Jihočeská univerzita v Českých Budějovicích, Branišovská 1645/31A

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

bakalářská práce

Vývoj desktopové aplikace pro sledování časových intervalů využitých při práci na projektech a úkolech

Vypracovala: Marie Džuganová

Vedoucí práce: Mgr. Radim Remeš Ph.D.

České Budějovice 2022

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Marie DŽUGANOVÁ
Osobní číslo:	E19721
Studijní program:	B6209 Systémové inženýrství a informatika
Studijní obor:	Ekonomická informatika
Téma práce:	Vývoj desktopové aplikace pro sledování časových intervalů využitých při práci na projektech a úkolech
Zadávající katedra:	Katedra aplikované matematiky a informatiky

Zásady pro vypracování

Cílem práce je vytvořit desktopovou aplikaci, která bude sledovat čas využitý při práci na projektech a úkolech. Aplikace bude umožňovat propojení na účty sociálních sítí a bude nabízet zobrazení přehledů využitého času za různá časová období a další vybraná statistická data. Aplikace bude vyvíjena v programovacím jazyce Python s využitím dalších dostupných knihoven.

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Závěr.

Rozsah pracovní zprávy:	40 – 50 stran
Rozsah grafických prací:	dle potřeby
Forma zpracování bakalářské práce:	tištěná

Seznam doporučené literatury:

1. Bonzanini, M. (2016). *Mastering Social Media Mining with Python*. Birmingham, UK: Packt.
2. Holden, S., Ravenscroft, A., & Martelli, A. (2017). *Python in a Nutshell*. Sebastopol, CA (USA): O'Reilly.
3. Lubanovic, B. (2019). *Introducing Python*. Sebastopol, CA (USA): O'Reilly.
4. Phillips, D. (2018). *Python 3 Object-Oriented Programming*. Birmingham, UK: Packt.
5. Pilgrim, M. (2009). *Dive into Python 3*. New York, USA: Apress.

Vedoucí bakalářské práce:	Mgr. Radim Remeš Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: 25. března 2021
Termín odevzdání bakalářské práce: 15. dubna 2022

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 31. března 2021

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své [typ] práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum: 14.4. 2022

Podpis studenta

PODĚKOVÁNÍ

Tímto bych ráda poděkovala panu doktorovi Mgr. Radimu Remešovi za poskytnuté rady a připomínky během vypracovávání bakalářské práce.

Také chci poděkovat Tomáši Čeloudovi za jeho rady a čas, který mi věnoval při vývoji aplikace.

OBSAH

1. Úvod

Tématem bakalářské práce je vývoj desktopové aplikace v programovacím jazyce Python. Hlavní funkcí aplikace je odpočet a uchování časových záznamů o studiu.

První část práce se zabývá základním rozčlením desktopových aplikací, jejich historickým vývojem, jsou zde popsány nejpoužívanější programovací jazyky současnosti. V poslední řadě též popisuji, jakým způsobem je tvořeno grafické rozhraní v aplikacích vyvíjených v jazyce Python a princip gamifikace, který je ve vytvořené aplikaci též obsažen.

Následně se práce věnuje popisu jednotlivých knihoven, které jsou využívány pro správnou funkcionalitu aplikace, například *Tweepy, Datetime*, či *SQLite.*

Třetí část je zaměřena na praktický aspekt bakalářské práce, tedy obsahuje především zdrojový kód vytvořené aplikace ve formě obrázků a jeho slovní popis.

1.1. Cíle aplikace

Důležitou součástí kvality studijních návyků je vytvoření si určitého systému učení a organizovat si svůj čas společně se studijní činností. Stanovení jasného plánu a časového rozvrhu může pomoci při problémech s prokrastinací, či špatnou organizací času, nebo nesoustředěnosti, což vede k neefektivnímu využívání času. Pro studium je vhodné rozdělit si své úkoly, či množství látky, do několika částí, jelikož mozek je schopný plně se soustředit pouze pár desítek minut. Poté se pozornost začíná snižovat a studium začíná být neefektivní. V tomto ohledu je vhodné si určit menší cíle, které například budou ve větším počtu. Monitorovat tyto krátké časové úseky může být občas obtížné, je složité záznamy dlouhodobě uchovávat na jednom místě, nepřehledné a nepřesné.

Aplikace vznikla, aby studentům zjednodušila plánování svého času a umožnila jim lepší přehled o tom, jak svůj čas kvalitně využívají, a to pomocí ukládání časových intervalů a následného sčítání celkového času. Studenti tedy přesně ví, kolik času věnovali aktivnímu studiu.

Právě nepřehlednost a nepřesnost záznamů, spolu se složitým uchováváním aplikace vyřeší za uživatele, je možné si přesně změřit čas, který je následně ukládán do databáze.

Zároveň je aplikace uzpůsobena tak, aby uživatele motivovala k co nejčastějšímu studování, díky přidané gamifikaci. Tedy-čím více bude uživatel studovat, tím více mincí v aplikaci získá. Tyto mince následně bude moci využít k zakoupení předmětů v obchodu aplikace. Předměty se mu následně uloží v inventáři.

