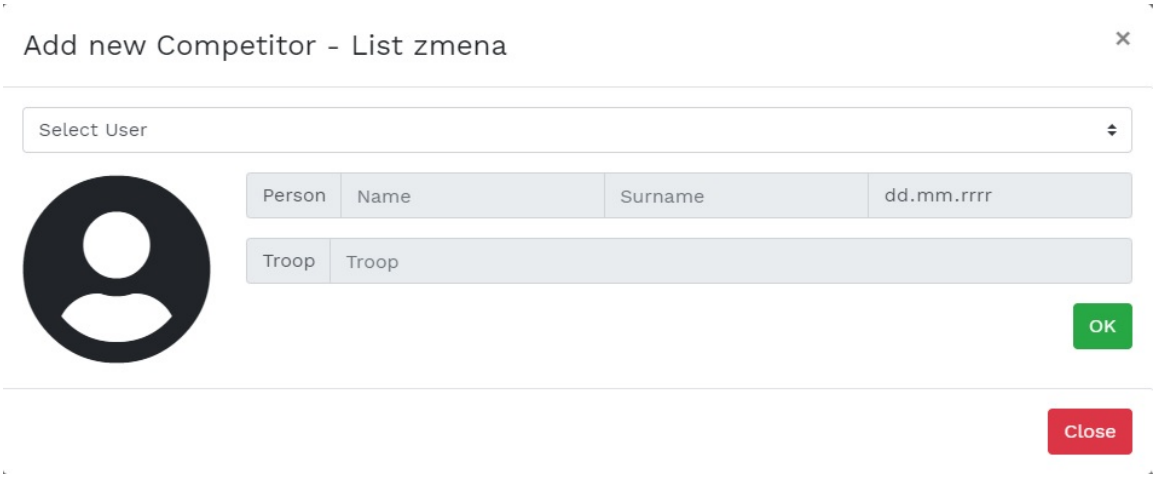
Funkce dokončení a uložení

Pro vypnutí módu editace je tlačítko dokončení. Toto tlačítko spouští funkci `finish`, která odebere ze všech buněk všech listů `<input>` a vloží místo nich jejich obsah. Dále tato funkce pošle data ze zásobníku úprav php skriptu `event-save-data.php`, který je uloží do databáze. Tento skript může v odpovědi poslat úpravu některých identifikátorů řádků. Pokud se tak stane, Javascript odpověď zpracuje a upraví příslušné hodnoty řádků. Když je vše zpracované, je zásobník změn vyprázdněn.

Tlačítko pro ukládání volá funkci `save`, která je téměř totožná s výše popisovanou funkcí `finish`. Jediným rozdílem je, že funkce `save` nevypíná mód editace.

Přidání závodníka

Dalším tlačítkem je přidání závodníka. Tím je myšleno již registrovaného závodníka. Stisknutí tlačítka volá funkci `competitor`. Tato funkce otevře modální okno, viz. obrázek 5.8. Dále je odeslán požadavek na php skript `settings-users.php` s prázdnými parametry `event`, `add` a s parametrem `valid=1`, který říká, že chceme pouze uživatele s validním účtem. Skript získá z databáze, pomocí sql příkazu, seznam obsahující jméno, identifikátor profilu a typ účtu všech validních uživatelů. Následně jej pošle zpět Javascriptu, který seznam aplikuje na možnost výběru závodníka. Výběr obsahuje uživatelské jméno a identifikátor profilu, který není zobrazen. Při změně výběru závodníka se zavolá funkce `addCompetitorChange`, která pošle požadavek na php skript `profile-data.php` s parametrem obsahujícím uživatelské jméno. Teto skript najde v databázi profil odpovídající uživateli a odešle informace zpět. Javascript poté odpověď zpracuje a informace vloží do náhledu profilu. Po kliknutí na potvrzovací tlačítko se zavolá funkce `addCompetitor`. Ta v aktuálním listu vytvoří nový řádek na jeho konci a vloží do něj data o závodníkovi. Vytvoření řádku je totožné jako u přidávání nového řádku. Nakonec se tato změna vloží do zásobníku úprav s názvem `addCompetitor`. Tato úprava má následující parametry: jméno akce `Event`, název listu `Category`, číslo řádku `RowNum` a identifikátor profilu `IdProfile`.



Obrázek 5.8: **Modální okno pro přidání závodníka**

Připojení a odpojení časomíry

Tlačítko pro připojení a odpojení integrované časomíry volá funkci `autoFillFunc`, která mezi těmito stavy přepíná. Pro připojování časomíry k aktuálnímu listu slouží funkce `autoFillOn`. Ta odešle informaci o připojení časomíry php skriptu `event-autofill.php` s parametry: název akce pro připojení `event`, připojení časomíry `autofill=1` a název listu `category`. Skript uloží do databáze informaci o připojení a odpoví zpět podle jeho úspěchu. Pokud Javascritp dostane v odpovědi úspěch, upraví ikonu signalizující stav připojení. Dále nastaví funkci `runListen` na periodicky se opakující s periodou jedné sekundy. Funkce `runListen` se při spuštění dotáže php skriptu `event-auto-fill.php` s parametry: `event` a `category`, jestli nejsou změny ve výsledcích. Skript si pamatuje poslední poslaná data, která má uložena v proměnné `session`. Při každém dotazu skript získá z databáze aktuální data a porovná je s uloženými daty. Pokud se shodují odesílá se prázdný řetězec, jinak se posílají jen data, která se změnila. Jakmile Javascript získá odpověď jinou než prázdný řetězec, vloží data do příslušné buňky. Dále se spouští po každém vložení data funkce: `cas`, `poradi` a `updateBestOrd`. Nakonec se všechny změny uloží.

Při odpojení časomíry se volá funkce `autoFillOff`. Ta volá, stejně jako funkce `autoFillOn`, php skript `event-autofill.php` se stejnými parametry až na parametr `autofill`. Ten je tentokrát roven nule. Skript opět provede změny a vrátí úspěch akce jako odpověď. Nakonec funkce zastaví opakování funkce `runListen`.

Při vrácení neúspěchu připojení i odpojení časomíry je uživateli zobrazeno modální okno s bližšími informacemi.

Zapnutí a vypnutí streamu výsledků

Pro zapnutí a vypnutí streamu výsledků jsou dvě tlačítka. První, na zapnutí streamu, volá funkci `play`. Ze všeho nejdříve tato funkce zjistí, zda–li není stream zapnutý u jiné záložky, než u které jej chceme zapnout nyní. Pokud takovou záložku najde, pošle dotaz na php skript `event-stream.php` s parametry: jméno akce `event`, jméno listu `category` a vypnutí streamu `stream=0`. Skript uloží tuto změnu do databáze a vrátí úspěch v odpovědi. Dále Javascritp změní ikonu indikující stav streamu této záložky a pošle stejnému php skriptu dotaz na zapnutí streamu. Ten se liší pouze hodnotou parametru `categori` obsahující název listu a zapnutí streamu `stream=1`. Po obdržení odpovědi Javascript změní status streamování u aktuálního listu.

U vypnutí streamu se vyžívá funkce `stop`, která vypne stream stejným postupem jako je popsaný výše. Pouze nepokračuje se zapínáním na další list.

Při vrácení neúspěchu zapnutí i vypnutí je stejně jako u připojení a odpojení časomíry uživateli zobrazeno modální okno s bližšími informacemi.

Export a tisk

Data uložená na jednotlivých listech lze stáhnout a tisknout. Uživatel má dvě možnosti stažení vždy aktuálně používaného listu. První možností je stažení ve formátu pdf a tou druhou je stažení ve formátu xlsx. Tlačítko pro stažení v pdf volá funkci `downloadPDF`. Tato funkce je totožná s funkcí `downloadPDF` v kapitole 5.2.4 a taktéž využívá knihovny `pdfmake` [2]. Název souboru je složen z názvu listu a přípony `-výsledky.pdf`. U stažení ve formátu xlsx se volá funkce `downloadXLSX`. Zde se využívá knihovna `SheetJS js-xlsx` [4]. Soubor stáhne se stejným názvem jako u formátu pdf. Liší se pouze příponou. U tisku se

volá funkce `print`. Stejně jako v kapitole 5.2.4, využívá knihovny `pdfmake` [2] a vytvoří pdf soubor, který se otevře pro tisk.

