

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

APLIKÁCIA PRE IMPORT A EXPORT
ÚDAJOV Z DATABÁZY TERMÍNOV
PRETEKOV
KATALÓG POŽIADAVIEK

2024

ADAM GOSTIČ, MIRIAM GRZNÁROVÁ, GABRIEL KOŽUCH,
TOMÁŠ KNEPP

Obsah

1	Úvod	2
1.1	Účel katalógu požiadaviek	2
1.2	Rozsah využitia systému	2
1.3	Slovník pojmov	3
1.4	Referencie	3
1.5	Prehľad nasledujúcich kapitol	4
2	Všeobecný popis	5
2.1	Perspektíva systému	5
2.2	Funkcie produktu	5
2.3	Používatelia	6
2.4	Všeobecné obmedzenia, predpoklady a závislosti	6
3	Špecifikácia požiadaviek	7
3.1	Import pretekov do lokálnej SQLite databázy	7
3.2	Prihlásenie bežcov na preteky	7
3.3	Vloženie prihlásených bežcov do súboru	8
3.4	Používateľ môže zobrazit' Štatistiky bežcov	8
3.5	Export pretekov ako udalosti pre Google calendar	9

Kapitola 1

Úvod

1.1 Účel katalógu požiadaviek

Tento dokument definuje požiadavky a funkcionality projektu "Aplikácia pre import a export údajov z databázy termínov pretekov". Vyvíjaný informačný systém bude slúžiť správcovi klubovej aplikácie Športového klubu Sandberg a taktiež bude výstupom projektu v rámci predmetu Tvorba informačných systémov na FMFI UK v akademickom roku 2024/2025. Dokument je určený pre riešiteľov projektu, zadávateľa a ostatných stakeholderov. Zároveň slúži ako záväzná dohoda medzi zadávateľom a riešiteľmi, ktorá určuje finálnu podobu požiadaviek na výsledný produkt, ktoré sa spísali a upravili na základe požiadaviek zadávateľa projektu.

1.2 Rozsah využitia systému

Vyvíjaný informačný systém bude slúžiť na synchronizáciu a spracovanie údajov medzi lokálnou SQLite databázou a registračným prihláskovým systémom Športového klubu Sandberg používaným na prihlasovanie na preteky v orientačnom behu. Aplikácia využije existujúce API prihláskového systému, ktoré umožňuje čítanie a zapisovanie údajov. Funkcionalita systému zahŕňa:

- Importy
 - Import údajov o blížiacich sa pretekoch z registračného systému.
 - Synchronizácia údajov medzi lokálnou databázou, ktorá získava údaje z existujúcej webovej aplikácie, a prihláskovým

systémom.

- Exporty
 - Export zoznamu pretekárov z lokálnej SQLite databázy do registračného prihláškového systému prostredníctvom API.
 - Export údajov o blížiacich sa pretekoch do zdieľaného Google kalendára.
 - Export štatistík bežcov po ich vyhľadani v databáze is.orientering.sk a výbere požadovaných údajov.

1.3 Slovník pojmov

- **Správca** - Človek, ktorý má zodpovednosť za administráciu a správu klubovej aplikácie
- **FMFI** - Fakulta matematiky, fyziky a informatiky
- **Lokálna databáza** - SQLite databáza vo webovej aplikácii ŠK Sandberg
- **API** - Application programming interface
- **Stakeholder** - Človek, ktorý má nejaký podiel na vývoji systému
- **SZOŠ** - Slovenský zväz orientačných športov
- **Klubová stránka** - webová aplikácia, ktorú používajú členovia Športového klubu Sandberg ako aj správca
- **Produkt** - Výsledný systém

1.4 Referencie

- Github repozitár pripravovaného systému:
<https://github.com/TIS2024-FMFI/preteky.git>
- Github predošlého projektu zameraného na túto tému:
<https://github.com/TIS2023-FMFI/sportovy-pretek>
- Opis API
<https://github.com/TIS2024-FMFI/preteky/tree/main/API/is.orientering.sk>

1.5 Prehľad nasledujúcich kapitol

V nasledujúcich kapitolách sa dokument podrobne zameriava na všeobecnú funkcionálnu štruktúru systému a špecifické požiadavky, ktoré sú naň kladené. V kapitole 2 sa popisuje perspektíva systému, jeho hlavné funkcie, charakteristiky používateľov, všeobecné obmedzenia a na záver aj predpoklady a závislosti, ktoré ovplyvňujú jeho fungovanie. Kapitola 3 následne obsahuje detailnú špecifikáciu požiadaviek na vyvíjaný systém, pričom každá požiadavka je presne definovaná a štruktúrovaná tak, aby spolu s ostatnými požiadavkami poskytovala úplný obraz o funkciách a vlastnostiach systému.

Kapitola 2

Všeobecný popis

2.1 Perspektíva systému

Športový klub Sandberg umožňuje prihlasovanie ich členov na preteky na ich klubovej stránke, lenže tá nie je prepojená s oficiálnou SZOŠ. Kvôli tomu musia byť preteky na klubovej stránke vytvorené manuálne a následne účastníci prihlásení na preteky v tejto klubovej aplikácii prihlásení naspäť do oficiálneho systému SZOŠ.

2.2 Funkcie produktu

Výsledný produkt by mal poskytnúť používateľské rozhranie v podobe konzolovej aplikácie, ktorá mu umožní pracovať s údajmi o pretekoch v klubovom internom systéme. Počas práce by sa užívateľovi vypisovali možnosti všetkých dostupných akcií, z ktorých by si vybral žiadaný úkon. Táto aplikácia by mala umožniť zvoliť mesiac a vyhľadať v ňom preteky a následne získať žiadané dáta z is.orientering.sk, ktoré spracuje, ošetrí vstupy pre kategórie pretekov a vytvorí takéto preteky v SQLite databáze Sandbergu. Po tejto akcii majú pretekári možnosť prihlásiť sa na klubovej stránke pričom admin nemusí manuálne pridať každé jedny preteky. Aplikácia by mala byť schopná aj exportu žiadaných pretekov z SQLite databázy Sandbergu do .csv súboru, ktorá následne môže slúžiť ako input pre SZOŠ API pri prihlasovaní pretekárov na oficiálnu stránku. Toto prihlásenie pretekárov môže byť automatizované za pomoci aplikácie. Ďalším možným vylepšením je import jednotlivých pretekov do google calendar.

2.3 Používatelia

Aplikácia je tvorená výhradne len pre administrátora Sandberg systému, takže nie je potrebné rozdelenie rolí ani prihlasovanie do aplikácie.

2.4 Všeobecné obmedzenia, predpoklady a závislosti

Táto aplikácia pozostáva z troch hlavných rozhraní. Prvé je komunikácia medzi konzolovou aplikáciou a SQLite databázou Sandbergu, kde aplikácia importuje preteky do databázy alebo získava údaje o pretekoch vo formáte csv. Druhé rozhranie je medzi aplikáciou a is.orientering.sk stránkou, ktoré bude poskytnuté už existujúcim API. Tretím rozhraním je komunikácia užívateľa a aplikácie v podobe výberu predvolených command-ov v konzole. Pri komunikácii s lokálnou databázou treba na serveri Sandbergu dať upozornenie, že sa pracuje s databázou. Aplikácia má svoj konfiguračný súbor, kde budú kľúče k API a nastavený oddelovač pre .csv súbory.

Kapitola 3

Špecifikácia požiadaviek

3.1 Import pretekov do lokálnej SQLite databázy

3.1.1 Stiahnutie údajov o pretekoch z API is.orientering.sk.

3.1.1.1 Používateľ si zvolí mesiac, na základe ktorého sa mu v konzole zobrazia všetky preteky, ktoré sa budú konať v príslušnom mesiaci.

3.1.1.1.1 Zobrazuje sa dátum, názov, deadline prihlasovania, miesto, kategórie.

3.1.1.2 Používateľ si vyberie jedny preteky zo zoznamu zobrazených pretekov.

3.1.1.3 Následne dá aplikácii pokyn, aby stiahla údaje o vybratých pretekoch.

3.1.2 Vybrané preteky sa vložia do databázy SQLite vo webovej aplikácii ŠK Sandberg.

3.1.2.1 Ak sa preteky už v databáze nachádzajú, aplikácia upozorní užívateľa a nevloží ho znova.

3.2 Prihlásenie bežcov na preteky

3.2.1 Používateľ si vyberie preteky z SQLite databázy webovej aplikácie ŠK Sandberg.

3.2.1.1 Aplikácia zobrazí preteky pridané do lokálnej databázy, ktorým ešte neuplynul deadline na prihlasovanie.

3.2.2 Aplikácia prihlási na vybrané preteky všetkých bežcov, ktorí sa prihlásili na preteky do deadlinu na prihlasovanie cez web stránku Sandbergu.