Gamifikace se v posledních letech ukázala jako prospěšná v mnoha případech. Obecně dochází při jejím využití k vyšší efektivitě a rozvoji jedince. Využití herních prvků je zaměřeno na přirozenou potřebu člověk dosahovat cílů a překonávat sebe či ostatní a tím nenásilnou formou zvyšovat svoji výkonnost pomocí her. Člověk, kterého více baví práce nebo studium se cítí lépe. Pokud se jedná o promyšlený systém, který s člověkem nemanipuluje, ale dává mu možnost hravě růst, pak je to skvělý nástroj a měl by být využíván co nejvíce. Právě z těchto důvodů je složka gamifikace v aplikaci také zahrnuta. (Gamifikace a její využití ve vzdělávání 2022)

Součástí aplikace je také propojení se sociální sítí Twitter pro zvýšení užitné hodnoty aplikace pro uživatele. K této funkci je využita API, která umožní přístup k datům ze sociální sítě.

2. Aplikační software

Aplikační software je v informatice programové vybavení počítače, umožňující provádět užitečné činnosti, jako je například řešení konkrétního problému, nebo interaktivní tvorbu uživatele. Aplikace využívají pro interakci s uživatelem grafické, nebo textové rozhraní, případně příkazový řádek. Dnes je již ale běžnému uživateli umožněno komunikovat s aplikací pomocí grafického rozhraní. Aplikace se může skládat z několika počítačových programů, případně je několik aplikací spojeno do skupiny, která je následně nazývána aplikačním balíkem (například Microsoft Office). Mezi aplikace není zařazován systémový software, tedy jádro, a další součásti operačního systému (Windows apod.). (Aplikační software 2022)

Aplikace prošly od jejich zrodu výraznou proměnou, stávají se čím dál více uživatelsky orientované. Manipulace s textem, čísly, audiem, grafikou ale i kombinací těchto prvků, již není problém a většina programů je již dnes uzpůsobena tak, aby jejich ovládání zvládl i běžný uživatel osobních počítačů. (Aplikační software 2022)

Aplikace lze řadit do několika kategorií, mezi které patří například:

- Podle platformy
    - Aplikace v operačním systému na počítači
    - Aplikace spuštěná na serveru
    - Mobilní aplikace
- Podle účelu
    - Antivirový program
    - Databázový systém
    - Grafické editory
    - Ekonomické a informační systémy
    - Hry
    - Webové prohlížeče
    - Atd.
- Podle práv užití
    - Open-source
    - Closed-source

- Podle jazyka, ve kterém je aplikace vyvíjena
    - Python
    - C#
    - JavaScript
    - Atd.

(Aplikační software 2022)

Desktopová aplikace musí být nainstalována na harddisk počítače. Jedná se o nejtypičtější formu softwaru, se kterou se již dnes běžný uživatel technologií setkává každý den. (Koďousková 2021)

2.1. Platformy a aplikace

Počítačová aplikace je závislá na operačním systému. Nelze ji tedy spustit v jiném prostředí, než pro které byla při vývoji určena. V případě potřeby fungování na více platformách se vývoj značně komplikuje. V tomto ohledu má velkou výhodu webová aplikace, která je spustitelná na jakémkoliv prostředí, nicméně desktopová verze je stabilnější a uživatelsky příjemnější. (Koďousková 2021)

3. Desktopové aplikace

3.1. Historie

Vývoj počítačových aplikací byl ovlivněn několika událostmi. Aplikace je tvořena kódem, který je složen z algoritmů, o nichž první historická zmínka pochází až z 19. století. Byl sepsán Adou Lovelace, zabýval se výpočtem Bernulliniho čísel, nicméně, tehdejší fáze, ve které se inženýrství nacházelo, neumožnilo algoritmus prakticky využít. Zůstal tedy pouze v teoretické podobě. (Hammerman & Russel 2015)

První moderní teorie softwaru pochází z roku 1935, jejím autorem je Alan Turing, problém rozhodování a počitatelných čísel v aplikaci je popsán v jeho eseji, která následně vedla k vytvoření akademických oborů, které jsou dnes známé jako softwarové inženýrství a informatika. Obě akademické větve se zabývají softwarem a jeho vývojem, informatika se ale tímto tématem zabývá v teoretické hladině, zatímco softwarové inženýrství je soustředěno spíše na konkrétní praktické záležitosti. (Hally 2005)

Nicméně, až do roku 1946 software v takové podobě, jako je dnes známý, neexistoval. Až do příchodu ENIAC počítače. Na jeho vývoji se podílely především ženy, které při fyzicky nenáročných, kancelářských pracích do té doby osobní počítače nahrazovaly. Byly jim poskytnuty plány na zapojení ENIAC počítače, které tým vývojářů poté využil k tomu, aby zjistili, jak stroj naprogramovat. Mezi nimi se nacházela i Kathleen Booth, která roku 1950 vyvinula programovací jazyk Assembly, který měl za úkol výrazně ulehčit programování počítačů na univerzitě Birkbeck. (Evans 2018)

V roce 1948 byl autorem Claudem Shannonem vydán rukopis „Matematická teorie komunikace“, která se zabývá binární logikou a jejím možným implementováním do programů počítače. Jeho teorie byla následně několikrát vyzkoušena praxi, kdy se první programátoři postupně snažili použít binární kód, ale proces byl velmi obtížný. Binární kód, který bylo nutné napsat, aby počítač věděl, jaká data přesně uložit, byl velmi dlouhý a složitý. Data spolu s kódem byly do počítačů nahrávány pomocí různých zdlouhavých mechanismů, které, pokud nebyly vykonány přesně podle návodu, způsobily časté chyby, kvůli kterým bylo nutné nahrávat celý program znovu od začátku. (Tse 2020)