Ostatní funkce

Tlačítko pro otevření aktuálního streamu výsledků volá funkci `view`, která otevře stránku streamu akce v novém okně.

Stejným principem funguje i tlačítko pro otevření stránky s výsledky akce. To volá funkci `results`, která na novém kartě otevře stránku s výsledky.

Předposlední funkcí editoru akcí je vytvoření QR kódu, který na danou akci odkazuje. Pro generování těchto kódů jsem využil služby Googlu [3]. Při kliknutí na tlačítko vygenerování QR kódu se volá funkce `qrCode`, která otevře modální okno s vygenerovaným obrázkem. Je zde možnost tento kód vytisknout stisknutím tlačítka `print`. Toto tlačítko volá funkci `printQR`, která otevře novou stránku `qrcode.php` obsahující název akce a QR kód. Stránka je připravena k tisku.

Poslední funkcí je tisk diplomů. Při stisknutí tlačítka se volá funkce `diplomas`, která otevře modální okno s výběrem diplomů. Při otevření okna se posílá dotaz na php skript `deditor-list.php`. Jako odpověď zašle skript seznam diplomů získaný z databáze. Javascript následně tento seznam vloží do výběru pro uživatele. Jakmile uživatel potvrdí výběr diplomu tlačítkem `OK`, je zavolána funkce `diplomasOK`, která otevře novou katru se stránkou pro tisk diplomů. Této stránce se věnuje kapitola 5.2.9.

5.2.8 Editor diplomů

Při načtení editoru diplomů se uživateli zobrazí nabídka, viz. obrázek 5.9, obsahující otevření a úpravu již existujícího diplomu, možnost nahrát diplom a vytvoření nového diplomu.

Obrázek 5.9: **Nabídka**

Načtení diplomu

Po kliknutí na tlačítko pro úpravu existujícího diplomu se zavolá funkce `openFile`, která pošle dotaz na php skript `deditor-list.php`. Ten vrátí Javascriptu seznam diplomů z databáze, který je po zpracování zobrazen uživateli v modálním okně. Potvrzením výběru diplomu se volá funkce `loadFile`, která schová nabídku a zobrazí menu pro editaci diplomu. Následně funkce pošle dotaz na php skript `deditor-data.php`, který získá z databáze veškerá data diplomu. Data obsahují pozadí diplomu, jeho typ a strukturu diplomu. Skript vše pošle v odpovědi zpět Javascriptu, který je zpracuje a vloží do náhledu diplomu. Náhled se následně zobrazí ve formátu A4.

Nahrání diplomu

Pro nahrání diplomu se opět otevře modální okno, které je téměř totožné s oknem pro načtení diplomu. Jediným rozdílem je, že v okně není nabídka diplomů, na místo ní je totiž

pole pro vložení souboru. Jakmile uživatel nahraje soubor ve specifickém formátu `.dip`, zavolá se funkce `uploaddFile`, která zpracuje data diplomu a vloží je do náhledu stejně jako u načítání diplomu. Název diplomu se bere z názvu souboru.

Vytvoření diplomu

Při vytváření nového diplomu je uživateli zobrazeno pole pro zadání jména diplomu. Po každé změně znaku se kontroluje, jestli není název duplicitní s jiným již vytvořeným diplomem. Kontrola probíhá stejně jako v kapitole 5.2.7. Jedinou změnou je php skript, který je k tomu použit. Zde je použit skript `deditor-check.php`. Při duplicitním jméně se zobrazí hláška `This diploma already exists. Please change diploma name.`. Potvrzení jména diplomu se volá funkce `fileCreate`, která vytvoří prázdný náhled diplomu a zobrazí editovací menu.

Editace diplomu

Po vytvoření nebo nahrání diplomu je zobrazeno editovací menu diplomu. Zde má uživatel na výběr několik funkcí.

První je vložení obrázku jako pozadí diplomu. Při kliknutí na toto tlačítko se otevře modální okno s polem pro nahrání souboru. Po zvolení obrázku se zavolá funkce `readURL`, která obrázek promítne přímo do náhledu.

Druhé tlačítko přidává další textová pole do diplomu. Toto má na starosti funkce `addField`.

Další tlačítko spouští pomocí funkce `editModeOn` editovací mód, který změní všechna pole na editovatelná a také zobrazí menu pro editaci textu.

Následující tlačítko naopak editovací mód vypíná. Vypnutí volá funkci `editModeOff`, která všechna pole změní zpět na needitovatelná a schová menu pro editaci textu.

Posunování a pozicování textových polí je možné pouze s vypnutým módem editace textu. Pro posun polí se využívá několik funkcí. Jednou je `mouseClick`, která zaznamenává, jestli bylo kliknuto na textové pole. Další funkcí je `mouseMove`, která, pokud je kliknuto na některé textové pole, sleduje pohyb myši a přepočítává umístění chyceného pole. Pohyb je omezen pouze na náhled diplomu a není možné pole umístit mimo něj. S poli lze pohybovat i pomocí šipek. U tohoto pohybu je pole posunuto vždy o pět pixelů. Pohyb polí se opět koná až po kliknutí na určité textové pole. Tento pohyb má na starosti funkce naslouchání stisknutí kláves v okně prohlížeče. Jedná so o JQuery funkci `$(document).keydown(function(e){});`. Tato funkce také naslouchá stisknutí klávesy `delete`, která pole odstraní.

Editace textu

Editování textů polí obsahuje několik základních formátovacích stylů. Tlačítka pro: tučný text, text kurzívou, podtržení textu, přeškrtnutí textu, pro zarovnání textu a pro posunutí odstavce volají funkci `execCmd`, do které se dává parametr úpravy textu. Tato funkce upraví text požadovanou operací. U tlačítka pro: změnu fontu písma, změnu velikosti písma a změnu barvy písma, se volá funkce `execCmdWArg`, která opět provede požadovanou operaci.

Editace textu také uživateli nabízí možnost přidat nahraditelný obsah. Jedná se o obsah, který bude při tisku diplomů nahrazen konkrétními hodnotami. Tímto obsahem může být pouze: jméno závodníka `[Name]`, příjmení závodníka `[Surname]`, oddíl závodníka `[Troop]`, nejlepší čas závodníka `[Time]` a umístění závodníka `[Order]`. Tento obsah musí být vždy

ohraničen hranatými závorkami. Jedná se o vnitřní reprezentaci danou formátem diplomu. Tento formát jsem sám navrhl. Nahraditelný obsah může uživatel vepsat do textu ručně nebo ho může, po vybrání z nabídky, vložit pomocí tlačítka přidat. Toto tlačítko volá funkci `addFunc`, která danou možnost vloží na konec vybraného textového pole.

Uložení a export diplomu

Po dokončení práce na úpravách diplomu má uživatel k dispozici tři možnosti. První možností je diplom uložit. Uložení volá funkci `saveFile`, která převede diplom na vnitřní reprezentaci a odešle je php skriptu `deditor-save.php`. Ten všechna data vloží do databáze a odpoví úspěchem operace. Pokud dostane Javascript odpověď `OK`, je uživateli zobrazeno modální okno s textem `Changes saved`. V případě chyby je uživateli zobrazen text `Changes NOT saved!! Please repeat the action later.`.

Druhou možností je stažení diplomu. Pokud si uživatel vybere tuto možnost, spustí se funkce `downloadFile`, která převede diplom do stejné struktury jako při ukládání. Následně vloží data do souboru s názvem složeným ze jména diplomu a přípony `.dip`. Nakonec soubor stáhne.

Poslední možností je změny diplomu neukládat a pomocí zavíracího tlačítka editor bez uložení zavřít. Při zavření se volá funkce `closeFile`, která vrátí editor do původního stavu a vrátí uživateli menu pro načtení diplomů.

