MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Vizualizácia štatistík hodnotenia pre nástroj Frag

Bakalárska práca

MICHAL ŠOLTIS

Brno, jeseň 2023

MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Vizualizácia štatistík hodnotenia pre nástroj Frag

Bakalárska práca

MICHAL ŠOLTIS

Vedúci práce: RNDr. Petr Ročkai, Ph.D.

Katedra počítačových systémov a komunikácií

Brno, jeseň 2023


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Vyhlásenie

Vyhlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Michal Šoltis

Vedúci práce: RNDr. Petr Ročkai, Ph.D.

Poďakovanie

Chcem sa poďakovať Petrovi Ročkaiovi za jeho odbornú pomoc a ochotu pri vedení mojej poslednej bakalárskej práce. Rovnako chcem vyjadriť vďaku svojej rodine a priateľom za ich podporu počas celej akademickej cesty. Vážim si všetko, čo pre mňa urobili.

Zhrnutie

V súčasnej digitálnej ére je dôležité mať nástroj na efektívne riadenie a správu procesu vzdelávania. Táto práca prispieva do tejto oblasti prostredníctvom webovej aplikácie zameranej na vizuálne príťažlivé sledovanie a analýzu úspešnosti študentov počas semestra prostredníctvom interaktívnych grafov. Umožňuje intuitívne prechádzanie medzi jednotlivými predmetmi a zjednodušuje monitoring bodového hodnotenia študentov. Práca je štruktúrovaná do štyroch kapitol, ktoré pokrývajú analýzu, výber technológii, implementáciu a nasadenie aplikácie.

Kľúčové slová

aplikácia, Python, Frag, štatistika, grafy

Obsah

Zoznam obrázkov

Úvod

Na fakulte informatiky sa vo viacerých predmetoch zameraných na programovanie používa softvér Frag, ktorý slúži okrem iného na automatické odovzdávanie a hodnotenie domácich úloh. Vzhľadom na veľký počet študentov je potrebné mať prehľad o aktuálnej úspešnosti študentov a ich počte bodov získaných z týchto úloh v akejsi vizuálne rozlíšiteľnej forme.

Cieľom tejto práce je vytvoriť webovú aplikáciu, ktorá bude schopná vizualizovať dáta a štatistiky týkajúce sa bodov, ktoré študenti dosiahli v rôznych predmetoch. Aplikácia bude poskytovať interaktívne prostredie s možnosťou prehľadného prechádzania medzi predmetmi a analýzou výsledkov pomocou interaktívnych grafov.

Práca je rozdelená do štyroch hlavných kapitol. Kapitola 1 sa zaoberá analýzou požiadaviek aplikácie a softvéru Frag. Kapitola 2 popisuje a odôvodňuje použité technológie použité v rámci implementačnej časti práce. Kapitola 3 rozoberá celkový návrh implementácie, kritické kroky a takisto výslednú štruktúru aplikácie. Kapitola 4 zhrňuje dosiahnuté výsledky a nasadenie aplikácie na používanie vo výuke na fakulte informatiky.

Výsledkom tejto práce je voľne prístupná aplikácia v rámci privátnej siete fakulty informatiky, ktorá prispieva k lepšej transparentnosti a zrozumiteľnosti bodového hodnotenia študentov v predmetoch, kde je Frag využívaný.

1 Analýza a návrh

Táto kapitola popisuje štruktúru a funkcionalitu softvéru Frag, zvlášť v kontexte bodovania a vizualizácie študentských výsledkov. Na konci kapitoly sú zadefinované požiadavky, ktoré by mala aplikácia spĺňať, aby z nej učitelia mali radosť.

1.1 Frag

Frag je softvér určený na podporu výuky v kurzoch zameraných na programovanie. Aktuálne je k dispozícii ako nástroj v termináli (anglicky *command line tool*). Medzi základnú funkcionalitu patrí distribúcia študijných materiálov, získavanie odovzdaných úloh študentov, poskytovanie spätnej väzby na odovzdané úlohy, známkovanie a podobne. Medzi predmety ktoré Frag využívajú patria:

- IB111 (Základy programování)
- PB152 (Operační systémy - cvičení)
- PB111 (Principy nízkoúrovňového programování)
- PV288 (Python Seminar)
- PB161 (Programování v jazyce C++)
- PV248 (Advanced Programming in C++)
- IB002 (Algoritmy a datové struktury I)

Ako relačný databázový systém (anglicky *Relational Database Management System* - DBMS) Frag používa PostgreSQL. Prístup k dátam je zabezpečený prostredníctvom systému založeným na rolách a sieťového autentifikačného protokolu Kerberos.

Kerberos je protokol, ktorý používa 'tikety' na overenie identity užívateľov v počítačových sieťach. Prístup k databáze Frag majú teda iba osoby s platným Kerberos tiketom. Tento tiket, získaný po prihlásení na školský server, slúži na overenie identity učiteľa alebo študenta,

a umožňuje im vykonávať operácie v databáze v súlade s ich oprávneniami.

Všetky predmety, ktoré používajú Frag majú rovnakú schému databázy a drvivá väčšina z nich má rovnaký systém bodovania čo zjednodušuje procesy tvorby, úprav a dotazovania dát. Každý predmet je reprezentovaný samostatnou databázou, v rámci ktorej sú definované aj pohľady, ktoré umožňujú efektívne a rýchle spracovanie dát bez potreby komplexných dotazov priamo na základné tabuľky.

Beh predmetov, ktoré majú jednotný systém bodovania, je rozdelený v rámci jedného semestra do troch, respektíve štyroch blokov, kde štvrtý blok sa berie ako skúškové obdobie. Každý blok má štyri týždne počas ktorého študenti zbierajú body podľa vlastného uváženia.