Úplně poprvé, kdy počítač s uloženým programem uchoval kus softwaru v elektronické paměti a úspěšně jej provedl, bylo 21. června roku 1948 na univerzitě v Manchesteru na počítači Manchester Baby. Autorem byl Tom Kilburn a úkolem programu bylo vypočítat nejvyšší faktor celého čísla 2^18 = 262 144. Počínaje velkým zkušebním dělitelem bylo provedeno dělení 262 144, opakovaným odečítáním a kontrolováním zbytku se proces úspěšně dokončil. (Yost 2018)

V této době se začaly objevovat první programovací jazyky, které se sdílely mezi více podniky. Mezi ně patří hlavně FORTRAN, který je vyvinut týmem v čele s Johnem Backusem v IBM v roce 1957. Hned o rok později byl vydán FORTRAN II, který podporoval procedurální programování, které umožnilo vývojářům vytvářet funkce, díky čemuž bylo možné kód aplikovat pro více funkcionalit. (Harper & Stockman 2022)

Popularita FORTRANu vedla k implementaci více verzí, které jednotliví výrobci informačních technologií používali pro vlastní produkty. Každý výrobce upravil kód jinak, takže nebylo zaručeno, že by jeden program určený pro jeden typ stroje fungoval i na jiných typech. Tento problém vyřešila společnost IBM odstraněním všech funkcí závislých na stroji ze své verze jazyka a tento programovací jazyk nazvala FORTRAN 66. (Harper & Stockman 2022)

Další významný přínos přišel roku 1970 ve formě Unixu. Jedná se o raný operační systém, který se stal velmi populárním, zvláště po jeho přepracování do programovacího jazyka C. Využívá se do určité míry dodnes. Nejoblíbenější variantou Unixu je dnes macOS. Jeho vliv je dnes možné vidět i na prostředí Linuxu, který na bázi UNIXu funguje dodnes. (History of UNIX How has it affected other systems 2022)

Rok 1980 odstartoval velký rozmach na trhu producentů operačních systémů, jako je Microsoft, Lotus Development a Apple. Tyto firmy dominovaly jak na americkém trhu, tak i v Evropě. Zásadní moment se odehrál při uvedení osobního počítače od společnosti IBM na trh. Společnost následně uzavřela smlouvu o spolupráci s Microsoft, která pro IBM vyvinula jejich první operační systém MS-DOS. Tato spolupráce výrazně pomohla Microsoftu získat skvělou pozici na trhu, kterou si Microsoft dokázal udržet až dodnes. (Yost 2018)

V devadesátých letech, s příchodem internetu, se objevily i první open-source aplikace. Linux Kernel, který je základem Linux OS byl vydán v roce 1991. Open-source aplikace dosáhly vrcholu popularity až ke konci dekády, s příchodem programovacího jazyka C a C++. (Yost 2018)

Zároveň se na trhu objevily aplikace jako AutoCad, Microsoft Word a Excel, které jsou velmi populární dodnes. (Yost 2018)

3.2. Vliv hardwaru na software

Postupem času s rozmachem programů a postupným zmenšováním hardwarových součástek docházelo ke snižování ceny elektroniky a výraznému zrychlení vývoje (ke kterému přispěl i internet v devadesátých letech).

Steve Jobs popsal důležitost softwaru při prvním představení iPhonu. Dle jeho názoru, jednou z mnoha výhod softwaru je jeho lehká přeměna. Tehdejší první dotykový telefon totiž umožnil přidání drobných funkcí i poté, co si jej zákazník zakoupil v obchodě, a to ve formě aktualizací telefonu. Dotyková obrazovka tehdy také umožnila i o poznání lehčí úpravu softwaru, jelikož různé funkce mohli být přidány kdykoliv, bez jakýchkoliv fixací na tehdejší tlačítka.

3.3. Nevýhody desktopových aplikací

V první řadě je nutné uvést hlavně navázání na jedno zařízení. Mezi poslední trendy patří hlavně využívání sdílených disků a úložišť, které umožňují připojení a přístup k veškerým datům uživatele odkudkoliv. Pokud ale aplikace tuto funkci neposkytuje, uživatel je dnes již značně omezován. (Koďousková 2021)

Dále mohou nastat také problémy v případě napadení softwaru. Počítačové programy jsou častým cílem hackerů, zvláště kvůli datům, která aplikace uchovává. Z tohoto důvodu jsou vydávány bezpečnostní aktualizace, které se instalují stejným způsobem jako nové verze softwaru. (Koďousková 2021)

Komplikované mohou být týmové spolupráce, mnoho aplikací dnes stále není uzpůsobeno na tento typ využívání, nicméně zájem o tyto funkce ze strany uživatelů roste, zvláště v posledních letech, kdy společnost byla nucena přesunout mnoho denních činností do online prostředí. (Koďousková 2021)