3.2.2.1 Prihlasuje ich na is.orientering.sk

3.2.3 Alternatívne sa budú dať prihlásení bežci vyexportovať do CSV súboru.

3.3 Vloženie prihlásených bežcov do súboru

3.3.1 Používateľ si vyberie preteky z lokálnej databázy.

3.3.2 Používateľ vie vygenerovať súbor vo formáte .txt alebo .csv z prihlásených bežcov, ktorý sú prihlásení v lokálnej databáze.

3.3.2.1 Vyberie si formát súboru.

3.3.2.2 V prípade .csv súboru sa oddeľovací znak nastaví podľa konfiguračného súboru.

3.3.2.3 Ak je už bežec pridaný v súbore, tak ho tam nepridá a upozorní užívateľa.

3.3.2 Následne sa súbor uloží do počítača používateľa na zvolenej ceste.

3.4 Používateľ môže zobrazit' Štatistiky bežcov

3.4.1 Používateľ vyhľadá bežca v databáze is.orientering.sk.

3.4.1.1 Vyhľadávať bude podľa presného mena (meno a priezvisko) alebo podľa IDcka ak ho bude vedieť.

3.4.1.2 Aplikácia zobrazí bežcov s tým menom a priezviskom a zobrazí dátum narodenia a klub, do ktorého patria.

3.4.2 Vie si vybrať, ktoré údaje chce o ňom zobrazit' za časový interval, ktoré zadá.

3.4.2.1 Poradie na pretekoch, čas aký mal na pretekoch, strata oproti prvému pretekárovi, účasť na pretekoch.

3.4.3 Tieto štatistiky sa dajú exportovať vo formáte aký si používateľ zvolí.

3.4.3.1 Používateľ má možnosť zvoliť si jeden z formátov: .html, .json, .csv.

3.4.3.2 Súbor sa uloží na zadanej ceste.

3.4.4 Štatistika umiestnení vybraného pretekára za vybraný časový interval sa dá zobrazíť v grafe.

3.4.4.1 Názov grafu bude zadaný časový interval a celé meno pretekára, ktorého sa štatistika týka a aj jeho kategória

3.4.4.2 Na x-ovej osi grafu budú zobrazené jednotlivé preteky za daný interval.

3.4.4.3 Na y-ovej osi grafu budú umiestnenia.

3.4.4.3.1 Prvé miesto má najväčšiu hodnotu.

3.4.4.3.2 Najmenšiu hodnotu budem mať ak sa pretekár neumiestnil v prvej päťici.

3.4.4.4 Graf bude znázornený ako bodový, kde jednotlivé body budú spojené čiarou.

3.5 Export pretekov ako udalosti pre Google calendar

3.5.1 Ak sme si nainportovali nejaké preteky, aplikácia vytvorí udalosti do google kalendára správcu.

3.5.1.1 Prvá udalosť bude v dátume pretekov.

3.5.1.2 Druhá udalosť bude v deadline prihlasovania na preteky.

3.5.1.3 Obe udalosti budú celodenné a budú vyznačené červenou farbou.

Návrh

1 Úvod

1.1 Účel návrhu

Tento dokument slúži ako detailný návrh informačného systému pre projekt “Aplikácia pre import a export údajov z databázy termínov pretekov”. Systém je vyvíjaný pre Športový klub Sandberg a je súčasťou predmetu Tvorba informačných systémov na FMFI UK v akademickom roku 2024/2025. Dokument obsahuje všetky potrebné informácie týkajúce sa implementácie, fungovania a dizajnu systému. Je určený predovšetkým pre vývojárov, ktorí budú systém realizovať a zahŕňa všetky požiadavky uvedené v katalógu požiadaviek.

1.2 Rozsah využitia systém

Tento dokument je úzko prepojený s katalógom požiadaviek a špecifikuje všetky požiadavky, ktoré sú v ňom uvedené. Okrem toho definuje vonkajšie rozhrania, formáty súborov a potrebné API pre správnu funkčnosť systému. Dokument tiež obsahuje návrh používateľského rozhrania konzolovej aplikácie vrátane vizualizácií a diagramov, ktoré detailne popisujú implementáciu systému.

1.3 Referencie

- Github repozitár projektu zameraného na tvorbu systému Športového klubu Sandberg z roku 2017: <https://github.com/TIS2017/SportovyKlub>
- Github repozitár projektu z roku 2023, ktorí menili časť databázy: <https://github.com/TIS2023-FMFI/sportovy-pretek-web>
- API rozhranie is.orientering.sk

2 Návrh komunikácie medzi konzolovou aplikáciou a stránkou is.orientering.sk

V tejto kapitole sa venujeme komunikácii so stránkou is.orientering.sk pomocou REST API. Všetku komunikáciu vieme rozdeliť na dva módy: Get a Post. Komunikácia je zabezpečená skrz bezpečnostný kľúč ktorý je uložený v config súbore

Mód Get

- používame keď získavame dáta z is.orientering.sk
- na každý request bude samostatná funkcia
- funkcie vracajú JSON string
- ako parametre funkcií vkladáme údaje na základe ktorých chceme dáta z is.orientering.sk filtrovať
- requesty:
 - preteky v mesiaci - podľa mesiaca, ktorý zadáme
 - preteky v intervale dátumov
 - * podľa intervalu dátumov ktorý zadáme
 - * registrácie (registrovaný pretekári) v danom klube
 - podľa id klubu

- detaily pretekov
 - * podľa id pretekov
- výsledky pretekára v intervale medzi dvoma dátumami
 - * podľa id pretekára
 - * datумы sú vo formáte YYYY-MM-DD
- výsledky preteku
 - * podľa id pretekov a id eventu
- zoznam všetkých kategórií s detailami

Mód Post

- používame keď vkladáme dáta na is.orienteeing.sk
- na každý request bude samostatná funkcia
- ako parameter vkladame id pretekov a data nakonfigurované v JSON stringu
- funkcie vracajú JSON string s informáciou o registrácii alebo deregistrácii
- requesty:
 - registrácia pretekára na preteky
 - zrušenie registrácie pretekára na preteky
 - * toto je použité len pre testovanie aplikácie, nejedná sa o funkcionálnosť aplikácie

3 Návrh komunikácie medzi konzolovou aplikáciou a lokálnou databázou Sandberg

Táto kapitola predstavuje návrh komunikácie medzi konzolovou aplikáciou a lokálnou databázou Sandberg. Keďže naša aplikácia bude bežať na rovnakom serveri ako lokálna databáza Sandberg, ale bude implementovaná v inom jazyku (naša bude bežať v pythone a aplikácia Sandberg v php), je potrebný prepis a sú rôzne prístupy:

RESTful API umožňuje aplikáciám komunikovať cez HTTP protokol. Aplikácia Sandberg môže poskytovať API endpointy, ktoré naša aplikácia volá na získanie alebo odoslanie údajov. Endpointy budú spracovávať HTTP požiadavky a vracajú odpovede vo formáte JSON. Tieto endpointy budú uložené v jednom PHP skripte, ktorý bude centrálnym bodom komunikácie.

1. Implementácia v Sandberg aplikácii: - Na strane PHP aplikácie vytvoríme dva nové PHP súbory. V súbore api.php sú umiestnené endpointy a v export_import.php sú naimplementované všetky funkcie, ktoré potrebujeme (tieto súbory vieme stiahnuť na server podľa postupu pre inštaláciu návodu). Tieto endpointy budú odchytať HTTP požiadavky z našej aplikácie a následne zavolať príslušné funkcie na strane PHP aplikácie. Výsledky budú vrátené vo forme JSON súboru, ktorý bude odoslaný späť do našej aplikácie.

2. Implementácia v našej aplikácii: - Aplikácia používa knižnice ako requests na volanie API endpointov a spracovanie odpovedí. Vytvoríme dva spúšťacie pythonovské skripty: - **Skript na import pretekov:** Tento skript bude spúšťať akciu importu vybraných pretekov. Bude posilať HTTP požiadavky na PHP aplikáciu Sandberg, ktorá spracuje tieto požiadavky a zavola príslušné funkcie na strane PHP aplikácie. - **Skript na export prihlásených bežcov:** Tento skript bude spúšťať akciu exportu prihlásených bežcov na daný pretek. Opäť bude posilať HTTP požiadavky na PHP aplikáciu Sandberg, ktorá spracuje tieto

požiadavky a zavolá príslušné funkcie na strane PHP aplikácie. - **Skript na export aktívnych pretekov** Tento skript bude spúšťať akciu exportu aktívnych pretekov v databáze Sandberg. Opäť bude posielať HTTP požiadavky na PHP aplikáciu Sandberg, ktorá spracuje tieto požiadavky a zavolá príslušné funkcie na strane PHP aplikácie.