Tento systém bodovania dáva študentom pomerne veľkú flexibilitu a voľnosť. V každom bloku je nutné získať minimálny počet bodov, ktoré sa dajú získať vypracovaním domácich úloh, vypracovaním príprav na cvičenia, za kvalitu kódu, účasť a aktivitu na cvičení. Celkové hodnotenie sa líši v závislosti od ukončenia predmetu. Vzhľadom na veľký počet študentov je žiadúce monitorovať náročnosť predmetov a úspešnosť študentov v jednotlivých kategóriách v každom bloku semestra.

1.2 Vizualizácia blokov

Schopnosť monitorovať výsledky a porovnávať úspešnosť študentov je v kontexte zlepšenia výuky veľmi užitočné. Jednou z možností ako to docieliť je vizualizovať dáta pomocou grafov, konkrétne pomocou tzv. skladaného stĺpcového grafu (anglicky *stacked bar chart*).

Skladaný stĺpcový graf predstavuje rozšírenie štandardného stĺpcového grafu z pohľadu na číselné hodnoty v rámci jednej kategorickej premennej na dve. Každý stĺpec v štandardnom stĺpcovom grafe je rozdelený na niekoľko čiastkových stĺpcov naskladaných od jedného konca k druhému, pričom každý z nich zodpovedá úrovni druhej kategorickej premennej. [1]

V kontexte aplikácie, každý stĺpec reprezentuje aktuálny počet bodov študenta za semester. Stĺpec je rozdelený do segmentov zodpovedajúcich rôznym kategóriám bodovania, ako sú domáce úlohy, príprava na cvičenia, kvalita kódu a tak ďalej. Farebné rozlíšenie jed-

notlivých segmentov uľahčí orientáciu v grafe, pričom každá kategória bude mať pridelenú svoju farbu s rôznymi odtieňmi.

1.3 Požiadavky

Aby bolo pre vyučujúcich jednoduché monitorovať a posudzovať výsledky študentov je nutné aby aplikácia splňovala určité požiadavky. Nasleduje štruktúrovaný zoznam požiadaviek na základe predchádzajúceho popisu.

1. poskytnutie pohľadu zobrazujúceho celkovú úspešnosť študentov vo forme stĺpcového grafu
2. farebné rozlíšenie jednotlivých bodových zdrojov v grafe
3. interaktívnosť grafu (filtrovanie a zmenu zobrazovaných dát v reálnom čase)
4. možnosť zmeny zobrazeného predmetu v grafe na požiadanie
5. exportovanie grafu do statických formátov (*.png* alebo *.jpg*)
6. rýchla a intuitívna navigácia v rámci aplikácie
7. možnosť prispôsobiť vzhľad grafu podľa potrieb vyučujúceho

2 Použité technológie

Táto kapitola popisuje kritériá a rozhodovací proces výberu technológií, ktoré boli použité pri implementácii.

2.1 Výber jazyka

Výber vhodného programovacieho jazyka je základným krokom, ktorý môže výrazne ovplyvniť priebeh a výsledok projektu. Výber bol uskutočnený na základe nasledujúcich kritérií:

- Požiadavky aplikácie: Aplikácia je zameraná na vizualizáciu dát, preto je potrebný jazyk, ktorý efektívne spracúva dáta a podporuje ich transformáciu do grafickej podoby.
- Dostupnosť knižníc a nástrojov: Jazyk s bohatým ekosystémom knižníc môže značne urýchliť vývoj.
- Komunitná podpora: Jazyk s aktívnou komunitou zabezpečuje dostupnosť zdrojov, tutoriálov a rýchle riešenie problémov.
- Časová efektivita: Vzhľadom na časové obmedzenia projektu na jeden semester je potrebný jazyk, ktorý umožní rýchlu implementáciu.
- Vlastnosti jazyka: Skriptovacie jazyky sú často považované za rýchlejšie vývojové nástroje v porovnaní s kompilovanými jazykmi.

Na základe vyššie uvedených kategórií neboli zvážené jazyky C, respektíve C++, ktoré neposkytujú dostatočnú úroveň abstrakcie a sú viac zamerané na nízkoúrovňové programovanie. Tieto jazyky, hoci sú výkonné a ponúkajú vysokú kontrolu nad systémovými zdrojmi, ich použitie by zvýšilo časové a vývojové nároky projektu čo by mohlo mať fatálne následky.

Ďalšou potenciálnou skupinou sú objektovo-orientované jazyky. Objektovo-orientované jazyky, napr. Java či C#, môžu byť prebytočné pre tento konkrétny účel, keďže aplikácia sa zameriava hlavne na tok informácií (anglicky *information flow*) kde významu rolu hrá získanie,

spracovanie, ukladanie a vizualizácia dát a nie na vytváranie objektov. Tieto faktory smerujú k funkcionálnemu prístupu a teda nie k objektovo-orientovanému. V jazykoch ako Java a C# je dôraz kladený práve na objektovo-orientované paradigmy, kde sú funkcie väčšinou integrované do objektov ako metódy.

Medzi uváženými jazykmi boli aj jazyky špecificky zamerané na štatistiku alebo numerické výpočty, ako sú R, MATLAB alebo Julia, ktoré majú svoje silné stránky ale môžu byť obmedzené v iných oblastiach softvérového vývoja, ako je napríklad v schopnosti internovať s databázou či inými nástrojmi a technológiami.

Z uvažovaných programovacích jazykov boli do užšieho výberu zaradené JavaScript (TypeScript), Python, Go a Rust. Opäť, každý z týchto kandidátov má svoje silné stránky, avšak nie všetky sú rovnako vhodné pre špecifické požiadavky aplikácie. JavaScript (TypeScript) sice vyniká v oblasti webového vývoja, ale môže si vyžadovať zložitejšie konfiguračné procesy, čo v kontexte aplikácie predstavuje nevýhodu. Rust a Go, napriek ich rýchlosti a bezpečnostným aspektom, nedisponujú tak rozvinutým ekosystémom pre prácu s dátami, aký je nevyhnutný pre efektívnu spracovanie a vizualizáciu dát v rámci projektu.