5.2.9 Tisk diplomů

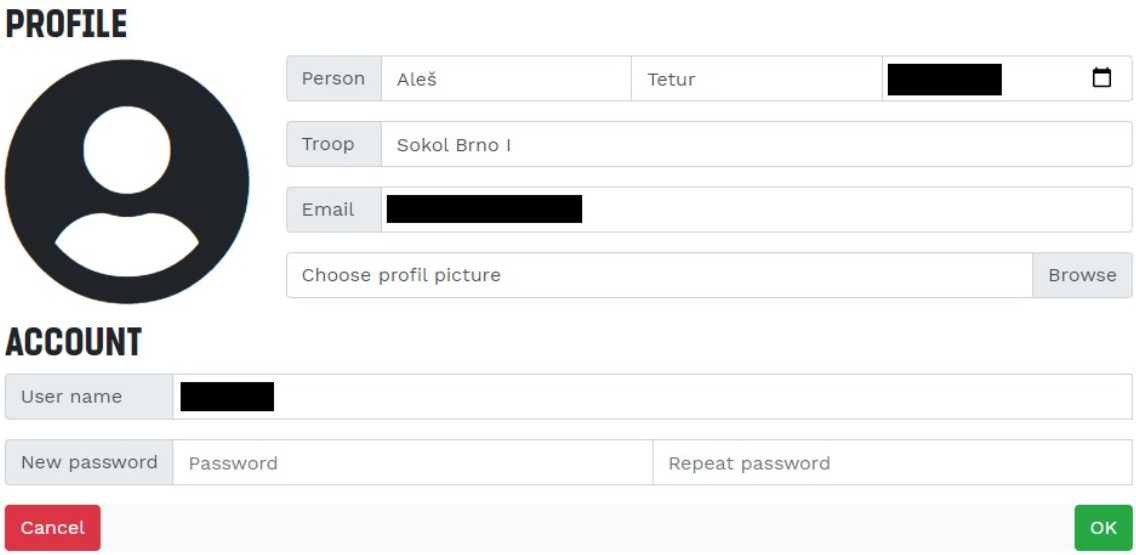
Na stránku pro tisk diplomů se uživatel dostane z editoru akcí po výběru diplomu, jak je popsáno v kapitole 5.2.7. Při načítání stránky je volána funkce `loadPage`, která zobrazí název akce, pro kterou jsou diplomy vygenerovány. Následně je volána funkce `loadDiplomas`. Ta pošle dotaz na php skript `deditor-data.php`, který je popsán v kapitole 5.2.8. Javascript po obdržení odpovědi data zpracuje a vytvoří náhled diplomu. Také vytvoří nabídku kategorií akce, ze které se uživatel vybere. Po vybrání kategorie se volá funkce `changeCategory`, která změní nápis počtu diplomů a následně zavolá funkci `setDiplom`. Tato funkce vygeneruje první diplom a nahradí zástupné objekty konkrétními hodnotami. Uživatel se může pohybovat mezi jednotlivými diplomy pomocí tlačítek šipek. Je zde možnost zobrazit další diplom v pořadí, předešlý diplom, první diplom a poslední diplom. Tento pohyb mají na starosti funkce `nextDiplom`, `prevDiplom`, `firstDiplom` a `lastDiplom`, které nastaví index zobrazeného diplomu a pomocí funkce `setDiplom` jej přegeneruje.

Pro tisk diplomů jsou zde dvě tlačítka. První volá funkci `printDiplom`, která vytvoří pdf soubor pro tisk za pomocí knihovny pdfmake [2]. Jedná se o tisk aktuálně načteného diplomu. Druhé volá funkci `printDiplomas`. Tato funkce využívá také knihovny pdfmake [2]. Tentokrát vytvoří pdf soubor se všemi diplomy vybrané kategorie a otevře jej na nové kartě k tisku. Po stisknutí kláves `ctrl+p`, se spustí funkce `printDiplomas`.

5.2.10 Stránka profilu

Na stránce profilu má registrovaný uživatel možnost upravit informace na jeho profilu, viz. obrázek 5.10. Při načítání stránky se volá funkce `getData`, která pošle dotaz na php skript `profile-data.php`. Ten vezme data profilu uživatele z databáze a pošle je jako odpověď. Javascript po získání odpovědi volá funkci `setData`, která přijatá data zobrazí. Uživatel může informace libovolně upravovat a po dokončení úprav je uložit. Při ukládání se volá funkce `updateData`, která pošle data php skriptu `profile-update.php`. Ten data

uloží do databáze a odpoví úspěšností akce. Pokud jsou data uložena správně zobrazí se uživateli modální okno s textem `Changes saved`, pokud nejsou data uložena textem v okně je `Changes NOT saved!! Pleas reapit action later.`. U změny uživatelského jména nebo hesla probíhá stejná kontrola jako u registrace. Ta je popsána v kapitole 5.2.6.



Obrázek 5.10: **Profil uživatele**

5.2.11 Nastavení

Nastavení je u každé úrovně přístupu do systému jiné. Níže budu popisovat nastavení úrovně administrátora a na konci kapitoly napíši co u dalších vrstev přístupu chybí. Celkem jsou tři úrovně přístupu, pokud nepočítám neregistrované uživatele, kteří do nastavení nemají přístup. Specifikace jednotlivých vrstev jsou v kapitole 2.2.3. Informace jsou v nastavení rozděleny do jednotlivých listů.

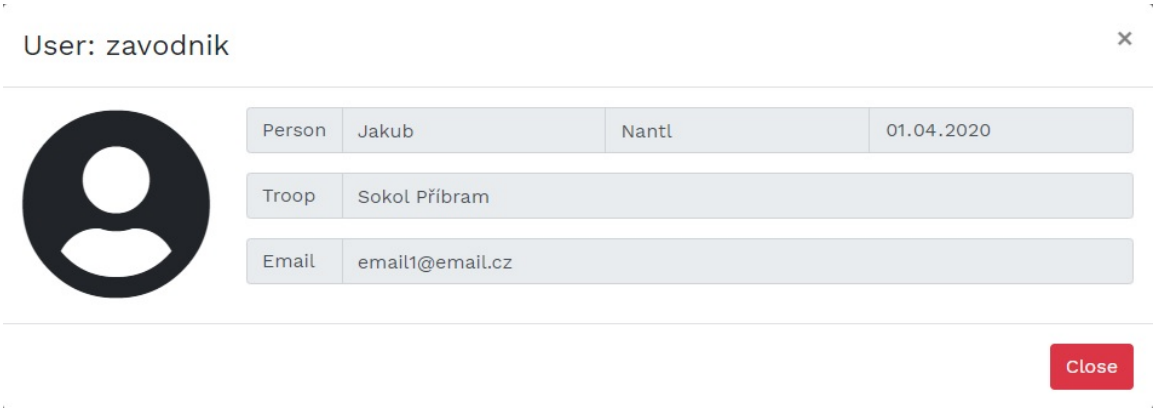
Načtení obsahu

Při načítání stránky se volá funkce `getData`, která odešle požadavek php skriptu `settings-list.php`. Tento skript získá seznam akcí, ke kterým má uživatel přístup, z databáze a pošle jej zpět jako odpověď. Při inicializaci stránky se následně zavolá funkce `tabsCreate`, která vytvoří jednotlivé listy. Jakmile Javascript dostane odpověď od php skriptu, volá funkci `setData`, která pošle dotaz na php skript `event-list.php`. Ten pošle seznam všech akcí zpět Javascriptu. Následně vloží interpretaci seznamu do listu `Competitions`. Dále vloží interpretaci seznamu s akcemi, ke kterým má uživatel přístup, do listu `Edit access`.