1. Import pretekov do našej aplikácie

- Tabuľky, ktoré sa budú používať:
 - Preteky
 - Kategorie
 - Kategorie_pre
- Funkcie:
 - `existuje_pretek($id)`: Skontroluje, či existuje pretek s daným ID.
 - `existuje_kategoria($id)`: Skontroluje, či existuje kategória s daným ID.
 - `existuje_kat_preteku($id_pret, $category_id)`: Skontroluje, či daná kategória existuje pre daný pretek.
 - `pridaj_pretek_s_id($ID, $NAZOV, $DATUM, $DEADLINE, $POZNAMKA)`: Pridá nový pretek s daným ID do databázy.
 - `pridaj_pretek_s_kontrolou($id, $nazov, $datum, $deadline, $poznamka)`: Skontroluje, či pretek existuje, a ak nie, pridá ho do databázy.
 - `pridaj_kategoriu_s_id($id, $nazov)`: Pridá novú kategóriu do databázy s daným ID.
 - `pridaj_kategoriu_s_kontrolou($id, $nazov)`: Skontroluje, či kategória existuje, a ak nie, pridá ju do databázy.
 - `pridaj_kat_preteku_s_id($id_pret, $id, $id_kat)`: Pridá kategóriu k preteku s daným ID do databázy.
 - `pridaj_kat_preteku_s_kontrolou($id_pret, $id, $id_kat)`: Skontroluje, či kategória preteku existuje, a ak nie, pridá ju.
 - `existuje_kat_pre($id_kat)`: Skontroluje, či kategória existuje v preteku.
 - `spracuj_pretek($competition, $categories)`: Spracuje pretek a priradí kategórie k preteku, pričom využíva ďalšie funkcie.
- Vstupný formát pre funkciu `pridaj_kat_preteku_s_id` bude obsahovať nasledovné parametre:
 - ID (Integer): Id pretekov.
 - NAZOV (String): Názov preteku.
 - DATUM (String): Dátum preteku vo formáte YYYY-MM-DD.
 - DEADLINE (String): Deadline pre registráciu vo formáte YYYY-MM-DD.
 - POZNAMKA (String): Poznámka k preteku, ktorá môže obsahovať aj URL odkazy.

2. Export prihlásených bežcov

- Tabuľky, ktoré sa budú používať:
 - Prihlaseni
 - Pouzivatelia
 - Kategorie
 - Kategorie_pre
- Funkcie:
 - `existuju_prihlaseni($id_pret)`: Skontroluje, či existujú prihlásení

- účastníci pre daný pretek.
- `existuje_pretek($id)`: Skontroluje, či existuje pretek s daným ID.
- `exportujJSON($id_pret)`: Exportuje údaje o účastníkoch preteku do formátu JSON.
- Výstupný súbor je vo formáte json.
 - Parametre:
 - * `meno (string)`: Meno prihláseného bežca
 - * `priezvisko (string)`: Priezvisko prihláseného bežca
 - * `os_i_c (string)`: Osobné číslo prihláseného bežca
 - * `cip (string)`: Číslo čipu prihláseného bežca
 - * `id_kat (integer)`: id kategórie pre daný pretek
 - * `poznámka (string)`: Poznámka

3. Export aktívnych pretekov

- Tabuľky, ktoré sa budú používať:
 - Preteky
- Funkcie:
 - `ziskaj_aktivne_preteky_id()`: Exportuje všetky aktívne preteky
- Výstupný súbor je vo formáte json, konkrétne ako list id pretekov, ktoré sú aktívne.

Komunikačný protokol

1. Endpointy API:

- **POST** `/api/competitions/competition`: Tento endpoint prijíma dáta o pretekoch a kategóriách.
- **GET** `/api/competitions/{id}/export`: Tento endpoint exportuje prihlásených bežcov pre daný pretek.
- **GET** `/api/competitions/active`: Tento endpoint exportuje aktívne preteky z databázy Sandberg.

2. Formát požiadaviek a odpovedí:

- **POST** `/api/competitions/competition`
 - **Požiadavka:**
 - * Obsahuje JSON objekt s informáciami o preteku a kategóriách.
 - * Príklad:

```
{
  "competition": {
    "nazov": "Názov preteku",
    "datum": "2024-11-28",
    "deadline": "2024-12-01",
    "poznámka": "Poznámka k preteku"
  },
  "categories": [
    {
      "id": "97531",
      "category_id": "160",
      "category_name": "A - muži"
    },
    {
      "id": "97541",
      "category_id": "161",
```

```

        "category_name": "A - ženy"
    ]
}

```

– **Odpoveď:**

* Obsahuje JSON objekt s výsledkom operácie.

* Príklad:

```

{
    "status": "success",
    "id": 123
}

```

• **GET** /api/competitions/{id}/export

– **Odpoveď:**

* Obsahuje JSON pole s informáciami o prihlásených bežcoch.

* Príklad:

```

[
    {
        "OS.ČÍSLO": "SKS1952",
        "ID_KATEGÓRIE": 99965,
        "ČIP": "2047994",
        "PRIEZVISKO": "Brňáková",
        "MENO": "Dana",
        "POZNÁMKA": ""
    },
    {
        "OS.ČÍSLO": "SKS7852",
        "ID_KATEGÓRIE": 99954,
        "ČIP": "2049195",
        "PRIEZVISKO": "Brňáková",
        "MENO": "Helena",
        "POZNÁMKA": ""
    },
    {
        "OS.ČÍSLO": "SKS7801",
        "ID_KATEGÓRIE": 99987,
        "ČIP": "2049912",
        "PRIEZVISKO": "Brňák",
        "MENO": "Martin",
        "POZNÁMKA": ""
    }
]
// Ďalšie záznamy...

```

– **GET** /api/competitions/active

– **Odpoveď:**

* Obsahuje JSON pole s id aktívnymi pretekmi.

* Príklad:

```

[ 1910, 1913, 1920 ... ]

```

3. Spracovanie požiadaviek v api.php:

• **POST** /api/competitions/competition

– Požiadavka je spracovaná nasledovne:

* Načítajú sa dáta z tela požiadavky.

* Skontroluje sa, či obsahujú potrebné informácie o preteku a

- kategóriách.
 - * Dáta sa spracujú pomocou funkcie `ExportImport::spracuj_pretek`.
 - * Výsledok sa vráti ako JSON odpoveď.
- **GET** `/api/competitions/{id}/export`
 - Požiadavka je spracovaná nasledovne:
 - * Načítajú sa dáta z URL.
 - * Dáta sa exportujú pomocou funkcie `ExportImport::exportujJSON`.
- **GET** `/api/competitions/active`
 - Požiadavka je spracovaná nasledovne:
 - * Načítajú sa dáta z URL.
 - * Dáta sa exportujú pomocou funkcie `ExportImport::ziskaj_aktivne_preteky_id`.
- Kód môžete vidieť v `api.php`.

4 Návrh “Procesora”

Táto kapitola opisuje centrálny subsystem procesor, ktorý má na starosti:
 - riadenie chodu aplikácie - prepája medzi sebou jednotlivé moduly ktoré komunikujú s vonkajším prostredím (is.orienteeering.sk, Sandberg, Google kalendár, UI)

poskytuje modulom pomocné moduly:

date_converter

- obsahuje metódy na upravovanie datumov do požadovaného formátu
 - obsahuje metódu na konvertovanie datumu zo stringu do datetime objectu
 - obsahuje metódu na konvertovanie času zo stringu do datetime objectu

error handler

- obsahuje triedy pre jednotlivé druhy errorov
 - `HandlerError`
 - `SandbergDatabaseError`
 - `GoogleCalendarServicesError`
 - `IsOrieteeringApiError`
 - `IuError`

config file reader

- zabezpečuje čítanie configuračného súboru
- súbor má nasledujúci formát:

file writer

- zapisuje pretekárov prihlásených na pretek (podľa race ID, ktoré je vstupný parameter)
- vstupné dáta sú vo formáte JSON string
- data obsahujú:
 - meno pretekara
 - priezvisko pretekara


```

1  IS_API_KEY = ""
2  IS_API_ENDPOINT = "https://is.orienteering.sk/api"
3  SANDBERG_API_ENDPOINT = "https://senzor.robotika.sk/sks/api.php/api"
4  GOOGLE_CREDENTIALS_PATH = ""
5  GOOGLE_EMAIL = "gabko.kozuch@gmail.com"
6  CLUB_ID = 46
7  HOME_DIR = ""