4. Významné osobnosti v oblasti technologického vývoje

4.1. Dan Bricklin a Bob Frankston – VisiCalc

VisiCalc byl první tabulkový procesor, vydaný v roce 1979 pro Apple II. Bylo snadné jej používat a umožňovalo třídění a ukládání dat v tabulkových řádcích a sloupcích. VisiCalc byl vytvořen, aby nahradil ruční zpracování dat. V aplikaci totiž znamenala změna jedné hodnoty, změnu i druhé závislé hodnoty, či rovnou celé tabulky. Bylo možné nastavení propojení buněk, které právě tyto navazující změny umožňovalo. Tento program byl jedním z hlavních důvodů, díky nimž byly osobní počítače přitažlivější pro každodenní zákazníky a společnosti, nikoli jen pro nadšence technologií. Demonstrovala hodnotu osobních počítačů pro malé podniky, protože dokázala zkrátit účtovací procesy, trvající několik hodin, na několik minut. (history-computer.com 2021)

Byla první aplikací označenou jako „killer app“, tedy zabijáckou aplikací, software, který je tak zásadní, že lidé byli ochotní koupit celý počítač, jen kvůli VisiCalc. Steve Jobs nakonec uznal, že právě díky VisiCalc dosáhl Apple II. takového úspěchu. (history-computer.com 2021)

Bricklin byl narozen v červenci roku 1951 ve Filadelfii, úspěšně vystudoval elektrotechniku na MIT, poté v roce 1977 začal se studiem na Harvardu. Již během jeho studií měl plno nápadů a byl si vědom, že po ukončení školy chce zůstat v oboru a prodávat produkty, které přispějí pro dobro společnosti. První prototyp obsahoval sloupce a řádky a základní aritmetiku. (history-computer.com 2021)

Brzy se k němu připojil i Frankston, který zajistil rychlost aplikace, lepší aritmetiku a další funkce, podařilo se mu program naprogramovat v Assembleru a výrazně snížil spotřebu paměti nutné k provozu. Po uvedení na trh firmou VisiCorp se aplikace velmi rychle stala populární. (history-computer.com 2021)

4.2. Tim Berners-Lee – World Wide Web

Sir Tim Berners-Lee vynalezl World Wide Web v březnu roku 1989. Tento vynález následně způsobil výrazné zrychlení v oblasti technologií, jelikož umožnil široké veřejnosti přístup neomezenému množství informací, které lze na internetu najít. (www.devtopics.com 2008)

První webová stránka byla Bernersem-Leem uveřejněna 6. srpna roku 1991. Stránka poskytovala vysvětlení o tom, co je World Wide Web, jak používat prohlížeč a jak nastavit webový server. Berners-Lee dal svůj nápad volně k dispozici, bez nároku na patent nebo licenční poplatky. World Wide Web je jedním z nejdůležitějších komunikačních vynálezů v historii a poskytuje standartní platformu pro globální komunikaci a obchod. Dnes existuje více než 100 milionů webových stránek. Hlavní myšlenka tohoto vynálezu byla založena na bezplatné technologii, aby je mohl snadno přijmout kdokoli. (www.devtopics.com 2008)

4.3. Bill Gates – Microsoft

Bill Gates je spoluzakladatelem a předsedou Microsoftu, největší světové softwarové společnosti se 181 tisíc zaměstnanců a ročním obratem 168 miliard dolarů za rok 2021. Více než 90 % osobních počítačů používá operační systém Microsoft Windows a téměř 50 % webových serverů používá software od společnosti Microsoft. (www.devtopics.com 2008)

Jedním z nejvýznamnějších produktů od Microsoftu je balíček Microsoft Office, který obsahuje PowerPoint, Excel, Word a další, dnes již základní aplikace, které značně usnadnili kancelářské práce. (www.devtopics.com 2008)

Microsoft neustále inovuje využívá nejmodernější technologie. Svým přístupem značně ulehčuje přístup k programovacím softwarům. Jedním z dceřiných společností je i Github – open-source portál, kde lidé mohou sdílet své zkušenosti a kód. Vzniká zde komunita programátorů, kteří si navzájem pomáhají a celý proces učení ulehčují a zrychlují. Microsoft také umožňuje studentům zdarma Visual Studio a Visual Studio Code, tedy vývojové prostředí pro tvorbu aplikací. (www.devtopics.com 2008)

5. Programovací jazyky

Programovací jazyk je používaný k vývoji počítačových softwarů – neboli programování, tedy proces algoritmizace určitého úkolu a vytváření postupu pro řešení dané problematiky. (Dělení programovacích jazyků 2022)

Zvláštní skupinou jazyků jsou jazyky XML, HTML a WML, které se oficiálně za programovací jazyky nepovažují, nicméně se řadí do skupiny značkovacích jazyků, jelikož jsou používány výhradně k formátování textů a definování prvků v dokumentech, zatímco programovací jazyky poskytují sadu příkazů a syntaxe pro provádění úkonů počítačovými systémy. (What is the difference between a markup language and a programming language 2022)

5.1. Vyšší a nižší jazyky

Nižší programovací jazyky význačné tím, že jejich instrukce odpovídají instrukcím procesoru, z tohoto důvodu mohou být někdy označeny jako primitivní. To znamená, že procesor bude provádět instrukce napsané programátorem. Jsou závislé na svém procesoru a nelze je přenést na jiný samostatný procesor. (Dělení programovacích jazyků 2022)

V praxi to představuje výhodu v tom, že programátor má přístup k funkcím počítače, ke kterým se v programovacím jazyce vyšší úrovně nedostane. Nicméně se tato skutečnost odráží na složitosti syntaxe kódu, jelikož ta je oproti syntaxi vyšších programovacích jazyků o mnoho složitější. (Dělení programovacích jazyků 2022)