Nakonec se volá funkce `userList`, která odešle dotaz php skriptu `settings-users.php` s parametrem `valid=1`. Ten z databáze zjistí seznam všech uživatelů s validním účtem a pošle ho zpět jako odpověď. Jakmile ji Javascript dostane, vloží data do jednotlivých listů uživatelů, podle jejich přístupu a celý seznam vloží i do listu `Valid users`. Dále se u administrátorské úrovně volá funkce `settings-users.php` znovu, tentokrát s parametrem `valid=0`. Nyní skript z databáze získá seznam všech uživatelů, kteří validní účet nemají a pošle jej zpět. Javascript poté vloží interpretaci seznamu do listu `Non-valid users`. Tím je vygenerován celý obsah stránky nastavení.

Seznamy uživatelů

První tři listy obsahují seznamy uživatelů jednotlivých úrovní přístupu do systému. Uživatelé mají možnost podívat se na jejich profily. Pro zobrazení profilu se volá funkce `user`, která pošle dotaz na php skript s parametrem uživatelského jména profilu `uname`. Skript zjistí z databáze informace profilu a pošle je zpět. Javascript následně odpověď zpracuje a data vloží do modálního okna. Nakonec je modální okno uživateli zobrazeno. Toto okno data pouze zobrazuje a nedají se upravovat, viz. obrázek 5.11.



Obrázek 5.11: **Modální okno pro zobrazení profilu**

Další možnost je úprava profilů. U ní se volá funkce `userEditModal`, která provede všechny kroky jako funkce `user`. Jediným rozdílem je, že data profilu jsou vložena do modálního okna pro úpravu profilů, které obsahuje editovatelná pole a potvrzovací tlačítko. Toto okno vypadá stejně jako stránka profilu, viz. obrázek 5.10. Po potvrzení změn v profilu je volána funkce `userEdit`, která pošle data profilu php skriptu `profile-update.php`. Funkce tohoto skriptu je popsána v kapitole 5.2.10. Nakonec se zobrazí modální okno s informacemi o uložení.

Přihlašování a odhlašování na akci

Přihlásit se lze pouze na akce, které ještě neproběhly. Přihlašování je v listu `Competitions`, která slouží také jako seznam všech akcí. U akce je možné se přihlásit do konkrétní kategorie i odhlásit se z ní. Pro přihlášení se volá funkce `singEvnModal`, která pošle dotaz php skriptu `settings-categories.php` s parametrem jména akce `event`. Ten zjistí seznam kategorií dané akce a vrátí ji. Javascript následně seznam zpracuje a vloží jej do modálního okna. Jakmile uživatel potvrdí přihlášení do kategorie, spustí se funkce `singEvn`. Tato funkce odešle php skriptu `settings-sing-up.php` dotaz s parametry: jméno akce `event` a jméno kategorie `category`. Php skript pak do databáze uloží nové informace a vrátí odpověď. Jsou tři možnosti odpovědi. První je úspěch, při kterém se zobrazí modální okno s textem `Changes saved`. Druhou možností je skutečnost, že uživatel je do kategorie již přihlášen. V tomto případě se otevře modální okno s textem `You are already sing-in this event!`. Poslední možností je chyba, u které se v modálním okně objeví text `Changes NOT saved!! Please repeat the action later.`.

U odhlašování z kategorie se volá funkce `logoutEvnModal`, která tentokrát odešle požadavek na php skript `settings-categories-log.php` s parametrem názvu akce `event`. Tento skript vrátí v odpovědi seznam kategorií, na které je uživatel přihlášen. Seznam je

uživateli zobrazen v modálním okně. Po potvrzení odhlášení z kategorie se posílá požadavek php skriptu `settings-log-out.php` s parametry: jméno akce `event` a jméno kategorie `category`. Skript dále provede změny v databázi a odešle odpověď. Javascript interpretuje odpověď stejným způsobem jako u přihlašování do kategorie výše.

U akcí, které již proběhly je možné se podívat na jejich výsledky, pomocí volání funkce `resultEvn`, která otevře výsledky na nové kartě prohlížeče.

Úprava práv editace akce

Seznam všech akcí je v listu `Edit competitions`. Zde se dá akce smazat a přidat nebo odebrat oprávnění na úpravu akce. Pro smazání akce se volá funkce `deleteEvn`, která pošle požadavek php skriptu `settings-delete.php`. Tento skript smaže informace o akci z databáze.

Pro odebrání práv na úpravu akce se volá funkce `removeUserModal`. Tato funkce otevře modální okno pro odebrání práv uživateli a zavolá funkci `userList` s parametrem název akce `event`. Bližší popis funkce `userList` viz. 5.2.11. Tato funkce vloží seznam uživatelů s právy na úpravu do modálního okna. Po potvrzení uživatele je volána funkce `removeUser`, která pošle požadavek php skriptu `settings-remove.php` a parametry: jméno akce `event` a uživatel `uname`. Skript odstraní informace z databáze a pošle odpověď. Javascript zobrazí modální okno podle odpovědi stejně jako v kapitole 5.2.11.

Pro přidání práv na úpravu akce se volá funkce `addUserModal`. Tato funkce otevře modální okno pro přidání práv uživateli a zavolá funkci `userList` s parametrem název akce `event`. Bližší popis funkce `userList` viz. 5.2.11. Tato funkce vloží seznam uživatelů, kteří práva na úpravu do modálního okna nemají. Po potvrzení uživatele je volána funkce `addUser`, která pošle požadavek php skriptu `settings-add.php` a parametry: jméno akce `event` a uživatel `uname`. Skript přidá informace do databáze a pošle odpověď. Javascript zobrazí modální okno podle odpovědi stejně jako v kapitole 5.2.11.

Validace uživatelů

Administrátor může validovat a unvalidovat všechny uživatele, kromě administrátorů. Seznam nevalidních uživatelů je v listu `Non-valid users`, kde je tlačítko umožňující validaci uživatele. Při stisknutí tohoto tlačítka se volá funkce `validUser`, která pošle požadavek php skriptu `settings-validation.php` a parametry: uživatelské jméno `uname` a validace `val=1`. Skript uloží tuto změnu do databáze. Následně přesune Javascript uživatele z listu `Non-valid users` do listu `Valid users`.

Spolu s validací má administrátor nevalidní účet smazat. K tomu se volá funkce `deleteUser`, která pošle php skriptu `settings-delete.php` parametr uživatelské jméno `uname` ke smazání. Tento skript odstraní uživatele z databáze. Javascript nakonec odstraní uživatele ze seznamu `Non-valid users`.

V listu `Valid users` je seznam všech validních účtů. Pro zrušení validace účtu uživatele je volána funkce `validUsers`, která pošle požadavek php skriptu `settings-validation.php` s parametry: uživatelské jméno `uname` a validace `val=0`. Skript opět uloží změnu do databáze a Javascript přesune uživatele z listu `Valid users` do listu `Non-valid users`.

Rozdíly v přístupech

U úrovně oprávnění typu pořadatel se pořadateli nezobrazuje možnost úpravy profilů. Také nemá zobrazené listy pro validaci účtů.

Úroveň určená pro závodníky nezobrazuje listy pro validaci účtů ani list pro přidělování práv editace akcí. Stejně jako u úrovně pro pořadatele se nezobrazuje možnost úpravy profilů.

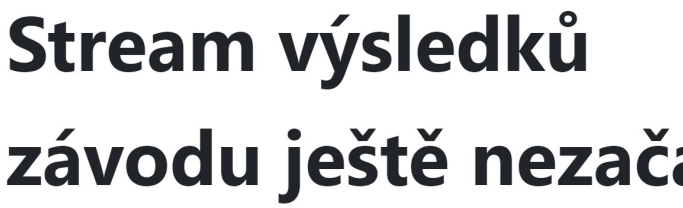
5.2.12 Statistiky

Na této stránce nalezne uživatel svoje závodní statistiky. Při načítání stránky se volá funkce `tabsCreate`, která vytvoří čtyři záložky pro grafy. Následně se volá funkce `setMyChar`, která vkládá prázdné grafy do jednotlivých záložek. Další volanou funkcí je `getMyData`. Ta pošle požadavek na php skript `statistics-my.php`, který z databáze vybere všechna potřebná data pro vytvoření grafů. Jakmile Javascript získá tato data, tak pomocí knihovny Chart.js [1] grafy vyplní. Grafy obsahují informace popsané v kapitole 4.7.1.