Výsledkom tejto úvahy je, že Python poskytuje vyváženú kombináciu jednoduchosti, rýchlosti vývoja, rozsiahleho a bohatého ekosystému knižníc a flexibilite. Tieto charakteristiky ho činia ideálnym riešením pre aplikácie zamerané na spracovanie a vizualizáciu dát, a preto bol vybraný ako preferovaný programovací jazyk pre realizáciu tohto projektu.

2.2 Výber knižnice

Python je obľúbený v širokej škále odvetví vďaka svojej všestrannosti, jednoduchosti a rozsiahlej komunitnej podpore. Jeho ekosystém ponúka mnoho knižníc pre vizualizáciu dát. Medzi najpopulárnejšie patrí Matplotlib, Seaborn, Bokeh, či Plotly. Pravda je, že mnohé z týchto vizualizačných knižníc majú podobné vlastnosti. Všetky sú po-

merne robustné a poskytujú rozsiahle možnosti na vytváranie grafov. Niektoré sú vyvíjané už viac ako 10 rokov[1].

Avšak každá z nich má určité unikátne funkcie alebo špecifické prístupy, ktoré môžu byť rozhodujúce pre konkrétny projekt alebo potreby programátora.

2.2.1 Aktuálne možnosti

Táto sekcia obsahuje stručný prehľad dostupných knižníc, ktoré sú vhodné na použitie v rámci aplikácie. Vyzdvihnuté sú predovšetkým silné stránky, ktoré danú knižnicu odlišujú od konkurencie.

- **Matplotlib**

  Matplotlib je jednou z prvotných vizualizačných knižníc v jazyku Python. Jej široké prijatie komunitou odráža nielen dlhú históriu, ale aj schopnosť spoľahlivo plniť potreby užívateľov. Matplotlib vyniká predovšetkým svojou flexibilitou, čo potvrdzuje rozsiahla dokumentácia a galéria[2] stoviek rôznych grafov. Súčasť, alebo skôr nadstavba nad Matplotlib je knižnica Seaborn poskytujúca vyššiu úroveň abstrakcie a špecializované nástroje na štatistickú vizualizáciu.

- **Bokeh**

  Bokeh je vizualizačná knižnica pre Python, ktorá je navrhnutá pre interaktívne webové aplikácie. Jej hlavnou prednosťou je teda interaktívnosť. Ďalšou dôležitou charakteristikou Bokehu je možnosť serverovej integrácie. Vďaka vlastnému serveru môže dynamicky aktualizovať vizualizácie v reálnom čase bez nutnosti obnovovať stránku alebo vykonávať ďalšie akcie. V neposlednom rade je Bokeh optimalizovaný pre prácu s rozsiahlym množstvom dát. Zameriava na to, aby zobrazovanie a vizualizácia boli výkonné.

---

1. Vprípade knižnice Matplotlib je možné hovoriť až o desaťročiach, keďže prvotné vydanie sa datuje ešte do roku 2003.
2. https://matplotlib.org/stable/gallery/index.html

- **Plotly**

  Plotly je všestranná vizualizačná knižnica, ktorá poskytuje podporu pre viaceré programovacie jazyky, vrátane Pythonu, R a Julie. Táto univerzálnosť uľahčuje jej integráciu do rôznych vývojových prostredí a umožňuje širokú škálu použitia. Jednou z kľúčových vlastností Plotly je Dash, inovatívna platforma určená na vývoj interaktívnych webových aplikácií. Dash výrazne zjednodušuje proces tvorby komplexných vizualizačných rozhraní tým, že eliminuje potrebu pre hlboké znalosti v oblasti webového vývoja.

- **Grafana**

  Grafana je široko uznávaná pre svoje schopnosti v oblasti monitorovania a vizualizácie metrík, ktoré sú kľúčové pre aplikácie vyžadujúce neustále sledovanie v reálnom čase. Umožňuje rýchlu identifikáciu a riešenie problémov, čím výrazne prispieva k efektívnemu prevádzkovaniu systémov. Jej silná stránka spočíva v schopnosti kombinovať dáta z rôznych zdrojov na jednom mieste. Grafana tiež umožňuje tvorbu tzv. interaktívnych dashboardov, ktoré poskytujú intuitívny prístup k dátam.

2.2.2 Kritéria hodnotenia

Pri hodnotení knižníc je nevyhnutné zamerať sa na niekoľko kľúčových aspektov, aby bolo zabezpečené, že zvolená knižnica bude vyhovovať požiadavkám aplikácie. Nasledujúce kritériá poskytujú rámec pre posúdenie a výber najvhodnejšej knižnice:

1. Interaktivita
2. Dokumentácia
3. Webové rozhranie
4. Flexibilita
5. Komunitná podpora

Výber správnej knižnice je zložitým rozhodnutím, ktoré zahŕňa technickú vhodnosť a tiež zhodu so skúsenosťami a preferenciami programátora. Po zvážení všetkých týchto faktorov som sa rozhodol použiť knižnicu Plotly.

Na základe zadaných požiadaviek v kapitole 1, interaktivita predstavuje kľúčové kritérium. Je žiadúce, aby knižnica podporovala interaktívne funkcie ako sú približovanie, škálovanie a filtrovanie dát. Knižnica Plotly spĺňa tieto požiadavky s rozsiahlym spektrom možností pre dynamické webové vizualizácie.

Dokumentácia[3] knižnice Plotly je jednou z najprehľadnejších, poskytuje návody a ukážky, ktoré sú k dispozícii nielen pre Python, ale aj pre iné programovacie jazyky ako JavaScript, R, Julia a MATLAB, čo umožňuje vývojárom vybrať si preferovaný jazyk a zároveň zachovať konzistentnosť vo vizualizáciách. Významnou pridanou hodnotou je platforma Dash, ktorá je detailne popísaná v nasledujúcej sekcii.

Okrem toho, robustná komunitná podpora Plotly s aktívnymi fórami, častými aktualizáciami a širokou užívateľskou základňou zaručuje, že akékoľvek problémy alebo otázky, ktoré vzniknú pri vývoji, môžu byť rýchlo a efektívne riešené. Tento faktor je kritický pre udržateľnosť a dlhodobú údržbu aplikácie, pretože zaistí, že aplikácia zostane aktualizovaná a kompatibilná s najnovšími technológiami.