Vyšší programovací jazyky jsou mnohem více exaktní, struktura jejich zdrojů je logická, jsou nezávislé na provozních principech počítače. Překladačem je převede na strojový kód (nebo je provede přímo interpret). Vysokoúrovňový programovací jazyk je ve skutečnosti cokoli, co není assembler, tedy například Basic, či Fortran. případě programovacího jazyka C se jedná o druh konverze mezi vyššími a nižší jazyky, ale je to blíže k vyšším. (Dělení programovacích jazyků 2022)

5.2. Imperativní, logické a funkcionální jazyky

Za imperativní jazyky je považována většina dnes běžně používaných programovacích jazyků. Algoritmus se používá k řešení určité problematiky. Například. C, Basic a PHP. (Dělení programovacích jazyků 2022)

U logických jazyků programátoři pouze popisují problémy pomocí logických příkazů. Poté z něj program získá potřebné informace. Někdy se používají k vytvoření umělé inteligence, pro skutečné programování jsou nepoužitelné. Jedním z těchto jazyků je Prolog. (Dělení programovacích jazyků 2022)

Ve funkcionálním jazyce je vše popsáno pomocí funkcí, často bez proměnných, program je jen komplexní soubor funkcí. Tento způsob programování se blíží klasické matematice. U některých problémů jsou však tyto jazyky vhodnější než imperativ. V této skupině se nachází jazyky jako například Lisp, Haskell, Miranda atd. (Dělení programovacích jazyků 2022)

5.3. Interpretované a kompilované

Interpretované jazyky nebudou přeloženy, dokud nebude program spuštěn. Nemají tak velké formální požadavky (inicializovat proměnnou není nutné, jelikož se její datový typ může v průběhu změnit), za to jsou značně pomalejší. Překládá je tlumočník neboli interpret, který také provádí pokyny během procesu překladu. Hlavní nevýhodou těchto jazyků je, že musí vždy běžet v interpretu. Do této skupiny patří všechny skriptovací jazyky, například Python, nebo PHP a valná většina verzí jazyka Basic. (Dělení programovacích jazyků 2022)

Kompilované jazyky musí být nejdříve plně přeloženy, než je lze spustit. Jsou rychlejší, mají vyšší požadavky na formální správnost kódu. Kompiluje je kompilátor, jehož výsledkem je, alespoň ve většině případů, soubor *.exe*. Tato skupina zahrnuje většinu běžných programovacích jazyků. (Dělení programovacích jazyků 2022)

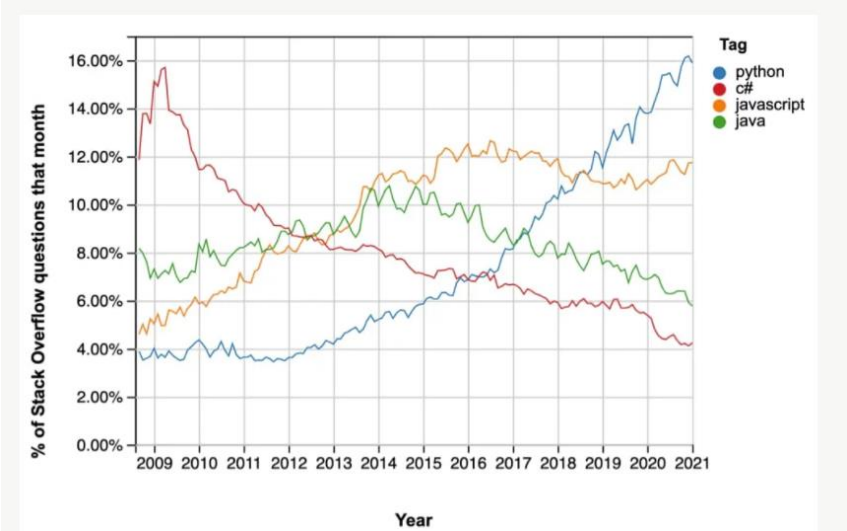
5.4. Další dělení

Dále je možné dělit imperativní jazyky na objektově orientované a strukturované. Objektově orientované jazyky umožňují lepší srozumitelnost a udržitelnost kódu, a proto je dnes velmi populární, do takové míry, že v praxi skoro není možné se mu vyhnout. Spočívá ve vytváření objektů, které reprezentují dané objekty

v reálném světě. Objektům jsou následně přiřazovány vlastnosti a schopnosti, které mohou následně jednotlivé instance objektů sdílet. (Remetei 2022)

5.5. Faktory pro výběr programovacího jazyku

Při výběru vhodného jazyka pro vývoj aplikace je nutné zvážit několik kritérií. Důležitými faktory může být například využitelnost na trhu práce, náročnost jazyka ale i platforma, pro kterou aplikaci vyvíjíme. Jak je možné vidět z tabulky níže, popularita Pythonu začala v posledních letech významně stoupat. (ENGETO 2022)



Obr. 1 - Popularita jednotlivých jazyků v průběhu 12 let

Obrázek z webové stránky: https://engeto.cz/blog/programovani/programuju-2-jaky-programovaci-jazyk-si-vybrat/