5.2.13 Streamování výsledků

Stránka pro streamování výsledků se vždy otevře v novém okně. Při načítání volá funkci `readFile`, která pošle dotaz php skriptu `view-data.php` s parametry: název akce streamu `event` a jedná–li se o inicializaci `init`. Php skript nejdříve z databáze zjistí, je–li stream u akce zapnutý. Pokud není odpoví zprávou obsahující `StreamOff`, ovšem pokud je stream zapnutý odešle `StreamOn` spolu s daty pro zobrazení. Data, která skript poslal si uloží do proměnné `session`. Při dalších dotazech se prvně podívá, jestli jsou data změněna. Data se vždy posílají až po jejich změně, kvůli optimalizaci.

Jakmile Javascript obdrží odpověď zpracuje ji. Pokud je stream vypnutý volá se funkce `streamNo`, která informuje pozorovatele o této skutečnosti, viz. obrázek 5.12.



Obrázek 5.12: **Vypnutý stream**

Pokud je však stream zapnutý volá se funkce `processData`, která příchozí data zpracuje a zobrazí. Vzhled streamu ukazuje obrázek 5.13.

Stream zobrazuje závodníka, který je na řadě s časem nula a v tabulce pod ním až pět následujících závodníků. Tabulka obsahuje jejich celé jméno, oddíl, rok narození a jejich nejlepší dosažený čas v závodě. Čas aktuálního závodníka se po jeho obdržení zobrazí. Vedle tohoto času se zobrazí i šipka, která ukazuje jestli, svůj nejlepší čas závodník překonal či nikoliv. Hodnoty časů, které mají speciální význam se přeloží do textové podoby. Jakmile po změně závodníkova času uplyne sedm a půl sekundy, změní se pohled. V něm je tentokrát jméno dalšího závodníka s nulovým časem.

5.3 Časomíra

V této kapitole se zaměřuji na mé řešení elektronické časomíry a její integraci do systému. Časomíra běží na operačním systému `Debian`, který je určený pro mnou vybraný hardware. Pro zobrazení grafického rozhraní jsem vybral program `Chromium` od společnosti Google.

Obrázek 5.13: **Zapnutý stream**

Důvodem výběru je možnost otevření prohlížeče při bootování systému v režimu `koisk`. Tento režim otevře prohlížeč v režimu plné obrazovky a neumožňuje uživateli dostat se mimo něj.

5.3.1 Server

Na časomíře běží vlastní server, který získává informace od operačního systému. Rozhodl jsem se pro server `Appache 2`. Dále je na časomíře nainstalované `PHP 7.4`. Tyto programy jsem si zvolil, protože v nich již dlouhou dobu pracuji, a tudíž je ovládám.

5.3.2 Kostra stránek

Stejně jako u informačního systému jsou všechny stránky generované dynamicky. Jedná se o část se statusy a modální okno pro připojení k Wi–Fi. Také se na všech stránkách shoduje tag `<heah>`, obsahující potřebná data pro fungování stránek.

5.3.3 Statusy

Tato část stránky obsahuje veškeré potřebné informace, pro zjednodušení ovládání časomíry, viz. obrázek 5.14. Při načtení stránky se vždy volá funkce `load` obsažená v Javascriptu k dané stránce, která volá funkci `stats`. Tato funkce při inicializaci systému zjistí všechny jeho statusy, které poté uloží do souboru cookies. Pro ukládání do souboru jsem se rozhodl, kvůli jednoduché práci se změnami některých stavů. Uložení probíhá pomocí volání funkce `setGetCookies`, která soubory cookies nainicializuje, pokud nejsou již nastaveny, a vrátí zpět pole s hodnotami stavů. Pokud jsou stavy již v souborech cookies uloženy, vezmou se při načítání stránky z nich.

Obrázek 5.14: **Statusy časomíry**

Připojení do systému

Prvním statusem je stav připojení časomíry k systému. Ten je vždy při inicializaci nastaven na odpojeno `Offline`. Změna tohoto stavu je popsána v kapitole 5.3.6. Tento stav je uložen v cookies souboru `evn-conn`.

Módy

Druhým stavem je mód časomíry. Časomíra má totiž dva módy. První je normální, u kterého se spouští stopování času hned při zaznění startovacího signálu. Ten druhý je experimentální. Také stopuje čas po zaznění startovacího signálu ovšem čas začíná běžet až po puštění spodního čidla. Výchozím módem je vždy ten první. Po kliknutí na status módu se spustí funkce `modeChange`, která status změní a uloží do souboru cookies s názvem `mode`.

Čidla

Třetí a čtvrtý status indikuje připojení čidel. Výchozím stavem je odpojení obou čidel. Při inicializaci se spouští naslouchání streamu php skriptu `switches-stat.php`, který při změně stavu čidel odešle zprávu s touto změnou. Javascript provede vždy po obdržení změny potřebnou rutinu, ve které také změní cookies soubory obou stavů. U spodního čidla jde o soubor `switch-1` a u druhého o `switch-2`.

Hlasitost

Další stav je hlasitost zvuků. Při inicializaci se volá funkce `soundSet`, která pošle dotaz php skriptu `sound-stat.php`. Ten pomocí příkazu na operační systém zjistí, jestli jsou zvuky zapnuty nebo vypnuty. Javascript dále tuto skutečnost uloží do souboru `sound`. Také se při inicializaci zvuku volá funkce `soundVolume`. To tentokrát pomocí skriptu `sound-volume.php` zjistí úroveň hlasitosti, která je následně uložena do souboru `volume`. Při kliknutí na ikonu tohoto stavu se zobrazí ovládací prvek hlasitosti, pomocí kterého je možné upravovat úroveň hlasitosti nebo ztišení zvuků. Pro ztlumení nebo zapnutí zvuku se volá funkce `sound`, která volá skript `sound.php` s parametrem `sound` určujícím konkrétní akci. Následně Javascript změní status zvuku. U nastavování hlasitosti se spouští funkce `modalVolumeSet`, která pošle požadovanou změnu skriptu `sound-volume.php` s parametrem konkrétní hodnoty. Ten provede změny v operačním systému. Javascript následně také změní úroveň hlasitosti.

Wi–Fi

Posledním stavem je stav připojení k Wi–Fi. Výchozím stavem je stav odpojeno. Při inicializaci se spustí naslouchání streamu php skriptu `wifi-stat.php`. Ten při každé změně týkající se připojení odešle zprávu. V této zprávě je informace o názvu Wi–Fi sítě a síly připojení k této síly. Jakmile Javascript dostane tyto informace, hned je zpracuje a zobrazí uživateli. Také tyto informace uloží do cookies souborů: pro jméno sítě `wifi-name`, pro sílu připojení `wifi`. Po kliknutí na ikonu tohoto připojení se voláním funkce `modalWifi` otevře modální okno s dostupnými sítěmi a možností se k nim přihlásit. Tyto informace jsou získány od skriptu `wifi-info.php`, který naskenuje dostupné sítě v okolí. Jelikož toto skenování trvá v řádu několika sekund, je uživateli na stránce zobrazeno načítaní do té doby, než Javascript dostane odpověď od php skriptu. Jakmile si uživatel vybere síť a zadá heslo, potvrdí připojení. Toto potvrzení zavolá funkci `connectWifi`, která pošle zadané informace skriptu `wifi-conn.php`. Tento skript změní Wi–Fi připojení a následně Javascript

změní status připojení na odpojeno. Tento stav je následně změněn již běžícím streamem `wifi-stat.php`.

Další stavy

Další stavy nejsou vidět v tomto řádku, ale jsou inicializovány pro další použití. Jedná se o stav uchovávající informaci o aktuální verzi časomíry. U její inicializace se volá funkce `versoinSet`, která za pomocí skriptu `version-stat.php` zjistí tuto informaci, kterou Javascript uloží do cookies souboru `version`. Dále se inicializuje soubor `update` na hodnotu `None`. Poslední informací stavu je název akce, ke které je časomíra připojena. Výchozí hodnotou cookies souboru je `evn-name=None`.