2.3 Frontend

Jedným z hlavných dôvodov, prečo som sa rozhodol použiť knižnicu Plotly, je schopnosť vytvárania interaktívnych webových aplikácií prostredníctvom platformy Dash. Názov Dash vychádza z anglického slova dashboard. Je vyvíjaný rovnakým tímom ako Plotly a umožňuje kombinovať vizualizácie s interaktívnymi ovládacími prvkami a webovým rozhraním. Pod povrchom Dash využíva webový framework Flask.

Flask je flexibilný mikro webový framework, ktorý poskytuje základné nástroje potrebné na postavenie webovej aplikácie. Medzi tieto nástroje patrí napríklad automatické načítavanie kódu, chybové hlásenia, smerovanie požiadaviek (anglicky *routing*), či správa relácii. Dash ponúka tieto ale aj mnohé iné integrované vývojové nástroje

3. https://plotly.com/python/

pod pojmom Dash Dev Tools[4], ktoré sú určené výhradne pre lokalný vývoj s cieľom zvýšenia efektivity a rýchlejšieho ladenia počas vývoja aplikácie. Aj keď spustenie Dash aplikácie v zásade znamená spustenie Flask serveru, v produkčnom prostredí sa odporúča prechod na robustnejšie WSGI (*Web Server Gateway Interface*) servery.

Dash ponúka možnosť pre vývojárov, ktorí nemajú hlboké znalosti frontendových technológií, aby mohli vytvárať interaktívne webové aplikácie bez potreby písať (niekedy) zložitý frontendový kód.

Základným stavebným kameňom Dash sú tzv. komponenty, vysoko úrovňové triedy, ktoré sú preložené do HTML, CSS a JavaScriptu pri ich vykresľovaní vo webovom prehliadači. Tieto komponenty disponujú rôznymi vlastnosťami a metódami, ktoré definujú ich správanie a vzhľad. Jednou z ich najdôležitejších charakteristík je schopnosť uchovávať svoj vnútorný stav, ktorý, pri zmene vyvolá aktualizáciu alebo načítanie daného komponentu. Komponenty môžu byť ľubovoľne vnorené alebo kombinované, čo výrazne prispieva k flexibilite a znovu-použiteľnosti kódu.

Tento koncept je veľmi podobný tomu, ako funguje React (JavaScript knižnica pre vývoj užívateľských rozhraní). React takisto umožňuje písať užívateľské rozhrania v zapuzdrených komponentoch, ktoré si spravujú svoj vlastný stav. V skutočnosti Dash mapuje svoje komponenty na React komponenty jedna k jednej. Dokonca je možné používať celý ekosystém Reactu v rámci Dash aplikácie [2].

Nasleduje krátka demo ukážka, ako tieto komponenty fungujú.

4. https://dash.plotly.com/devtools

```
from dash import Dash, Input, Output, callback,
    dcc, html

app = Dash(title="Example")

# create a layout from various components
app.layout = html.Div(
    [
        html.H1(...),
        dcc.Dropdown(id="dropdown-selection",
            options=...),
        dcc.Graph(id="graph-content"),
    ],
)

# interaction between two components
@callback(
    Output("graph-content", "figure"),
    Input("dropdown-selection", "value"),
)
def update_graph(value):
    ...

if __name__ == "__main__":
    app.run(debug=True)
```

2.4 Interakcia s databázou

Pre spoľahlivú a efektívnu interakciu s databázou som sa rozhodol použiť kombináciu administratívnej platformy PgAdmin pre jednoduchšie pochopenie štruktúry databázy a Psycopg ako SQL adaptér na pripojenie k databáze a získavanie dát.

PgAdmin teda predstavuje administratívnu a vývojovú platformu pre PostgreSQL. Je dostupná aj ako samostatný balíček v rámci Python ekosystému, čo výrazne zjednodušuje jej použitie aj keď nie je kritickou súčasťou aplikácie ako priama závislosť, ide skôr o doplnkový nástroj počas vývoja.

Psycopg je najpopulárnejší PostgreSQL databázový adaptér pre programovací jazyk Python. Je napísaný v jazyku C. Jeho hlavnými vlastnosťami sú bezpečná práca s vláknami a úplná implementácia špecifikácie Python Database API Specification v2.0. [3]

Toto API (rozhranie) reprezentuje akýsi štandard, ktorý umožňuje vytváranie konzistentných a prenositeľných databázových aplikácií nezávisle od použitého databázového systému. Špecifikácia definuje rozhranie metód a objektov, ktoré musia byť implementované v každom module databázového adaptéra. Prístup k databáze je realizovaný pomocou tzv. objektu pripojenia (anglicky *connection object*) cez ktorý prebieha všetka komunikácia s databázou. Objekt pripojenia disponuje metódami na správu transakcií, vrátane ich potvrdzovania alebo vracania, a zodpovedá za vytváranie kurzorov. Kurzor, ako hlavný prostriedok pre prácu s dátami, umožňuje vykonávanie SQL príkazov a získavanie výsledkov. Špecifikácia ďalej definuje sadu štandardných výnimiek, ktoré umožňujú jednotné ošetrenie chýb pri práci s databázami, voliteľné rozšírenia a tak ďalej. Toto rozhranie zabezpečuje, že aplikácie môžu vykonávať základné databázové operácie s unifikovanou metodikou [4].