5.5.1. Python

Python je jedním z nejlepších backendových jazyků posledních let. Byl vyvinut tak, aby bylo možné psát automatizační skripty, nebo rychlé prototypové aplikace. Má konzistentní syntaxi, jednotnou standartní knihovnu a nejlepší dokumentaci ve své třídě. Umožňuje objektově orientované programování a řešení pro většinu platforem. (ENGETO 2022)

V momentální době Python patří mezi nejrozšířenější víceúčelové programovací jazyky na vysoké úrovni. Umožňuje i objektově orientovaný přístup. Programy navrhované v Pythonu jsou obvykle v podobě menších projektů než například programy navržené v Javě. Programátoři si v tomto jazyku vystačí s poměrně krátkými kódy a jazyk je velmi dobře čitelný. Je využíván technologickými společnostmi jako je Google,

Amazon, Meta, Uber apod. Největší předností jazyka je obrovská sbírka standartních knihoven, které lze při programování využít, například při vývoji uživatelského rozhraní (*Tkinter*), webových frameworků (*Django*, využívaných například pro YouTube, nebo Instagram) atd. (GeeksforGeeks 2022)

Python se používá v rychle rostoucích oblastech, jako je strojové učení, analýza dat, testování, umělá inteligence a hluboké učení (deep learning). V posledních letech zaznamenal výrazný nárůst popularity. Používají jej Kiwi.com, Red Hat, IBM, ROI Hunter, Oracle a mnoho dalších. Psaní kódu v Pythonu je velmi podobné normálnímu psaní. Jedním řádkem kódu lze napsat frázi „Ahoj světe“. Pro srovnání, jak je možné vidět v tabulce níže, pro vykonání stejného úkolu v Javě, je nutné znát 3 komplexní pojmy jako třída, statická metoda a balíček. (ENGETO 2022)

C#

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

JavaScript

```
console.log("Hello World!");
```

Python

```
print("Hello World")
```

Java

```
class HelloWorldApp {
    public static void main(String[]
args) {
        System.out.println("Hello
World!"); // Prints the string to
the console.
    }
}
```

Obr. 2 - Srovnání složitosti jazyků

Obrázek z webové stránky: https://engeto.cz/blog/programovani/programuju-2-jaky-programovaci-jazyk-si-vybrat/

V tomto jazyce byly vyvinuty například aplikace jako Youtube, Instagram, Reddit, Spotify, Dropbox, Quora a Google. (codeinstitute.net 2022)

Nevýhodou Pythonu je rychlost. Přesto, že pro většinu aplikací je Python dostačující, u náročnějších programů, jejichž rychlost se zároveň odvíjí od rychlosti procesoru počítače, na kterém je aplikace spuštěn, může docházet k časovým

prodlevám. U takových programů je lepší využít jazyky jako C#, nebo C++. (Lubanovic, 2019)

5.5.2. JavaScript

JavaScript je nejčastěji používaným skriptovacím jazykem internetového klienta. Je spoušten ve webovém prohlížeči (na rozdíl od PHP, které je spouštěno na serveru). Z bezpečnostních důvodů to může ovlivnit pouze webovou stránku, nikoli počítač uživatele. (14. Programovací jazyky 2022)

JavaScript je základem vývoje uživatelského rozhraní. Často je využívaný k obohacení webových stránek a jejich funkcionalitě. Pomocí JS je možné do webových aplikací přidávat vyskakovací okna, efekty, a dokonce i minihry, umožňuje filtrování produktů, zobrazování recenzí apod. (ENGETO 2022)

S vydáním ECMAScript 6 a frameworků jako Angular, Node, Express a React začali vývojáři používat JavaScript pro programování na straně klienta i serveru.Mezi hlavní výhody JavaScriptu patří přístup k několika frameworkům, funkce ověření dat a jeho kompatibilita s několika programovacími jazyky. (Svoboda 2022)

Bohužel, jazyk není tak stabilní jako Java nebo Python, neustále se vyvíjí, a proto je potřeba ho neustále sledovat kvůli novým trendům a vylepšením. Jazyk má také komplikovanější syntaxi v porovnání s Pythonem. (ENGETO 2022)

V tomto jazyce vznikla například Wikipedia, Ebay, Amazon a Yahoo. (Svoboda 2022)

5.5.3. C#

C# je jazyk vyvinutý společností Microsoft v roce 2000. C# používají známé společnosti v České republice, například Solarwinds, FNZ, Notino, Deloitte, ARTIN, Siemens, Oriflame nebo Komerční Banka. Jazyk je také velmi výkonný, protože dokáže pohánět složité webové nebo desktopové aplikace. (ENGETO 2022)

Nejčastěji se používá v internetových obchodech, bankovních aplikacích, monitorovacích nástrojích atd. Stejně jako Java najde své uplatnění při vývoji aplikací pro Android. C# je ale spíše syntaktický jazyk. (ENGETO 2022)

C# je, stejně jako Java, používán hlavně ve velkých podnicích s robustními systémy a infrastrukturou. Je velmi stabilní, nabízí mnoho pracovních příležitostí a je

velmi pravděpodobné, že v následujících letech bude patřit mezi prvních pět nejpoužívanějších jazyků. (ENGETO 2022)

Pomocí jazyka C# byl vyvinut Bing, Azure portal, Microsoft portal .NET, dále také Adobe produkty, Illustrator atd. (Kumar, 2020)