5.3.4 Update časomíry

Při připojení Wi–Fi se volá funkce `getUpdate`, která pošle dotaz na php skript `get-update`, který je umístěn na serveru informačního systému. Tento skript zjistí poslední verzi updatu a pošle ji zpět. Jakmile časomíra dostane tuto odpověď, uloží číslo nejnovější verze do souboru `update`. Také zobrazí ikonu indikující nutnost aktualizace. Jakmile na ní uživatel klikne, spustí se funkce `update`, která ze vzdáleného serveru stáhne aktualizaci ve formátu zip. Následně funkce `updateSave` uloží soubor do časomíry za pomocí skriptu `udate-save.php`. Hned po uložení se volá funkce `updateInstall`, která aktualizace nainstaluje zavoláním skriptu `update-install.php`. Po dokončení instalace se volá funkce `updateRestart`, která zapne odpočítávání restartu časomíry. Restart provádí php skript `update-restart.php`. Po celou dobu updatování časomíry je uživatel informován o postupu pomocí modálního okna.

5.3.5 Domovská stránka

Domovská stránka obsahuje, kromě stavů, celkem šest tlačítek. Každé přesměruje uživatele do jiné části. První přesune uživatele na připojení časomíry do informačního systému. Druhé ho naopak přesune na ovládání. Dalším tlačítkem uživatel načte nastavení, ve kterém je souhrn všeho nastavitelného v časomíře, viz. kapitola 5.3.9. Čtvrté tlačítko uživatele přesune na zapínání hdmi výstupu. Předposlední tlačítko odkazuje na stránku s manuální ovládáním, u kterého není zapotřebí připojení čidel. Poslední tlačítko vypíná časomíru. Pro vypnutí se volá funkce `powerOff`, která zobrazí vypínací obrazovku. Následně zavolá php skript `power-off.php`, který časomíru vypne.

5.3.6 Připojení a odpojení k systému

Při načítání stránky pro připojení časomíry do systému se volá funkce `load` načítající stavy časomíry. Dále se spustí funkce `saveSecc`, která pomocí skriptu `login-session.php` zjistí zda–li je uživatel již přihlášen do systému. Pokud přihlášen není zobrazí se mu přihlašovací okno. Toto okno je totožné s oknem v informačním systému, viz. kapitola 5.2.5. Také využívá stejných skriptů umístěných na vzdáleném serveru. Jakmile dostane Javascript odpověď ze serveru, uloží si tuto informaci pomocí volání funkce `saveSecc`, která opět volá php skript `login-session.php` tentokrát však s parametrem uživatelského jména `uname`.

Po přihlášení se uživateli zobrazí nabídka akcí, ke kterým má přístup. Tento seznam je získán pomocí skriptu `event-list.php` umístěném na serveru informačního systému. Jeho funkce je popsána v kapitole 5.2.11. Skript je volán s parametrem jména uživatele `uname`.

Jakmile si uživatel vybere akci, ke které chce časomíru připojit, stiskne tlačítko volající funkci `connEvent`. Ta změní status připojení časomíry na `Online` a uloží jej do cookies souboru. Následně zavolá skript `event-connect.php` s parametry: identifikátoru akce `id` a názvem kategorie `name`. Skript tyto informace uloží na server časomíry, pro další použití v časomíře.

Pro odpojení k časomíry od systému slouží odhlášení. To má na starosti funkce `logout`, která změní stav připojení časomíry, uloží jej a zavolá skript `logout.php`. Skript uloží změnu na serveru časomíry a provede odhlášení uživatele.

5.3.7 Ovládání

Na normální ovládání se uživatel dostane po stisknutí tlačítka `Control`.

Načtení

Při načítání se opět volá funkce `load`, která volá nastavení stavů. Na stránce jsou zobrazeny stavy tlačítek, čas a ovládací tlačítka, viz. obrázek 5.15. V případě, že je časomíra připojena k informačnímu systému se volá funkci `getDBData`. Ta se dotáže skriptu `data.php` s argumenty: název akce `event` a jestli se jedná o inicializaci `init`.

Obrázek 5.15: **Kontrola časomíry**

Tento skript je umístěn a serveru systému. Jako odpověď vrátí informace z databáze týkající se závodníků kategorie, která má nastavené naslouchání časomíře. Skript si data uloží do proměnné `session` a vrátí je zpět. Pro úsporu dat se odesílají informace jen po jejich změně. Javascript po obdržení odpovědi data zpracuje funkcí `processData`, která zobrazí pod startovacím tlačítkem celé jméno závodníka, kterému je právě měřen čas.

Stopování

Pro stopování musí být obě čidla připojená, aby se zpřístupnila tlačítka pro manuální stopnutí času a pro restart času. Startovací tlačítko se zpřístupní po stisknutí spodního čidla závodníkem. Jakmile je čidlo sepnuté a uživatel časomíry stiskne start, zavolá se funkce `start`. Tato funkce vynuluje stopky a zavolá funkci `sound_start`. Ta má na starosti vygenerování se spuštění startovací signalizace. Dále časomíra sleduje, zda–li nebylo spodní čidlo puštěno dříve, než zaznělo třetí pípnutí. Pokud bylo puštěno dříve, je poslední pípnutí

prodlouženo a jako čas je uvedena hodnota `777.00`. Tato hodnota je následně uložena pomocí skriptu `time-save.php` do databáze informačního systému. Interpretace této hodnoty je `Předčasný start`.

Pokud proběhne start bez problémů, volá se po třetím pípnutí funkce `stopWatchStart`. Ta zapne naslouchání streamu skriptu `stopwatch-start.php`, která zapne stopování času a každou setinu sekundy posílá hodnotu času. Čas se stopuje na serveru časomíry, která posílá aktuální čas na stránky. Pro tuto variantu jsem se rozhodl, protože jde o nejpřesnější možné stopování a také čidla komunikují přímo s tímto serverem. Prodleva komunikace stopování a čidel je tady zanedbatelná. Po zastavení času Javascript spustí funkci `sound_stop`, která vygeneruje signalizaci stopnutí času. Následně uloží dosažený čas spuštěním funkce `setDBData`, popsané výše. Nakonec se zastaví naslouchání streamu skriptu `stopwatch-start.php`.

Pokud závodník nedokončí pokus, uživatel vypne stopování času tlačítkem `Stop`. To volá funkci `stop`, který pomocí skriptu `stopwatch-stop.php` zastaví čas místo čidla. Dále se zruší naslouchání streamu `stopwatch-start.php` a do informačního systému se odešle funkcí `setDBData` čas `999.00`, který značí `Nepodařený pokus`.

Tlačítko `Reset` volá funkci `reset`. Ta zastaví naslouchání streamu a vynuluje čas.

Automatický mód

Pokud je zapnutý automatický mód, spouští se stopování času samo po stisknutí spodního čidla, a to s prodlevou jedné a půl sekundy. Spuštění se provádí pomocí volání funkce `start`. Vše probíhá stejně jako u normálního stopování popsané výše v kapitole 5.3.7. Pokud závodník pustí spodní čidlo ještě před spuštěním startovací sekvence, startování se neprovede. Po opětovném zmáčknutí čidla se opět proces spustí od začátku.