Nasleduje krátka demo ukážka, bez použitia konkrétnej knižnice.

```
# a library that conforms to Python DB API 2.0
from ... import connect

connection = connect(
    database=...,
    user=...,
    password=...,
)
cursor = connection.cursor()

query = """
        INSERT INTO students
        (first_name, last_name)
        VALUES ('Michal' 'Soltis');
        """
cursor.execute(query)
connection.commit()

query = """
        SELECT * FROM students;
        """
cursor.execute(query)
students = cursor.fetchall()
for student in students:
    print(student)

cursor.close()
connection.close()
```

3 Implementácia

Táto kapitola sa zaoberá hlavými fázami implementácie aplikácie. Implementačná časť stavia na použitých technológiách špecifikovaných v kapitole 3.
Aplikácia je pomyselne je rozdelená do 3 vrstiev:

1. databázová vrstva
2. logická vrstva
3. užívateľské rozhranie

3.1 Databázová vrstva

Prvotným krokom pri procese vizualizácie dát je ich efektívne a rýchle načítanie z databázy. Všetky SQL príkazy sú umiestnené v samostatnej zložke.

Väčšina programovacích kurzov používajúcich systém Frag majú rovnaký systém bodovania, čo umožňuje používať univerzálny SQL príkaz na extrakciu dát. Tento prístup zjednodušuje a zrýchľuje prácu s databázou a zároveň zabezpečuje konzistentnosť dát naprieč rôznymi kurzmi. Pre kurzy, ktoré tento systém bodovania nepoužívajú, je logicky potrebné vytvoriť nový SQL príkaz.

Keďže každý kurz používa vlastnú databázu, je nutné pre každý z nich vytvoriť nové databázové spojenie. V súbore *db.py* sa nachádza kontextový správca (anglicky *context manager*), trieda s názvom `NewDatabaseCursor`.

Kontextový správca je konštrukcia špecifická pre jazyk Python, ktorá umožňuje alokovať zdroje na začiatku bloku a automaticky uvoľniť tieto zdroje po ukončení bloku. Pre dosiahnutie tejto funkcionality je potrebné definovať v danej triede dve metódy s názvom `enter` a `exit`. Metóda `enter` alokuje potrebné zdroje a metóda `exit` ich uvoľní. Detailnejšie informácie sú obsiahnuté v dokumentácii[1] jazyka Python.

Práve trieda `NewDatabaseCursor` umožňuje automatické vytváranie a uzatváranie databázových spojení a kurzorov, čím sa znižuje riziko chýb spojených s riadením týchto spojení.

1. `https://book.pythontips.com/en/latest/context_managers.html`

Na pripojenie k databáze a načítanie dát som použil knižnicu *psycopg2*. Táto knižnica poskytuje všetky základné funkcie potrebné na interakciu s databázou, hoci existuje aj novšia verzia *psycopg3* s modernejšími vlastnosťami a dizajnom. Avšak pre účely tejto aplikácie bohate postačuje *psycopg2*.

3.2 Logická vrstva

Logická vrstva aplikácie predstavuje spojovací prvok medzi databázovou vrstvou a užívateľským rozhraním. V tejto vrstve sa dáta získané z databázy transformujú do štruktúrovaného a lepšie pochopiteľného formátu a aby boli optimálne usporiadané pre následnú vizualizáciu. V oblasti vizualizácie grafov sa tento proces často uskutočňuje prostredníctvom objektu `DataFrame` (dátový rámec).

`DataFrame` je dvojrozmerná dátová štruktúra, ktorá transformuje vstupné dáta ako tabuľku s riadkami a stĺpcami a inými metadátami, podobne ako tabuľka v relačnej databáze. Z hľadisku Pythonu sú tieto dáta definované ako parameter rôzneho typu, najčastejšie ako slovník. `DataFrame` je dostupný v knižniciach ako *pandas*[2] alebo *polars*[3], ktoré sa vyznačujú vysokou popularitou v oblasti dátovej analýzy.

Ukážkový kód nižšie demonštruje jednoduché vytvorenie dátového rámca pomocou knižnice *polars*:

2. `https://pandas.pydata.org`
3. `https://www.pola.rs`

```
import polars as pl

data = {
    "student": [
        "xsoltis1",
        "xsoltis2",
        "xsoltis3",
    ],
    "b1_points": [10, 15, 5],
    "b2_points": [15, 5, 10],
    "b3_points": [5, 10, 15],
}
df = pl.DataFrame(data)
print(df)

# output
shape: (3, 4)
-------------------------------------------------
| student   | b1_points | b2_points | b3_points |
| ---       | ---       | ---       | ---       |
| str       | i64       | i64       | i64       |
|===============================================|
| xsoltis1  | 10        | 15        | 5         |
| xsoltis2  | 15        | 5         | 10        |
| xsoltis3  | 5         | 10        | 15        |
-------------------------------------------------
```

Využitím tohto štruktúrovaného formátu je ďalej možné efektívne manipulovať s dátami a preniesť ich do vizuálnej formy ako napríklad stĺpcový graf, kde každý stĺpec reprezentuje súčet získaných bodov daného študenta v priebehu semestra.

3.2.1 Vyrovnávacia pamäť

Komunikácia s databázou je pomerne drahá operácia, ktorá často prichádza s určitou mierou latencie. Tento problém je zvlášť výrazný pri spracovaní rozsiahlych dát, ako je spracovanie výsledkov veľkého po-

čtu študentov počas semestra. Jedným zo spôsobov, ako tento problém riešiť, je využitie vyrovnávacej pamäte (anglicky *cache*).

Vyrovnávacia pamäť slúži na ukladanie výsledkov funkcií podľa unikátneho kľúča, ktorý reprezentuje vstupné argumenty danej funkcie. Pri opakovanom volaní funkcie s rovnakými argumentami sa najprv skontroluje, či už výsledok s daným kľúčom v pamäti existuje. Ak áno, výsledok sa okamžite vráti, čím sa ušetrí čas potrebný na opätovné spracovanie. V prípade, že výsledok pre daný kľúč v keši neexistuje, funkcia sa vykoná a jej výsledok sa uloží do vyrovnávacej pamäti, čím sa optimalizuje výkon pre budúce volania.