6. Grafické uživatelské rozhraní

Uživatelské grafické rozhraní (zkráceně GUI – graphical user interface) je systém interaktivních vizuálních komponent pro počítačový software. Uživatelské rozhraní zobrazuje objekty na monitoru, které přenášejí informace a představují akce, které mohou být uživatelem provedeny. GUI bylo vyvinuto společností Xerox PARC v roce 1981, jeho autory jsou Alan Kay, Douglas Engelbart. Později, v lednu 1983, svou verzi grafického rozhraní představila i společnost Apple. (Computer Hope 2021)

Zahrnuje objekty jako kurzor, tlačítka a veškeré ikony, které jsou zobrazovány na monitoru počítače. Tyto objekty umožňují uživateli využívat funkce počítače, aniž by bylo nutné využívat příkazy, jelikož ty jsou předávány právě pomocí objektů. (Computer Hope 2021)

Uživatelské grafické rozhraní významně usnadňuje ovládání počítače, je uživatelsky příjemnější než příkazový řádek, čímž je umožněna práce s počítačem prakticky komukoliv, uživatel nemusí znát žádný programovací jazyk. (Computer Hope 2021)

Proto, aby bylo grafické rozhraní co nejlehčí na ovládání jsou zavedena neformální pravidla o významu objektů. Tlačítka jsou reprezentací pro započetí akce, dialogová okna informují uživatele o průběhu akce, ikony reprezentují programy, funkce, nebo soubory. Menu je seznamem možných příkazů, které jsou dostupné prostřednictvím ikon, okno je klasickou reprezentací pro program, který je aktuálně spuštěný. (Computer Hope 2021)

Mezi výhody uživatelského rozhraní patří samozřejmě jednoduchost. Vizuální podání složitých příkazů, které musely být zadávány do terminálu, užívání počítače velmi usnadňuje. Hledání souboru a programů v počítači je díky GUI také daleko jednodušší, jelikož jsou zobrazeny pomocí ikon a je možné zobrazit i informace o jednotlivých složkách. (Pedamkar 2022)

Za nevýhody, i přes to, že jich není mnoho, lze považovat například určité omezení, protože uživatel může spustit, či započít akce, které jsou předem naprogramované a základní funkce systému se nedají změnit. Některé pokročilé funkce systému nejsou naprogramovány v uživatelském rozhraní a uživatel v případě potřeby

bude nucen využít příkazový řádek, který ale požaduje již znalosti v programování. Grafické rozhraní je také náročnější na spotřebu baterie a místa na grafické kartě. (Pedamkar 2022)

6.1. Komunikace mezi uživatelem a počítačem

Uživatel komunikuje s hardwarem počítače pomocí jednoduchých úkonů, jako například dvojklik, který GUI chápe jako pokyn k porozumění činu uživatele a následného přeložení do jazyka, který je schopen počítač pochopit. Rozhraní také umožňuje získat informace o aktuálním procesu, množství operační paměti, které je momentálně využíváno, velikosti souborů apod. K těmto informacím se uživatel dostane velmi rychle a jednoduše, většinou pomocí kliknutí, nebo dvoj-kliknutí na ikonu. (Pedamkar 2022)

6.2. Tvorba *g*rafického rozhraní v Pythonu pomocí knihovny *Tkinter*

Tkinter je knihovna definována ve standartní knihovně Pythonu, obsahující moduly definující funkce pro vytváření uživatelského rozhraní. Postupem času vznikla řada dalších knihoven a frameworků pro práci s rozhraním, nicméně *Tkinter* je integrální součástí standartní knihovny – tedy není nutné instalovat žádný další program, vše je obsažené již při stažení Pythonu. (Pecinovský 2020)

Základem grafického rozhraní je okno, obvykle je základní okno aplikace nazýváno *root*. Veškerá okna, společně s tlačítky, widgety, štítky a dalšími částmi rozhraní, jsou vytvářena právě díky knihovně *Tkinter*. (Dawson 2010)

6.2.1. Třída Tlačítko

Tlačítka jsou do aplikace přidávány pomocí obecné syntaxe, kdy dojde k nadefinování jména tlačítka a určení jeho datového typu. Následně je nutné nadefinovat také jaké okno je rodičem tlačítka (tedy v jakém okně se má tlačítko zobrazit), jeho velikost, barvu, styl písma, pozici a funkci, kterou bude stisknutí tlačítka spouštět. (GeeksforGeeks 2020)

6.2.2. Třída *Plátno*

Plátno (anglicky Canvas) je používán k rozložení ploch v okně – tedy například k vytvoření grafiky. Opět je nutné přiřazení k rodičovskému oknu, kde se má *canvas*

zobrazit. Dále je možné nastavit další parametry, jako barva pozadí, velikost, formu kurzoru na plátně apod. Konstruktor *plátna* je obdobný jako u tlačítek a ostatních objektů. (GeeksforGeeks 2020)

6.2.3. Zadávání vstupů

K zadání informací do vstupu k následnému zpracování a uložení je využívána datový typ *Entry*, ze kterého je následně možné ukládat zadaný vstup do proměnné a pracovat s ním. Je možné nadefinování parametrů, podobně jako u tlačítek a *plátna*. (GeeksforGeeks 2020)

6.2.4. Třída *Okno*

Okna (anglicky widget) je využíván pro uložení, seskupování a organizaci oken. (GeeksforGeeks 2020)