5.3.8 Manuální ovládání

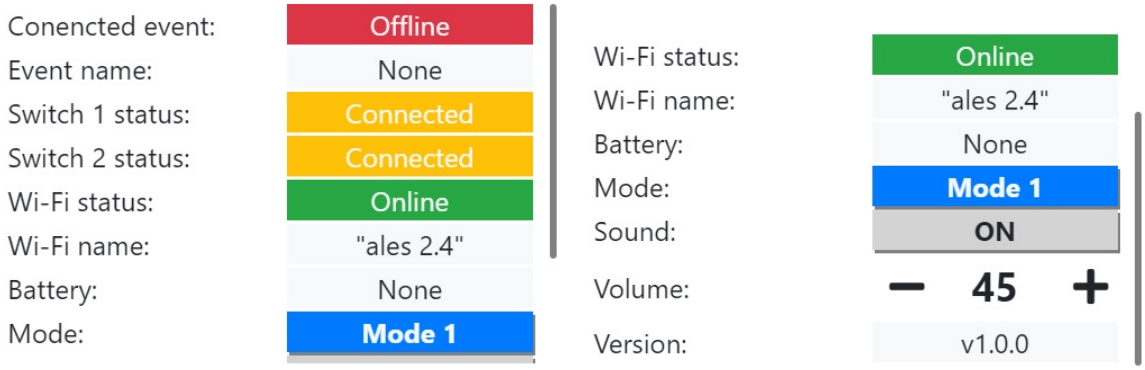
U manuálního stopování není nutné mít připojená čidla. Stopování času se provádí stejně jako u normálního ovládání, viz. kapitola 5.3.7, a za pomocí stejných funkcí. Je zde však několik rozdílů. Jedním je, že při startování se nekontroluje stisknutí spodního čidla. Není zde ani možnost automatického startování. Stopnutí času probíhá stejně jako u normálního ovládání, a to buď ručně tlačítkem `Stop`, nebo pomocí horního čidla. Je zde i tlačítko `Reset`, pro vynulování času.

5.3.9 Nastavení

Na stránce s nastavením jsou přehledně vypsány všechny stavy časomíry, viz. obrázek 5.16. Při načítání stránky se volá funkce `load`, která volá funkci pro nastavení stavů a dále funkci `statsSett` volající funkci `setFromCookies`. Ta zobrazí veškeré informace uložené v souborech cookies. Dále je možné některé statusy měnit.

Změna módu

Lze změnit mód časomíry kliknutím na tlačítko módu. Změna se provádí stejně jako při kliknutí na ikonu statusu módu popsané v kapitole 5.3.3. Při změně se u experimentálního módu zobrazí nová položka určující nastavitelnou délku prodlevy startování. Při mačkání tlačítka na zvětšení nebo zmenšení hodnoty se volá funkce `postDelSet` s parametrem jedna nebo nula. Ta zvýší prodlevu o jednu desetinu sekundy a změnu uloží do souboru cookies. Při



Obrázek 5.16: **Tabulka stavů časomíry**

kliknutí na hodnotu se otevře modální okno s možností jeho upravení pomocí posunovacího tlačítka. Po změně se volá funkce `modalPostDelSet`, která opět vše uloží.

Zvuk

Zvuk lze v nastavení také upravovat. Stejně jako u módu, viz. 5.3.9, je zde možnost upravit hlasitost pomocí tlačítka pro navýšení hlasitosti a tlačítka pro snížení hlasitosti. Posun probíhá vždy o jednu úroveň a volá funkci `volumeSet`. Ta k aktuální hodnotě zvuku přičte jednotku a uloží ji pomocí php skriptu `sound-volume.php`. Celý postup je popsán v kapitole 5.3.3. Další možností úpravy hlasitosti je po kliknutí na aktuální hodnotu. V tomto případě se otevře opět modální okno jako v kapitole 5.3.9, tentokrát však okno obsahuje informace o hlasitosti. Po nastavení hlasitosti se volá funkce `modalVolumeSett`, která pomocí skriptu `sound-volume.php` výslednou hlasitost uloží, viz. 5.3.3.

Uživatel zde může zvuky ztlumit i obnovit hlasitost. Toto zajišťuje funkce `sound` popsaná v kapitole 5.3.3.

5.3.10 Zapnutí výstupu hdmi

Při načítání stránky je opět spuštěna funkce `load`. Pro zapnutí streamování na hdmi výstup je zde tlačítko `Start`, které volá funkce `viewStart`. Tato funkce otevře stream na novém okně prohlížeče posunuté o rozměry displeje časomíry. Tlačítko pro restart spouští tu samou funkci. Tentokrát však okno načte znovu.

5.3.11 Stream na výstupu hdmi

Tato stránka je téměř totožná se stránkou streamu výsledků z informačního systému popsanou v kapitole 5.2.13. Jediným rozdílem je, že při načtení stránky se zapne naslouchání streamu skriptu `view-stopwatch.php`, který posílá data obsahující aktuálně stav stopování. Jsou zde tři možné stavy, a to `Start`, `Wait`, `Stop`. Stav `Start` spouští v náhledu běžení stopek. Při stavu `Wait` se stopky zastavují a čas se zakryje. Po příchodu stavu `Stop` se vloží do času data, která přišla s tímto stavem a čas se opět zobrazí. Tento postup jsem zvolil z důvodu snížení zátěže serveru a nepřesnosti běžení času v prohlížeči. Čas je vynulován vždy s obnovením obsahu, stejně jako tomu je u streamu výsledků v kapitole 5.2.13.

Pokud časomíra není k informačnímu systému připojena, zobrazuje se pouze běžící čas, který se po zastavení nuluje. K nulování dojde po uplynutí pěti a půl sekundy.

5.4 Čidla

Čidla jsou k časomíře připojena pomocí Wi–Fi. Obsluha čidel je naprogramovaná v jazyce C a komunikace probíhá pomocí protokolu `xmlhttp`. Jednotlivá stisknutí se posílají po zaznamenání interuptu na pinu, ke kterému je připojen čip pro měření kapacity. Pro tuto variantu jsem se rozhodl z důvodu šetření baterie. Čidlo se vždy vzbudí při stisknutí, pošle informaci o této skutečnosti časomíře a poté se opět uspí.

Při zapnutí čidla proběhne inicializace a připojení k serveru časomíry. Hned po úspěšném navázání spojení se u čidel zavolá skript `switches.php` s parametrem stavu. A to `Start_switch` u dolního čidla a `Stop_switch` u horního čidla. Parametr `Start_switch` obsahuje `SWITCH_1_ONLINE` a parametr `Stop_switch` obsahuje `SWITCH_2_ONLINE`. Jakmile skript obdrží tyto informace, okamžitě je uloží na serveru. Dále čidla spustí skript `switch-ping-1.php` a `switch-ping-2.php`, které s periodou jedné sekundy kontrolují jejich konektivitu. Jakmile skript zjistí, že je čidlo odpojené okamžitě změnu uloží na serveru a skončí.

Data sepnutí čidel se posílají také skriptu `switches`, se stejnými parametry, ovšem tentokrát obsahují hodnoty `SWITCH_1_ON` u dolního čidla a `SWITCH_2_ON` u toho horního. Při ukončení stisknutí je postup analogický jen je konec `ON` u parametru nahrazen `OFF`. Všechny změny se okamžitě ukládají na server časomíry, aby s nimi mohla časomíra dále pracovat.

Spodní čidlo má nastavenou fixní ip adresu na `192.168.4.13` a horní zase na `192.168.4.14`.

Kapitola 6

Testování a zhodnocení

Tato kapitola je zaměřena na testování integrované časomíry a informačního systému v reálných podmínkách závodu.

6.1 Časomíra

U časomíry je požadovaná přesnost na několik tisícin sekundy. Výsledný čas se zobrazuje v řádu setin sekund, které musejí být naprosto přesné.

6.1.1 Odezva systému časomíry

Ze všeho nejdříve bylo nutné otestovat prodlevu mezi serverem časomíry a grafickým rozhraním. Důvodem je spouštění stopování z GUI a zastavení stopování horním čidlem. Prodleva by tedy mohla zkreslovat dosažený čas závodníka. U testování jsem využil Javascriptovou funkci `setTimeout`, které jsem nastavil čas sepnutí přesně deset sekund po odstartování stopování. Po uplynutí tohoto času se aktivuje funkce vypínající stopování. Je zde tedy, oproti normálnímu stopování, měřená odezva dvakrát. Po patnácti měřeních vyšel stopovaný čas vždy přesně deset sekund.

Další test odezvy jsem prováděl na čase pěti sekund. Postup měření byl stejný. Výsledkem patnácti měření bylo vždy rovných pět sekund.