```

Obr. 1: config file

- registracne cislo
 - sportident (cislo cipu)
 - meno kategorie na ktoru je prihlaseny
 - poznamka
- zapisuje ich do súboru vo formate csv, txt, html
 - na každý formát súboru (csv, txt, html) má samostatnú metodu

graph creator

- vstupne dáta vo formate JSON string prekonvertuje na statistiky vo forme grafov
- pre vizual grafov pozri kapitolu 9

okrem pomocných modulov obsahuje aj modul **HandlerOfInputsFromUi**

- na základe dopytu z UI volá funkcie z iných modulov
- modulu UI vracia dáta ktoré treba vypísať
 - volá si pomocné moduly ak treba
- uklada si do cache niektore udaje:
 - slovník kategorii: globalne id kategorie : meno kategorie
 - * vytvori sa pri spusteni aplikacie
 - slovník zavodov: id preteku : detaili preteku
 - * preteky sa pridavaju vzdy pri ziskavani preteku z is.orienteering.sk
 - * pouzivame ich ked pridavame pretek do sandberg databazy
 - zoznam pretekarov:
 - * obsahuje prihlasovacie formulare pretekarov
 - * pouzivame ho ked prihlasujeme pretekarov na is.orienteering.sk a pri exporte do suboru
- popis metod HandlerOfInputsFromUi:
 - get_races_from_IsOrienteering_in_month(month: str)
 - * vrati preteky v danom mesiaci (datum, nazov, datum deadlinu na prihlasovanie, miesto, mena kategori)
 - * prida preteky do cache
 - fill_out_cache(input_race: dict)
 - * prida dany pretek do cache
 - * vyhodi error ak uz zadany pretek je v cache
 - import_race_to_Sandberg_Database(race_id: int)
 - * vyberie pretek z cache na zaklade id
 - * prida dany pretek do sandberg databazy
 - * ak pretek s danym id nie je v cache vyhodi chybu

- get_active_races()
 - * ziska id pretekov ktore su pridane v sandberg databaze
 - * ak preteky nie su cache tak ich tam prida
 - * vrati zoznam pretekov (id, datum, nazov, datum deadlinu na prihlasovanie, miesto, mena kategorii)
- fill_runners(race_id: int)
 - * vycisti zoznam pretekarov v cache
 - * prida do cache registracne formulare pretekarov prihlasenych na pretek s danym id
 - * ak sa pretek s danym id nenachadza v cache tak ho tam prida
- sign_runners_to_IsOrienteering(race_id: int)
 - * zavola funkciu fill_runners s id daneho preteku
 - * prihlasi vsetkych pretekarov z cache na pretek s danym id
- fill_runners_with_category_names(race_id)
 - * funkcia funguje rovnako ako fill_runners, s tym rozdielom ze namiesto id kategorii su nazvy kategorii
- convert_data(converter_class, race_id=None)
 - * zavola funkciu fill_runners_with_category_names(race_id) s danym id preteku
 - * ulozi pretekarov z cache do suboru podla converter_class
- convert_html(self, race_id=None), convert_csv(self, race_id=None), convert_txt(self, race_id=None)
 - * volaju funkciu convert data, s parametrom converter_class bud HTML, CSV alebo TXT
- add_to_google_calendar(race_id: str)
 - * prida pretek ako udalost do google kalendara
 - * prida udalost v termine deadlinu prihlasovania
- update_google_event(event_id: str, calendar_id: str, new_data: dict), delete_from_google_calendar(event_id: str, calendar_id: str)
 - * sluzia na upravovanie pridanych udalosti v google kalendary
 - * nie je to funkcionalita nasej aplikacie
- get_runners_from_club()
 - * vrati zoznam pretekarov (id pretekara, meno pretekara, priezvisko pretekara) z klubu podla id zadanom v config subore
- get_runner_results(runner_id, date_from, date_to)
 - * vrati zoznam parametrov:
 - attendance: slovník rok-mesiac : pocet ucasti na pretekoch v danom mesiaci
 - times_after_first: slovník meno preteku : rozdiel casu pretekara daneho runner_id a prveho pretekara
 - date_placement: slovník meno preteku : datum preteku, umiestnenie pretekara, pocet sutaziacich
 - runner_name: meno pretekara
 - meno klubu
 - date_from
 - date_to
- get_race_results(race_id, event_id, competition_category_id)
 - * vrati cas prveho pretekara na danom preteku v danej kategorii a pocet pretekarov v danej kategorii

5. Návrh komunikácie medzi konzolovou aplikáciou a Google Kalendárom

V tejto sekcii detailne popíšeme návrh implementácie komunikácie medzi konzolovou aplikáciou a Google Kalendárom, ktorá umožní automatické pridanie, aktualizáciu a odstránenie udalostí pri registráciách a správach pretekov. Implementácia bude realizovaná prostredníctvom **Google Calendar API** a použitia **Service Account autentifikácie** pre zvýšenie bezpečnosti a automatizácie procesu.

5.1 Implementácia funkčnosti

Autorizácia a autentifikácia

Na komunikáciu s Google Calendar API využívame **Service Account**. Tento prístupový mechanizmus eliminuje potrebu manuálneho prihlasovania, čím sa celý proces plne automatizuje.

1. Vytvorenie Service Account:

- V Google Cloud Console je potrebné vytvoriť Service Account a povoliť prístup k API.
- Po vytvorení stiahnite JSON súbor obsahujúci poverenia.

2. Zdieľanie kalendára:

- Nájdite emailovú adresu Service Account v sekcii **IAM & Admin > Service Accounts**.
- Zdieľajte požadovaný kalendár s týmto emailom a pridajte mu rolu **Make changes to events** (Editor).

3. Autentifikácia v aplikácii:

- Nahrajte súbor s povereniami (napr. `service_account.json`) do aplikácie.
- Pri inicializácii autentifikácie použite knižnicu `google.oauth2.service_account`.

Automatické vytvorenie udalosti

Pri registrácii bežcov na preteky aplikácia automaticky pridá udalosti do kalendára admina. Parametre udalosti sú: - **Názov udalosti**: Obsahuje názov pretekov. - **Dátum a čas**: Definované podľa rozpisu pretekov. - **Miesto**: Lokalita pretekov, ak je dostupná. - **Popis**: Ďalšie informácie alebo URL odkaz na detaily pretekov.

Príklad kódu pre vytvorenie udalosti:

```
def add_event(self, summary, location, description, start_date, end_date, calendar_id):
    event = {
        'summary': summary,
        'location': location,
        'description': description,
        'start': {'date': start_date, 'timeZone': 'Europe/Bratislava'},
        'end': {'date': end_date, 'timeZone': 'Europe/Bratislava'},
```

```
}  
self.service.events().insert(calendarId=calendar_id, body=event).execute()
```

Zrušenie alebo úprava udalosti

Pri zrušení alebo úprave údajov pretekov aplikácia: - **Aktualizuje udalosť** pomocou metódy `events().update()`. - **Odstráni udalosť** pomocou metódy `events().delete()`.

Tieto operácie vyžadujú ID udalosti, ktoré sa uloží pri jej vytvorení.

Formátovanie dátumu a času

Dátumy a časy sú formátované podľa štandardu ISO 8601 (napr. 2025-01-20T10:00:00+01:00), aby boli kompatibilné s požiadavkami Google Calendar API.

5.2 Kroky na implementáciu API komunikácie

1. Vytvorenie projektu v Google Cloud Console

1. Navštívte Google Cloud Console.
2. Kliknite na **Create Project** a zadajte názov projektu.
3. Prejdite do **API & Services > Library** a povolte **Google Calendar API**.

2. Vytvorenie Service Account

1. Prejdite do **IAM & Admin > Service Accounts**.
2. Kliknite na **Create Service Account** a nastavte názov.
3. Pridajte rolu **Editor** alebo **Owner** pre správu udalostí v kalendári.
4. Stiahnite JSON súbor poverení a uložte ho do aplikácie ako `service_account.json`.

3. Zdieľanie kalendára so Service Account

1. Nájdite emailovú adresu Service Account v sekcii **IAM & Admin > Service Accounts**.
2. Otvorte Google Kalendár, kliknite na **Nastavenia > Zdieľanie a povolenia**.
3. Pridajte email Service Account a nastavte oprávnenie **Make changes to events**.

4. Inštalácia knižníc

Nainštalujte potrebné knižnice:

```
pip install --upgrade google-api-python-client  
google-auth-httpplib2 google-auth-oauthlib
```

5. Inicializácia Google Calendar API

Implementujte autentifikáciu pomocou Service Account:

```
from google.oauth2.service_account import Credentials
from googleapiclient.discovery import build

def authenticate():
    creds = Credentials.from_service_account_file(
        'service_account.json',
        scopes=['https://www.googleapis.com/auth/calendar']
    )
    return build('calendar', 'v3', credentials=creds)
```

6. Pridanie udalostí

Pri registrácii alebo zmene údajov pretekov aplikácia automaticky zavolá metódy na pridanie, aktualizáciu alebo zrušenie udalostí.