Dash poskytuje vlastné mechanizmy v tejto oblasti, ktoré sú integrované s tzv. *callbackami* - funkcie, ktoré sa spustia po zmene stavu definovaného komponentu. Zvyčajne sa tak deje po interakcii užívateľa s aplikáciou. Značnou nevýhodou je nízka flexibilita. Takisto nie vždy je nutné ukladať do vyrovnávacej pamäti výsledok volania *callbacku*.

Hoci jednou z možností, ktoré ponúka Dash je kombinácia Celery a Redis, rozhodol som sa použiť individuálny prístup k využitiu Redisu z dôvodu väčšej flexibility a kontroly nad procesom kešovania. Ďalšou výhodou je možnosť pristupovať k uloženým dátam aj po reštartovaní procesu aplikácie, čo znamená, že spracované dáta zostávajú dostupné a ihneď k dispozícii až po dobu expirácie. Redis ako taký môže byť použitý nielen ako vyrovnávacia pamäť, ale aj ako databáza, streamovací prostriedok, či sprostredkovateľ správ [5].

3.3 Užívateľské rozhranie

Užívateľské rozhranie je kľúčovou vrstvou v celej aplikácii, ktorého hlavným cieľom je poskytnúť užívateľom prehľadný, intuitívny a efektívny spôsob interakcie. Aplikácia využíva knižnicu Dash, ktorá je postavená na webovom frameworku Flask. Tieto technológie sú detailne popísané v kapitole 2.

Základom tohto rozhrania je koncept už spomínaného dashboardu. Vstupným bodom celej aplikácie je súbor *main.py* ktorá vytvára tzv. layout. Tento layout je zostavený z rôznych komponentov, uložených v zložke *components*. Každý komponent je definovaný samostatne pre zvýšenie prehľadnosti a modularity kódu. Pre vylepšenie vizuálneho aspektu rozhrania aplikácia využíva *dash-mantine-components*,

ktoré rozširujú štandardné komponenty v Dash v kontexte štýlovania. Ako alternatívu k *dash-mantine-components* existuje aj *dash-bootstrap-components*, poskytujúce podobné možnosti štýlovania.

Centrálnym prvkom rozhrania je vizualizácia dát prostredníctvom stĺpcového grafu. Vďaka integrovaným funkcionalitám knižníc Dash a Plotly, je možné s grafom pracovať vysoko interaktívne, čo zahrňuje približovanie, filtrovanie jednotlivých blokov, exportovanie grafu do obrázku, navigačné prvky a podobne. Každý blok dát je rozlíšený unikátnou farbou, ktorú si môže užívateľ prispôsobiť v súbore *settings.py*, pričom jednotlivé bodové sekcie sú rozlíšené odtieňmi danej farby.

Figúra 3.1 ilustruje stĺpcový graf (exportovaný ako obrázok) s reálnymi dátami.

3.4 Rozšíriteľnosť

Rozšíriteľnosť je dôležitým aspektom softvéru, ktorý zabezpečuje, že aplikácia môže byť v budúcnosti jednoducho upravovaná alebo rozširovaná o nové funkcionality. V kontexte tejto aplikácie, to znamená poskytnúť možnosť pridávania nových predmetov do systému s minimálnym úsilím a bez narušenia existujúcej funkcionality. Nie všetky predmety, ktoré používajú systém Frag, majú rovnaký systém bodovania, čo predstavuje výzvu pri implementácii aplikácie vzhľadom na jednoduchú rozšíriteľnosť.

Na začlenenie nového predmetu do aplikácie je potrebné vykonať nasledujúce kroky:

1. Napísať nový SQL príkaz.

   SQL príkaz, ktorý bude extrahovať potrebné dáta pre nový predmet z databázy. Súbor, respektíve príkaz by mal byť umiestnený v rovnakej zložke ako ostatné príkazy pre zachovanie peknej štruktúry projektu.

2. Napísať novú funkciu na vygenerovanie grafu.

   Na pridanie nového predmetu je potrebné vytvoriť alebo upraviť existujúce funkcie v súbore *figures.py*, zodpovedné za spracovanie a vizualizáciu dát špecifických pre tento predmet.

3. Aktualizovať komponent v užívateľskom rozhraní.

   Pri pridávaní nového predmetu je nevyhnutné upraviť funkciu definovanú v súbore *graph.py*, ktorá je zodpovedná za aktualizáciu grafov v aplikácii na základe vstupných parametrov od užívateľa.

4. Pridať nový predmet to užívateľského rozhrania.

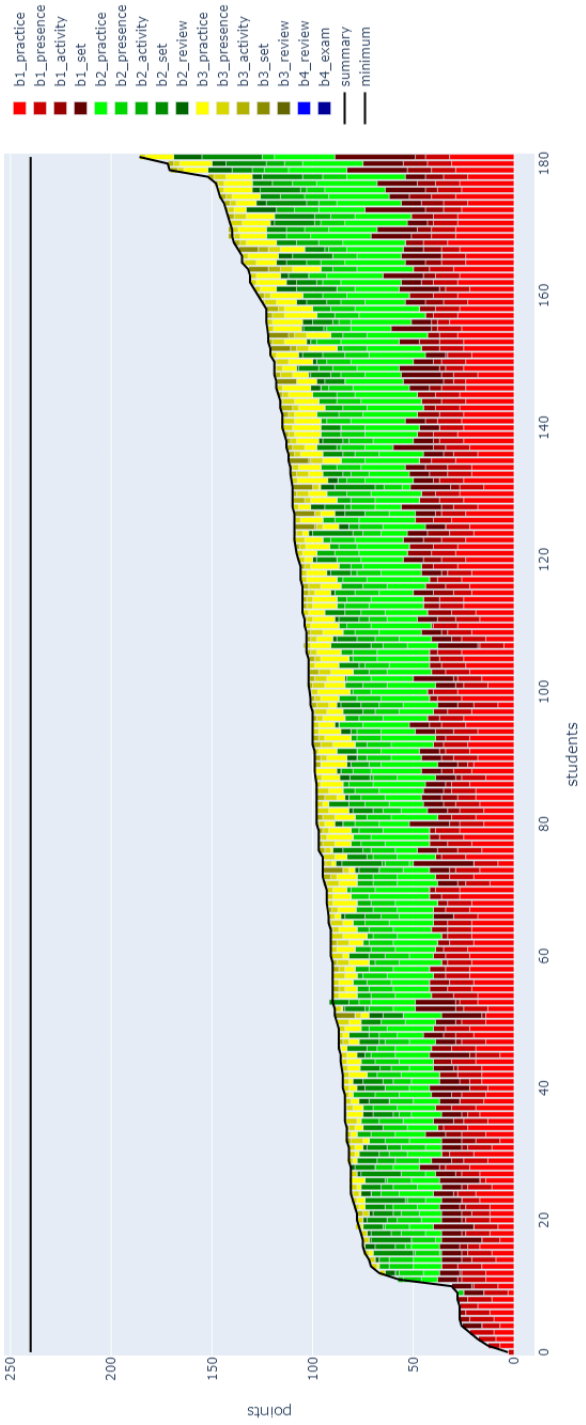
   V súbore *settings.py* pridať do zoznamu všetkých kurzov respektíve aktívnych kurzov názov predmetu, čím sa rozumie názvov databázy v rámci systému Frag, ktorá reprezentuje nový predmet. Typicky kód predmetu v informačnom systéme.