6.1.2 Hardware čidel

Dále bylo potřeba otestovat správnou funkci čidel, a to v reálných závodních podmínkách. Tomu jsem věnoval první testování v tělocvičně. Během testování jsem sledoval, jestli se čidla spínají správně a bezchybně. Během testování nedošlo k žádnému problému s citlivostí čidel ani s jejich spolehlivostí. Celkem zde proběhlo dvacet testovacích šplhů.

Dále jsem zjistil, že měření předčasného startu má zpoždění. Sérií testů jsem zjistil, že je potřeba jej posunout o dvě stě milisekund. Dalším nedostatkem zjištěným při testování byla mechanická pevnost horního čidla. Závodníkům se podařilo během pěti pokusů ulomit tři ze čtyř šroubů držících čidlo na laně. Proto jsem celý model zpevnil.

6.1.3 Test přesnosti

Druhé testování v tělocvičně se zaměřovalo na přesnost stopování času. Celkem proběhlo patnáct testovacích šplhů, viz. tabulka 6.1. Během testování bylo již nainstalované nové, zpevněné horní čidlo. Během měření se nová konstrukce čidla osvědčila.

V tabulce jsou napsány časy získané pomocí časomíry a časy změřené testerem se stopkami. Jak je vidět, časy se většinou liší o dvě desetiny sekundy. Obvyklý reakční čas lidského mozku se udává okolo dvou desetin sekundy. Z toho je patrné, že čas časomíry není zcela chybný. Pro co nejpřesnější změření přesnosti časomíry, jsem natočil dvanáctý pokus na video s rozlišením 4K na 60fps. Po prozkoumání videa a spočítání počtu framů od počátku startu po dotyk horního čidla, jsem došel k počtu 215 framů. Při hodnotě snímání 60fps to dává přesný čas 3,5833. Tento čas se bez sedmi tisícin shoduje s časem naměřeným časomírou.

Tabulka 6.1: Naměřené časy

	Elektronická časomíra	Člověk
1. měření	6,27 s	6,05 s
2. měření	6,34 s	6,05 s
3. měření	7,58 s	7,48 s
4. měření	6,10 s	5,76 s
5. měření	3,52 s	3,13 s
6. měření	3,53 s	3,23 s
7. měření	6,20 s	6,01 s
8. měření	4,13 s	3,86 s
9. měření	9,18 s	9,14 s
10. měření	3,51 s	3,30 s
11. měření	6,87 s	6,82 s
12. měření	3,59 s	3,31 s
13. měření	6,45 s	6,21 s
14. měření	6,99 s	6,69 s
15. měření	6,81 s	6,57 s

6.2 Informační systém

Informační systém byl otestován na funkčnost všech částí. Testoval jsem editor akcí, editor diplomů, integraci časomíry i vše ostatní. Vše fungovalo bez problému. Je ovšem zapotřebí systém otestovat při větší zátěži na závodech. Tento test jsem neprovedl, jelikož to situace nedovoluje.

6.3 Zhodnocení

U integrované elektronické časomíry lze na základě testování a měření říci, že odpovídá všem navrhovaným požadavkům a že je její přesnost, pro měření závodů Šplhu na laně, dostačující. Informační systém také odpovídá všem navrhovaným požadavkům. Je však žádoucí otestovat ho v plném zatížení na závodech. Z měření jsem také usoudil, že čidla jsou dostatečně citlivá a přesná. Jejich hardware je dobře navržen a software se nijak nezasekává. Také konstrukce časomíry i čidel je dostatečně pevná a ergonomická.

Kapitola 7

Závěr

Cílem práce bylo vytvořit informační systém spolu s integrovanou elektronickou časomírou pro správu akcí Šplh na laně. Tohoto cíle se mi podařilo dosáhnout.

Ze všeho nejdříve jsem prozkoumal a nastudoval dosavadní řešení, viz. kapitola 2.1. Z tohoto řešení jsem vytvořil požadavky na nové řešení popsané v kapitole 2.2. Dále jsem z těchto požadavků vytvořil návrh informačního systém spolu s integrovanou elektronickou časomírou. Celý návrh je popsán v kapitole 4. Dále jsem návrh realizoval. Celý postup řešení jak informačního systému tak i elektronické časomíry a její integrace je popsána v kapitole 5. Nakonec jsem otestoval vytvořené řešení na funkcionalitu a také jsem zkontroloval splnění všech požadavků. Testování se prováděol v několika fázích, jak je popsáno v kapitole 6. Nejdůležitější částí testování bylo otestování přesnosti časomíry.

V práci by bylo možné pokračovat přidáním rozšíření elektronické časomíry o možnost měřit čas více závodníkům naráz, přidání možnosti připojit k časomíře zařízení pomocí technologie Bluetooth nebo přidání baterie k zařízení časomíry. Také čidla by mohla být rozšířena o zapínání pomocí dálkového ovládání. Já budu ve vývoji a vylepšování časomíry nadále pracovat.

Literatura

[1] *Chart.js* [online]. [cit. 2021-04-18]. Dostupné z: https://www.chartjs.org/.

[2] *Pdfmake* [online]. [cit. 2021-04-18]. Dostupné z: http://pdfmake.org/#/.

[3] *QR Codes* [online]. [cit. 2021-04-18]. Dostupné z: https://developers.google.com/chart/infographics/docs/qr_codes.

[4] *SheetJS js-xlsx* [online]. [cit. 2021-04-18]. Dostupné z: https://github.com/SheetJS/sheetjs.

[5] *Tool box parametric* [online]. [cit. 2021-04-18]. Dostupné z: https://www.prusaprinters.org/prints/39729-tool-box-parametric.

[6] RaspberryPi. *Raspberry Pi Documentation* [online]. [cit. 2021-04-15]. Dostupné z: https://www.raspberrypi.org/documentation/.

[7] Varotsis, A. B. *Introduction to FDM 3D printing* [online]. [cit. 2021-04-15]. Dostupné z: https://www.3dhubs.com/knowledge-base/introduction-fdm-3d-printing/.

[8] W3Schools. *CSS Introduction* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/css/css_intro.asp.

[9] W3Schools. *HTML Elements* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/html/html_elements.asp.

[10] W3Schools. *HTML Introduction* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/html/html_intro.asp.

[11] W3Schools. *Introduction to SQL* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/sql/sql_select.asp.

[12] W3Schools. *JavaScript Tutorial* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/js/default.asp.

[13] W3Schools. *JQuery - Filters* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/jquery/jquery_filters.asp.

[14] W3Schools. *JQuery Introduction* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/jquery/jquery_intro.asp.

[15] W3Schools. *JQuery Syntax* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/jquery/jquery_syntax.asp.

[16] W3Schools. *PHP Introduction* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/php/php_intro.asp.

[17] W3Schools. *PHP Syntax* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/php/php_syntax.asp.

[18] W3Schools. *PHP Variables* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/php/php_variables.asp.

[19] W3Schools. *SQL INSERT INTO Statement* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/sql/sql_insert.asp.

[20] W3Schools. *SQL SELECT Statement* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/sql/sql_intro.asp.

[21] W3Schools. *SQL WHERE Clause* [online]. [cit. 2021-04-15]. Dostupné z: https://www.w3schools.com/sql/sql_where.asp.

Příloha A

Obsah přiloženého paměťového média

- **./doc/xtetur01-System-pro-rizeni-sportovnich-akci.pdf** – text bakalářské práce
- **./doc/src** – adresář se zdrojovými kódy textu bakalářské práce
- **./doc/bom.txt** – soubor se součástkami pro horní i dolní čidlo
- **./src/InformationSystem** – adresář se zdrojovými kódy informačního systému
- **./src/Timer** – adresář se zdrojovými kódy elektronické časomíry
- **./src/Sensors/src** – adresář se zdrojovými kódy pro čidla
- **./src/Sensors/DownSensor.zip** – soubor se zdrojovými kódy pro výrobu spodního čidlo
- **./src/Sensors/UpperSensor.zip** – soubor se zdrojovými kódy pro výrobu horního čidlo
- **./models** – adresář obsahuje modely tisknutelných dílů