Príklad vytvorenia udalosti:

```
calendar_service = authenticate()
event = {
    'summary': 'Názov pretekov',
    'location': 'Miesto konania',
    'description': 'Detaily o pretekoch',
    'start': {'date': '2025-01-20', 'timeZone': 'Europe/Bratislava'},
    'end': {'date': '2025-01-21', 'timeZone': 'Europe/Bratislava'},
}
calendar_service.events().insert(calendarId='primary', body=event).execute()
```

5.3 Testovanie a validácia

1. Testovanie autentifikácie:

- Uistite sa, že Service Account má oprávnenia na správu udalostí.

2. Testovanie pridania udalosti:

- Otestujte metódu na vytvorenie udalosti a overte, či sa objaví v kalendári.

3. Testovanie aktualizácie a odstránenia:

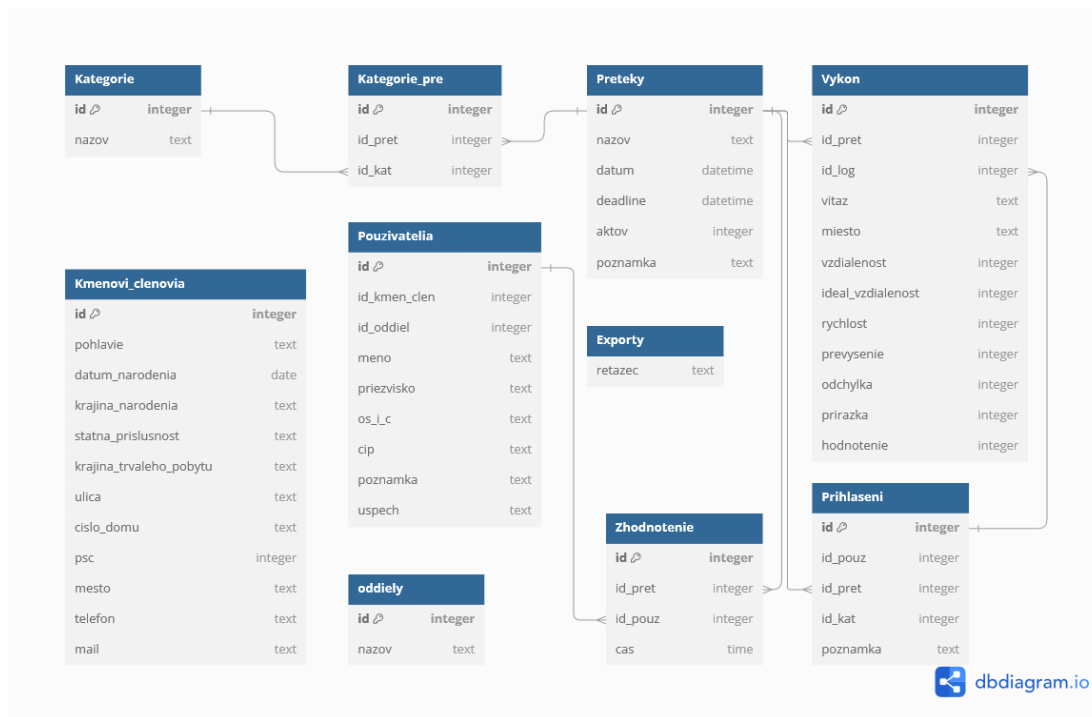
- Skontrolujte, či zmeny v aplikácii správne aktualizujú kalendár.
-

5.4 Výstup a potvrdenie

Po úspešnom pridelení udalosti Google Calendar API vráti ID udalosti. Toto ID sa uloží na neskoršie použitie pri aktualizáciách alebo odstraňovaní. Funkcie vracajú status operácie, ktorý môže byť použitý na logovanie alebo zobrazenie v aplikácii.

6 Návrh dátového modelu

Dátový model je reprezentovaný entitno-relačným diagramom, ktorý ilustruje vzťahy medzi jednotlivými entitami. Entita predstavuje objekt, ktorý existuje samostatne a nezávisle od iných objektov. Vzťahy medzi entitami opisujú prepojenia a interakcie medzi týmito objektmi. Dátový model je prevzatý z existujúcej aplikácie.



Obr. 2: datovy_model

7 Analýza použitých technológií

- RESTful API: Na komunikáciu medzi našou aplikáciou a stránkou is.orientering.sk, ako aj medzi konzolovou aplikáciou a lokálnou databázou Sandberg.
- HTTP protokol: Na volanie API endpointov a spracovanie odpovedí.
- JSON: Na formátovanie dát pre GET a POST požiadavky.
- OAuth 2.0: Na autorizáciu a autentifikáciu pri komunikácii s Google Calendar API.
- Google Calendar API: Na synchronizáciu udalostí medzi našou aplikáciou a Google Kalendárom.
- SQLite: Použitá v pôvodnej aplikácii pre databázové operácie.
- PHP: Použitá v pôvodnej aplikácii Sandberg.
- Python: Použitá v našej aplikácii na implementáciu rôznych funkcií a komunikáciu s API.
- Matplotlib: Na zobrazenie štatistík pretekára.

8 Návrh konzolového rozhrania

Úvodné okno, ktoré sa zobrazí

Pozri Uvodne okno

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- MENU ---
> Import preteku
  Prihlásenie pretekárov
  Export do súboru
  Štatistiky pretekára
```

Obr. 3: Uvodne okno

Po zvolení Import sa zobrazí výber mesiaca

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- ZVOĽTE MESIAC ---
> November, 2024
  December, 2024
  January, 2025
  February, 2025
  March, 2025
  April, 2025
  May, 2025
  June, 2025
  July, 2025
  August, 2025
  September, 2025
  October, 2025
  Next
```

Obr. 4: Mesiace okno

Zoznam pretekov, vyobrazí sa po zvolení mesiaca v Importe, po zvolení prihlásenia na preteky a po zvolení exportu do súboru

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back, 's' to sort items)
--- ZVOĽTE PRETEK (zoradené podľa DÁTUM) ---
> 2023-2-01 | Race 1 | 2023-1-11 | Location 1 | Category 1
  2023-2-02 | Race 2 | 2023-2-12 | Location 2 | Category 2
  2023-4-04 | Race 4 | 2023-10-14 | Location 4 | Category 1
  2023-5-03 | Race 3 | 2023-5-13 | Location 3 | Category 3
  2023-5-05 | Race 5 | 2023-5-15 | Location 5 | Category 2
```

Obr. 5: Preteky okno

Voľba formátu na export

Pozri Formaty okno

Path uloženia vyexportovaného súboru (pri exporte a štatistikách)

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- Prevolená path je: . Chcete ju zmeniť? ---
> Nie
  Áno
```

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- ZVOĽTE FORMAT ---
> csv
  txt
  html
```

Obr. 6: Formaty okno

```
Zadajte novú path:
```

Vyhľadavanie pretekára v štatistike

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- Chcete zvoliť filtre? ---
> Meno
  ID
  Bez filtra
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back, 's' to sort items)
--- ZVOĽTE PRETEKÁRA ---
> ['meno', '1.1.2000', 'klub']
  ['meno2', '1.1.2001', 'klub2']
```

Zadávanie parametrov štatistiky

```
--- ZVOĽTE INTERVAL ŠTATISTIKY PRE meno ---
Začiatok intervalu (dátum): Nenastavený
Koniec intervalu (dátum): Nenastavený
> Nastavte začiatok intervalu
  Nastavte koniec intervalu
  Zvoľte typ štatistiky
```

Obr. 7: Interval okno

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- Zvoľte typ štatistiky pre meno ---
> Poradie na pretekoch
  Časy na pretekoch
  Strata oproti prvému
  Účast na pretekoch
```

Obr. 8: Statistika okno

“services”: [//pole služieb, ktoré si chce objednať (1-X, v poli sú iba tie služby, ktoré si objednáva)]

Po stlačení q, ukončuje konzolovú aplikáciu