5. Upraviť užívateľského rozhranie nového predmetu.

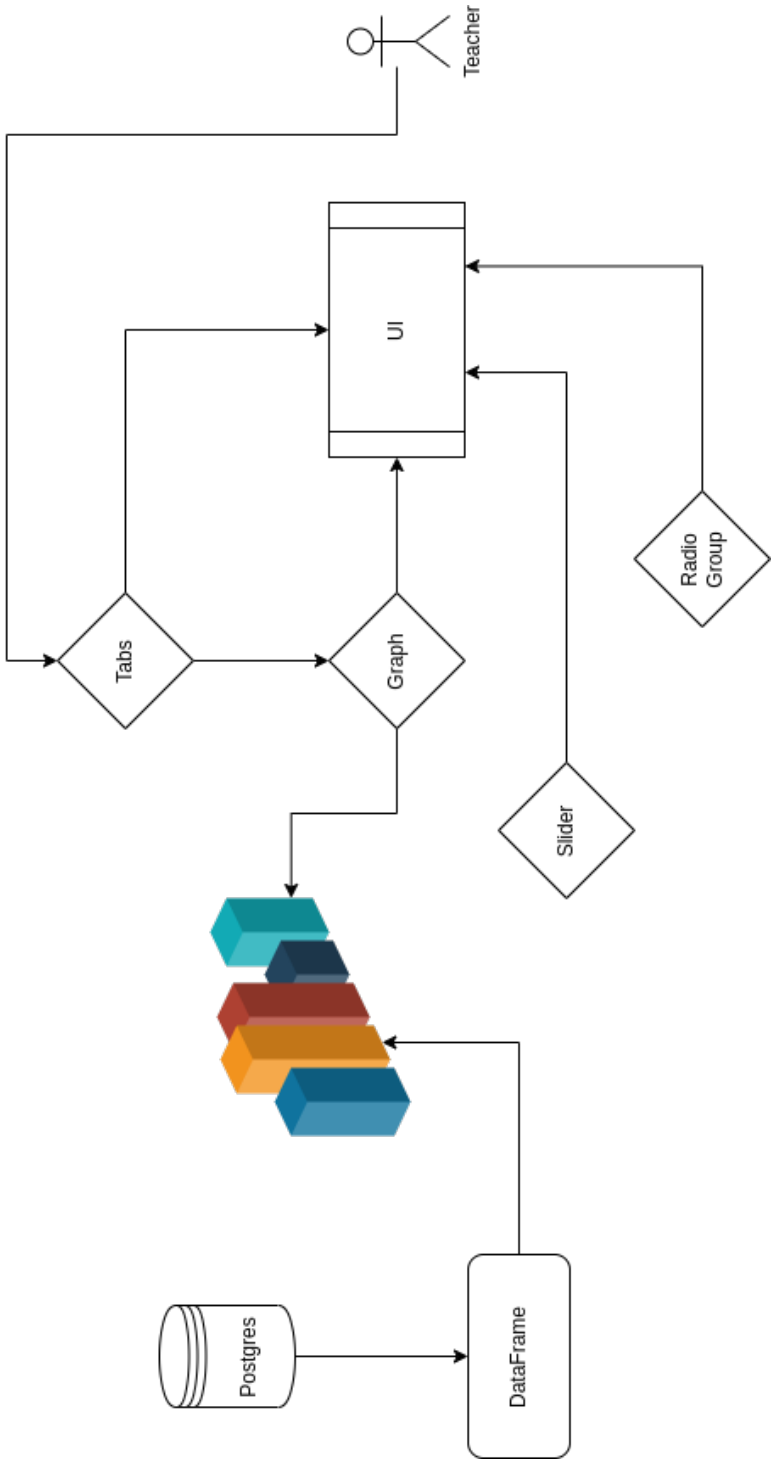
   Nový predmet môže vyžadovať špecifické komponenty alebo zmeny v užívateľskom rozhraní. To môže znamenať už výraznejšie zmeny vo viacerých súboroch.

3.5 Zhrnutie

Figúra 3.2 zobrazuje zjednodušený diagram architektúry aplikácie s dôrazom na tok a vizualizáciu dát. Na ľavej strane diagramu je databázová vrstva, symbolizovaná ikonou PostgresSQL databázy, odkiaľ dáta pomyselne smerujú do dátového rámca, čo predstavuje ich transformáciu do štruktúrovanej formy pripravenej na analýzu a vizualizáciu. Tieto štruktúrované dáta sú potom vizualizované ako stĺpcový graf, ktorý je nezávislým elementom v rámci komponentov užívateľského rozhrania, t.j. graf je vstupným parametrom Dash komponentu. Graf sa dynamicky aktualizuje prostredníctvom interakcie užívateľa s aplikáciou vo webovom prehliadači, reprezentovanej obdĺžnikom v pravom hornom rohu. Po zmene stavu niektorej z komponentov, napríklad po stlačení tlačidla myši na iný kurz, sa vygeneruje graf s novými dátami na základe požadujúcej zmeny.