```
(press 'q' to quit, UP and DOWN to navigate, ENTER to select option, 'b' for back)
--- Chcete naozaj odísť ---
> Nie
  Áno
```

Obr. 9: quit

9 Návrh zobrazenia štatistík

Ako zobrazenie štatistík pretekára má užívateľ možnosť ich zobrazíť a vyexportovať v PDF súbore za pomoci default Python knižnice Matplotlib

Údaje štatistiky

Pozri tabulka1

Meno	Alexander Fekete
Klub	Pretekársky klub Rimavská sobota
Časový interval	1.1.2000 -> 2.2.2002

Obr. 10: tabulka1

Graf počtu účastí na pretekoch za mesiac

Pozri graf1

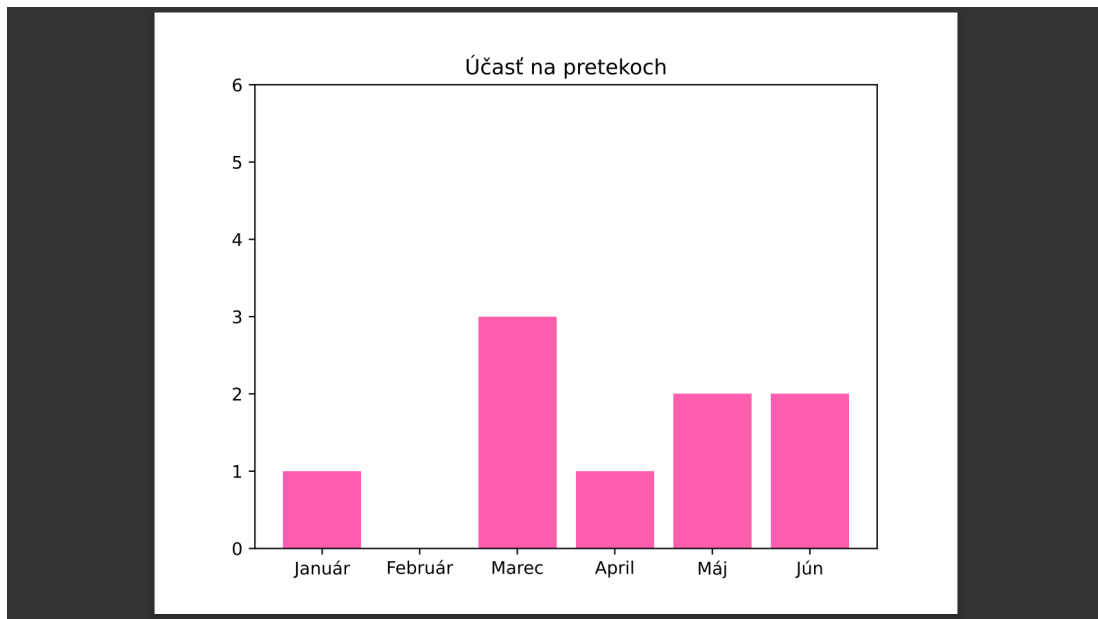
Graf umiestnení na pretekoch, ktorých sa zúčastnil

- 1,2,3 reprezentujú miesta, zlomky (napr. 1/4) reprezentujú, že v koľkej štvrtine počtu umiestnení sa umiestnil

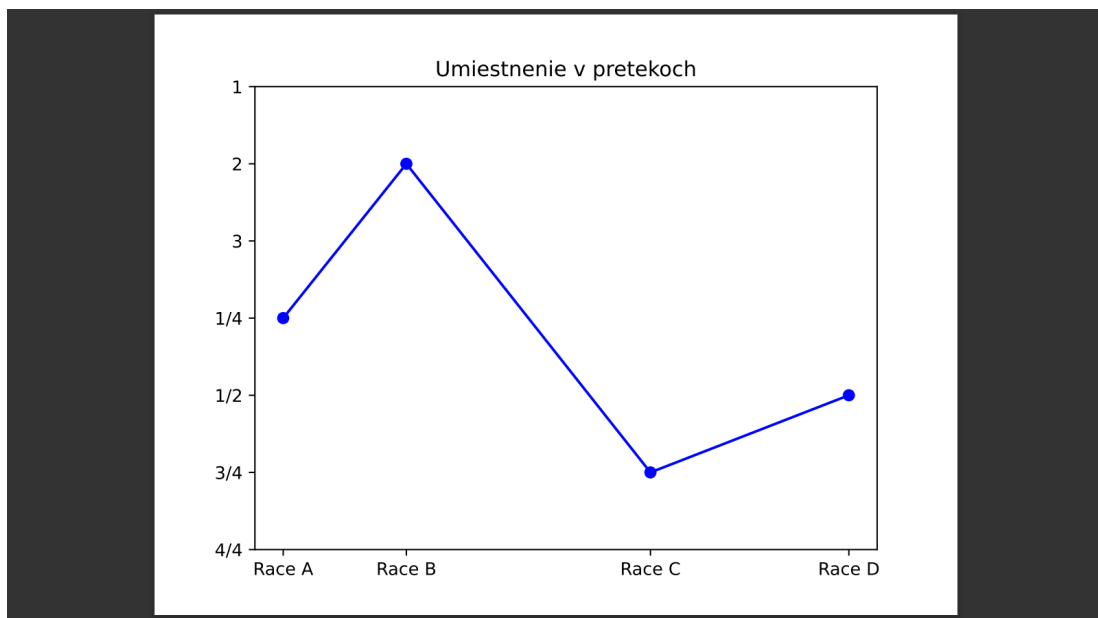
Pozri graf2

Graf časového rozdielu od prvého pretekára na jednotlivých pretekoch

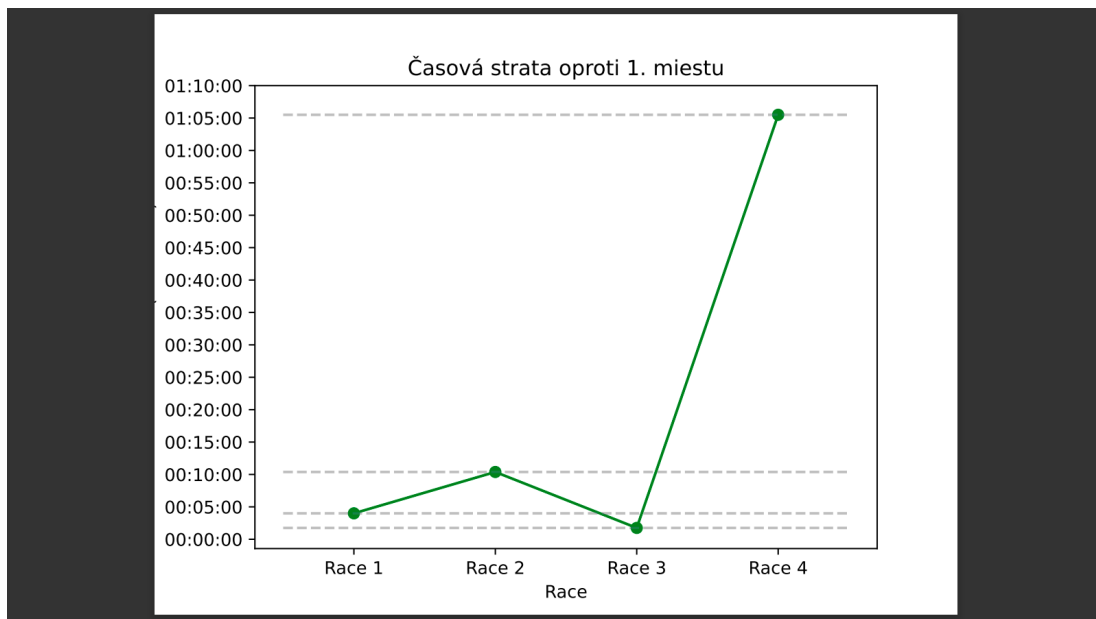
Pozri graf3



Obr. 11: graf1



Obr. 12: graf2



Obr. 13: graf3

10 Diagramy

10.1 Use-case diagram

Slúži na pomenovanie základných hrubých používateľských scenárov a zobrazuje všetky činnosti, ktoré bude vykonávať správca systému. Diagram slúži najmä ako sumarizujúci pohľad a prehľad všetkých používateľských scenárov.

Pozri use_case

10.2 Component diagram

Pozri component_diagram

10.3 Class diagram

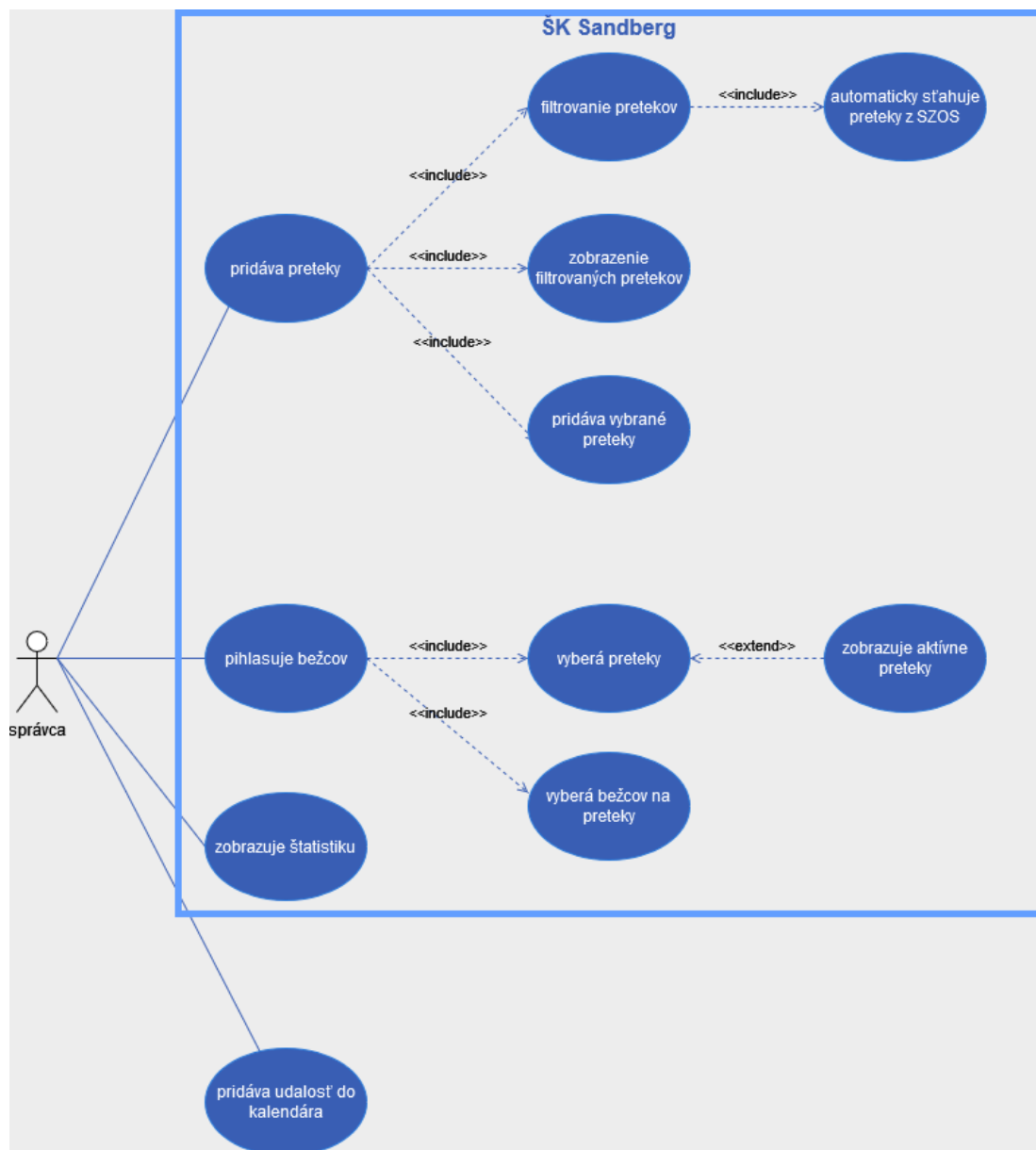
Pozri class_diagram

11 Harmonogram implementácie a testovania

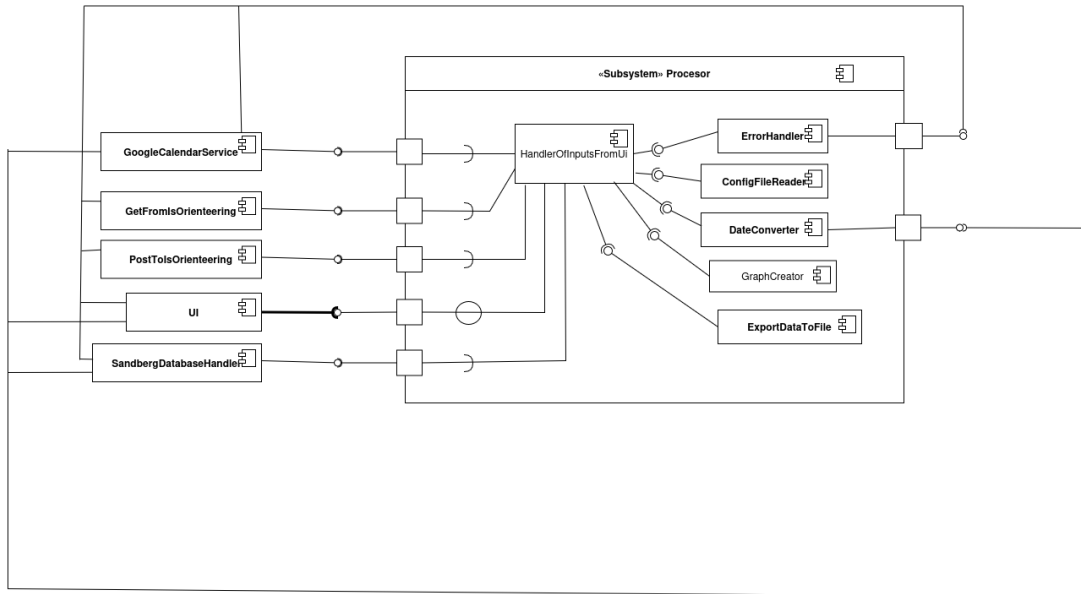
Implementáciu a testovanie rozdelíme na vlny. Každá vlna sa skladá z troch častí implementácia, testing, oprava. Jednotlivé vlny budú vyzeráť takto:

1 vlna:

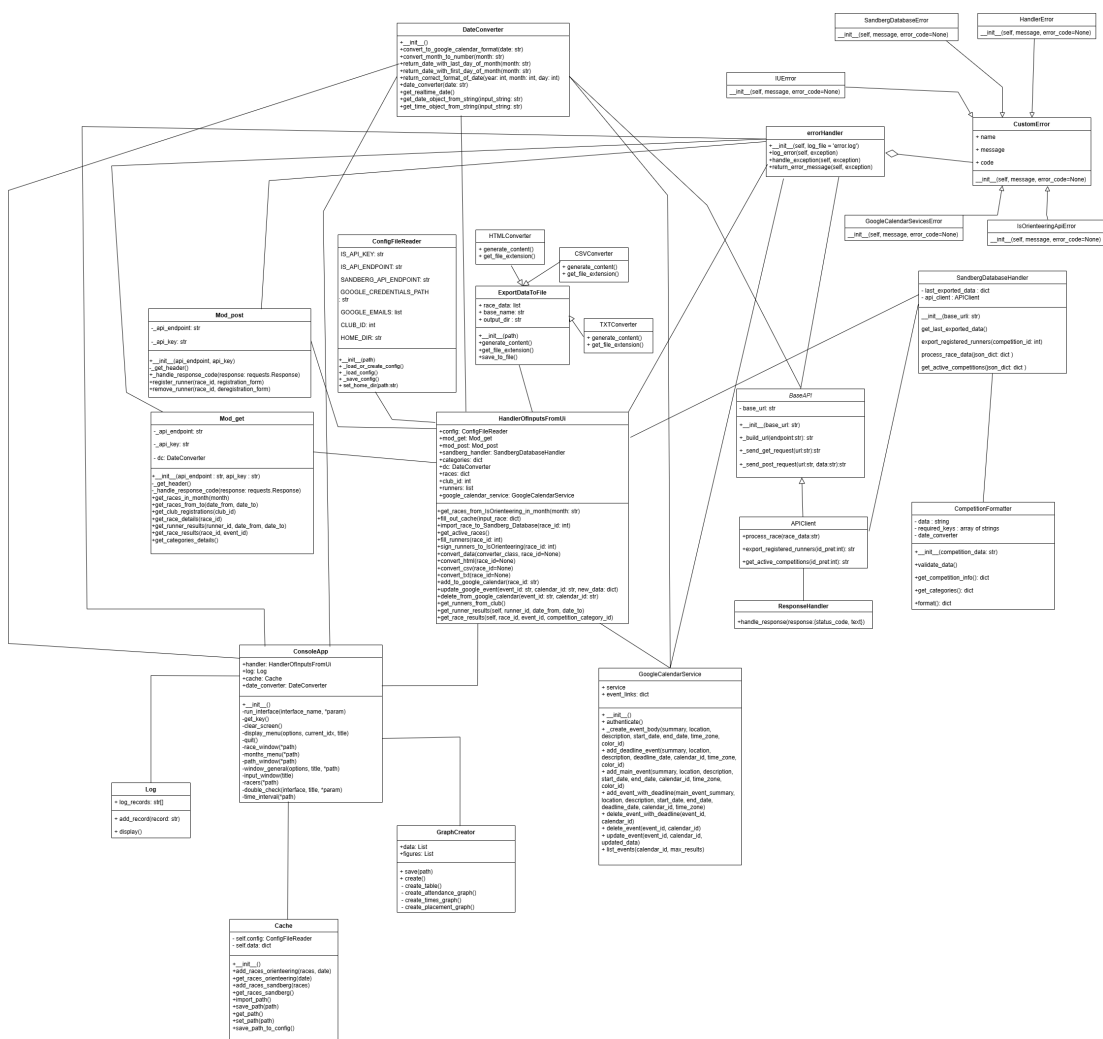
- implementácia:
 - podmoduly procesora - error handler, date_converter, config file reader
 - UI
- testing:
 - podmoduly procesora - error handler, date_converter, config file reader



Obr. 14: use_case



Obr. 15: component_diagram



Obr. 16: class diagram

- UI

2 vlna:

- implementácia:
 - Pripojenie do Google kalendara
 - Get_mode
 - Post_mode
 - * Pripojenie na sandberg databazu
 - import
 - export
- testing:
 - Pripojenie do Google kalendara
 - Get_mode
 - Post_mode
 - * Pripojenie na sandberg databazu
 - import
 - export

3 vlna

- implementácia:
 - podmodul procesora - file writer
 - handlerOfInputsFromUi
 - * komunikacia so vsetkymi modulmi
 - * graph creator
- testing:
 - podmodul procesora - file writer
 - graph creator
 - * handlerOfInputsFromUi
 - komunikácia so všetkými modulmi

4 vlna

- implementácia:
 - všetky úpravy a optimalizácie kódu
- testing:
 - celkové fungovanie aplikácie

Testovacie scenare

Scenár 1: Import pretekov do lokálnej SQLite databázy

1. Používateľ vykoná kroky z README.md na inštaláciu a konfiguráciu aplikácie.
2. Používateľ spustí aplikáciu a vyberie možnosť Import pretekov.
3. Používateľ si zvolí mesiac, na základe ktorého sa mu zobrazia všetky preteky v danom mesiaci.
 - **Vedľajšia možnosť:** Ak sa v zvolenom mesiaci nekonajú žiadne preteky, aplikácia zobrazí správu: "V zvolenom mesiaci sa nekonajú žiadne preteky."
4. Používateľ si vyberie jedny preteky zo zoznamu a dá pokyn na ich import.
 - Aplikácia stiahne údaje o vybraných pretekoch a vloží ich do lokálnej SQLite databázy.
 - **Úspešná správa:** "Preteky boli úspešne importované do databázy."
 - **Vedľajšia možnosť:** Ak sa preteky už v databáze nachádzajú, aplikácia upozorní používateľa správou: "Tieto preteky už existujú v databáze."

Požiadavky pokryté: - 3.1 Import pretekov do lokálnej SQLite databázy

Scenár 2: Prihlásenie bežcov na preteky

1. Používateľ vykoná kroky z README.md na inštaláciu a konfiguráciu aplikácie.
2. Používateľ spustí aplikáciu a vyberie možnosť Prihlásenie pretekárov.
 - Aplikácia zobrazí preteky pridané do lokálnej databázy, ktorým ešte neuplynul deadline na prihlasovanie.
 - **Vedľajšia možnosť:** Ak neexistujú žiadne preteky s neuplynutým deadline, aplikácia zobrazí správu: "Nenašli sa žiadne preteky s neuplynutým deadline."
3. Používateľ si vyberie preteky a aplikácia prihlási všetkých bežcov, ktorí sa prihlásili cez web stránku Sandbergu.
 - **Úspešná správa:** "Všetci bežci boli úspešne prihlásení na vybrané preteky."
 - **Vedľajšia možnosť:** Ak sa žiadni bežci neprihlásili, aplikácia zobrazí správu: "Nenašli sa žiadni prihlásení bežci."
4. Alternatívne sa bežci vyexportujú do CSV súboru.
 - **Úspešná správa:** "Bežci boli úspešne exportovaní do CSV súboru."

Požiadavky pokryté: - 3.2 Prihlásenie bežcov na preteky

Scenár 3: Vloženie prihlásených bežcov do súboru

1. Používateľ vykoná kroky z README.md na inštaláciu a konfiguráciu aplikácie.
2. Používateľ spustí aplikáciu a vyberie možnosť Export do súboru.
3. Používateľ si vyberie preteky z lokálnej databázy.
 - **Vedľajšia možnosť:** Ak neexistujú žiadne preteky v databáze, aplikácia zobrazí správu: "Nenašli sa žiadne preteky v databáze."

4. Používateľ vygeneruje súbor vo formáte .txt alebo .csv z prihlásených bežcov.
 - **Úspešná správa:** “Súbor bol úspešne vygenerovaný.”
 - **Vedľajšia možnosť:** Ak sú bežci už pridaní v súbore, aplikácia zobrazí správu: “Bežec už existuje v súbore.”
5. Súbor sa uloží na zvolenej ceste.
 - **Úspešná správa:** “Súbor bol úspešne uložený na zvolenú cestu.”

Požiadavky pokryté: - 3.3 Vloženie prihlásených bežcov do súboru

Scenár 4: Zobrazenie štatistík bežcov

1. Používateľ vykoná kroky z README.md na inštaláciu a konfiguráciu aplikácie.
2. Používateľ spustí aplikáciu a vyberie možnosť štatistiky pretekára.
3. Používateľ vyhľadá bežca podľa mena alebo ID.
 - **Vedľajšia možnosť:** Ak sa bežec nenájde, aplikácia zobrazí správu: “Bežec sa nenašiel.”