Obr. 3.1: Štatistika bodov PB152



Obr. 3.2: Diagram toku dát

4 Nasadenie aplikácie

Táto kapitola sa zaoberá nasadením aplikácie na serveroch fakulty informatiky, konkrétne na platforme Stratus.

4.1 Stratus

Stratus (Stratus.FI) je privátne cloudové úložisko poskytované fakultou informatiky, postavené na softvéri OpenNebula[1]. Stratus ponúka flexibilné prostredie, ktoré sa hodí nielen na experimentovanie a rýchle testovanie, ale aj na produkčné použitie softvéru, ktorý z rôznych dôvodov nemôže byť inštalovaný priamo na serveroch spravovaných fakultou. K dispozícii sú už predkonfigurované virtuálne stroje, ktoré sú ihneď pripravené na použitie a sú vybavené všetkým nevyhnutným softvérom. Po aktivácii virtuálneho stroja je možné sa naň pripojiť cez SSH (anglicky *Secure Shell*) protokol v rámci privátnej siete fakulty. Každý vytvorený virtuálny stroj má pridelenú privátnu IPv4 adresu a verejnú IPv6 adresu [6].

4.2 Konfigurácia

Pred samotným nasadením aplikácie je dôležité vybrať produkčný Python WSGI (*Web Server Gateway Interface*) server ktoré zabezpečí jej spoľahlivý a efektívny beh. V oblasti UNIXových operačných systémov je vysoko populárny server Gunicorn, ktorý je široko kompatibilný s rôznymi webovými frameworkami, je nenáročný na serverové zdroje a je pomerne rýchly. Často sa používa spolu so serverami ako Nginx alebo Apache, predovšetkým kvôli doplnkovým funkciám ako HTTP proxy, HTTP keš, load balancing a podobne [7].

Nasadenie aplikácie vyžadovalo tiež správnu konfiguráciu sieťového firewallu. Je nevyhnutné explicitne povoliť prichádzajúce spojenia na príslušnom porte, aby bola aplikácia dostupná aj učiteľom alebo študentom. To je možné docieliť napríklad pomocou nástroja *firewalld*.

---

1. `https://opennebula.io/`

```
$ firewall-cmd \
      --permanent \
      --zone=public \
      --add-service=http
```

Tento príkaz, otvára HTTP službu vo verejnej zóne. Pre HTTP aplikácie je to typicky port 80, alebo, v prípade zabezpečeného HTTPS spojenia, port 443. Použitie prepínača *permanent* znamená, že zmeny zostanú zachované aj po reštarte služby alebo systému. Aplikácia potom môže byť prístupná na pridelenej IP adrese virtuálnych stroja na platforme Stratus, bez nutnosti, špecifikovať port v prípade, že samotná aplikácia beží na porte 80. V opačnom prípade, je nutné opäť explicitne povoliť prichádzajúce spojenia pre príslušný port, ako je podrobne opísané v dokumentácii *firewalld*[2].

2. https://firewalld.org/documentation/howto/open-a-port-or-service.html

Záver

Táto bakalárska práca sa zaoberala návrhom a implementáciou webovej aplikácie na vizualizáciu bodov študentov v rôznych predmetoch na fakulte informatiky. Aplikácia bola vyvinutá s využitím moderných programovacích nástrojov a technológií, ako sú Python, Dash, Plotly a Redis.

Hlavným cieľom práce bolo poskytnúť učiteľom lepší prehľad o výkonnosti a pokroku študentov v priebehu semestra. Tento cieľ bol úspešne dosiahnutý.

V závere tejto práce chcem poukázať na to, že podobné aplikácie ako je táto môžu predstavovať krok smerom k efektívnejšiemu vzdelávaciemu procesu. Dúfam, že výsledky a skúsenosti získané v rámci tejto práce poslúžia ako inšpirácia pre ďalšie vývojárske projekty v tejto oblasti.

Bibliografia

1. *Chartio* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://chartio.com/learn/charts/stacked-bar-chart-complete-guide/.
2. *React for Python Developers* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://dash.plotly.com/react-for-python-developers.
3. *Psycopg documentation* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://www.psycopg.org/docs/index.html.
4. *Python Database API Specification v2.0* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://peps.python.org/pep-0249/.
5. *Redis* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://redis.io/.
6. *STRATUS.FI* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://www.fi.muni.cz/tech/unix/stratus.html.cs.
7. *Gunicorn - Wikipedia* [online]. 2023. [cit. 2023-11-01]. Dostupné z : https://en.wikipedia.org/wiki/Gunicorn.

A Príloha

Archív obsahujúci zdrojový kód celého projektu je priložený ako elektronická príloha bakalárskej práce.

Pre správne spustenie a konfiguráciu projektu je nevyhnutné upozorniť na súbor *pyproject.toml*, ktorý špecifikuje všetky metadáta, vrátane názvu projektu, verzie, závislostí a vstupného bodu programu. Použitie súboru *pyproject.toml* v súlade so štandardom PEP 621[1] (*Python Enhanced Proporsal*) zabezpečuje konzistentnú štruktúru a kompatibilitu s ďalšími nástrojmi a knižnicami, ktoré tento štandard dodržujú.

Vo vnútri zložky *src* sa nachádza užívateľsky orientovaný konfiguračný súbor *settings.py*. Tento súbor slúži na definovanie konfiguračných premenných a nastavení, ktoré ovplyvňujú funkčnosť a vzhľad grafu v aplikácii. Jeho prítomnosť umožňuje užívateľom ľahko meniť parametre vizualizácie bez potreby zasahovať do dôležitých častí kódu aplikácie.

1. `https://peps.python.org/pep-0621/`