4. Používateľ si vyberie údaje, ktoré chce zobraziť za zadaný časový interval.
5. Štatistiky sa zobrazia a môžu sa exportovať vo formáte .html, .json, alebo .csv.
 - **Úspešná správa:** “Štatistiky boli úspešne exportované.”
6. Štatistika umiestnení sa dá zobraziť v grafe.
 - **Úspešná správa:** “Graf štatistík bol úspešne vygenerovaný.”
 - **Vedľajšia možnosť:** Používateľ si vie graf exportovať do pdf súboru

Požiadavky pokryté:

- **3.4 Používateľ môže zobraziť štatistiky bežcov**

Scenár 5: Export pretekov ako udalosti pre Google Calendar

Prípad 5.1: Pridanie udalosti s deadline

Predpoklady:

- Preteky majú definovaný dátum a deadline pre registráciu.
- Service Account má pridelené práva na pridanie udalostí do cieľového kalendára.
- K dispozícii je platný súbor `service_account.json`.

Kroky testovania:

1. Admin pridá informácie o nových pretekoch do aplikácie.
2. Aplikácia vygeneruje dve udalosti:
 - **Hlavná udalosť** obsahujúca dátum konania pretekov.
 - **Deadline udalosť** obsahujúca konečný termín registrácie na preteky.
3. Aplikácia odošle požiadavky na vytvorenie oboch udalostí do Google Calendar API:
 - API požiadavky zahŕňajú parametre ako:
 - Názov udalosti (napr. „Preteky X“ a „Deadline: Preteky X“).
 - Dátum udalosti (pre hlavný event aj deadline).
 - Miesto udalosti (ak je k dispozícii).
4. API vráti úspešné odpovede pre obidve udalosti, vrátane ich ID.

5. Aplikácia zobrazí správu: „Udalosti boli úspešne pridané do Google Kalendára.“

Očakávaný výsledok:

- Obe udalosti sú viditeľné v cieľovom kalendári:
 - **Hlavná udalosť** je označená modrou farbou a obsahuje detaily o pretekoch.
 - **Deadline udalosť** je označená červenou farbou a obsahuje konečný termín registrácie.
 - Udalosti majú správne nastavené:
 - Dátumy a časy.
 - Názvy a popisy.
 - Miesto (ak je k dispozícii).
-

Prípad 5.2: Pridanie udalosti bez deadline

Predpoklady:

- Preteky majú definovaný iba dátum konania bez deadline na registráciu.
- Service Account má pridelené práva na pridanie udalostí do cieľového kalendára.
- K dispozícii je platný súbor `service_account.json`.

Kroky testovania:

1. Admin pridá informácie o nových pretekoch do aplikácie.
2. Aplikácia identifikuje, že udalosti nemajú definovaný deadline.
3. Aplikácia odošle požiadavku na vytvorenie iba jednej udalosti do Google Calendar API:
 - API požiadavka zahŕňa parametre ako:
 - Názov udalosti (napr. „Preteky X“).
 - Dátum udalosti.
 - Miesto udalosti (ak je k dispozícii).
4. API vráti úspešnú odpoveď pre udalosť, vrátane jej ID.
5. Aplikácia zobrazí správu: „Hlavná udalosť bola úspešne pridaná do Google Kalendára.“

Očakávaný výsledok:

- Hlavná udalosť je viditeľná v cieľovom kalendári:
 - Označená modrou farbou.
 - Obsahuje správne nastavené:
 - * Dátum a čas.
 - * Názov a popis.
 - * Miesto (ak je k dispozícii).