6.2.5. Třída *Štítek*

Label je užíván pro zobrazování textových polí, které je možné vložit kamkoliv do okna. (GeeksforGeeks 2020)

6.2.6. Zobrazení více oken v aplikaci

Pro zobrazení více oken je využívána funkce *TopLevel*. Tato funkce umožňuje vytvoření více oken v jedné aplikaci. V praxi může fungovat například jako otevření nového okna pro přihlášení do účtu. Opět je nutné určit, jaký objekt je oknu nadřazený, lze upravovat velikost okna, barvu apod. (GeeksforGeeks 2020)

6.2.7. Funkce *mainloop*

Jedná se o metodu používanou ke spuštění aplikace. Metoda běží v nekonečné smyčce, která vyčkává na akci ze strany uživatele, nebo dokud se okno nezavře. (GeeksforGeeks 2020)

7. Gamifikace v aplikacích

Pojem gamifikace se ve svém současném smyslu se začal objevovat až na počátku 21. století. Mezi širší veřejnost se dostal až teprve v posledních letech. Jeho definice jako taková zůstává nejistá a její pojetí se může mnohdy lišit. Obecně gamifikace označuje zavádění herních prvků a mechanik do neherních kontextů, obvykle za účelem ovlivnění chování cílové skupiny a zvýšení jejího zájmu a motivace. (Werbach & Hunter 2012)

V praxi to znamená například zavedení odznaků, bodů, vzájemným sdílením mezi uživateli, nebo dalších praktik. Je nutné vzít v potaz kdo bude aplikaci využívat, jak bude aplikace strukturována a jak je možné gamifikaci zavést, aby užívání aplikace bylo pro uživatele pohodlné a opravdu motivující. (Fiala 2019)

Mezi populární aplikace s prvky gamifikace v momentální době patří Kahoot!, Toglic a Socrative, které se aktivně využívají ve školství a pomáhají žáky vzdělávat zábavnou formou miniher a kvízů. (Fiala 2019)

8. Jednotlivé komponenty aplikace

8.1. Visual Studio Code

Visual Studio Code je výkonný editor zdrojového kódu, funkční jako desktopová aplikace a je dostupný pro Windows, MacOS a Linux. Dodává se s vestavěnou podporou pro JavaScript, TypeScript a Node.js a má bohatý ekosystém rozšíření pro další jazyky (jako C++, C#, Java, Python, PHP, Go) a běhové prostředí (jako .NET a Unity). Microsoft jej nabízí zdarma, je možné jej stáhnout na webových stránkách. (Microsoft 2022.)

VS Code bylo vydáno v roce 2015, od té doby nasbíral až 14 milionů uživatelů a celosvětově asi 24 milionů uživatelů. Mezi jeho přednosti patří především podpora pro GitHub, zvýraznění syntaxe, kontextový našeptávač, podpora pro ladění a refaktorizaci.

8.2. SQLite databáze

Structured Query Language (zkráceně SQL) je standartní databázový jazyk používaný k vytváření, údržbě a načítání relační databáze. (GeeksforGeeks 2021)

Jak již z názvu napovídá, databáze je využívána v případě strukturovaných dat. Relační databáze představuje prostor pro ukládání, i načítání dat ve formě relací (tabulek). V případě nerelačních databází se využívá databáze jménem NoSQL, která je schopná zpracovávat a uchovávat data bez pevné struktury. (GeeksforGeeks 2022)

SQLite databáze nelze do aplikace importovat obdobně jako Tkinter, je nutné stáhnout speciální balíček, který se musí instalovat mimo prostředí Visual Studio Code. Nicméně proces není dlouhý, po instalaci lze databáze jednoduše propojit s kódem pomocí jednoduché syntaxe.

Každý prvek v databázi má vždy přesně stanovený datový typ (číslo, znak, text atd.). Každý sloupec je schopný uchopit pouze jeden datový typ, který se stanoví při vytváření databáze v kódu. (Čápka 2022)

Data lze sice ukládat do textových souborů, nicméně práce s databází je daleko efektivnější a jednodušší, zvláště pokud se jedná o ukládání většího množství dat. Umožňuje vytváření SQL dotazů a spojování několika dotazů, zamezuje nekonzistenci

dat. Databáze představuje 3. vrstvu aplikací – je označována jako datová vrstva. (Čápka 2022)

Databáze je charakterizována těmito vlastnostmi:

- Nedělitelnost – v případě výskytu chyby v průběhu operace se databáze vrátí do původního stavu, například pokud dojde k chybě během bankovního převodu, částka není odeslána a celý proces je nutné zopakovat.
- Validita – Stav databáze musí být vždy konzistentní, musí respektovat všechna definovaná pravidla a omezení.
- Izolace – Operace se navzájem neovlivňují, v případě nutnosti vykonat více operací najednou do stejného řádku, je databáze vyřeší postupně, aniž by jakýmkoliv způsobem mohla narušit validitu výsledku.
- Trvanlivost – Tato vlastnost zajišťuje odolnost vůči ztrátě dat, tedy v případě nového zápisu/přepsání dat, je vše uloženo na pevný disk. Pokud dojde k výpadku sítě, či elektrické energie, databáze by měla být zachována ve stavu těsně před výpadkem.

(Čápka 2022)

8.3. API

Třída API poskytuje přístup ke všem metodám RESTful API na Twitteru a dalších sociálních sítích. Každá metoda může přijímat různé parametry a vracet požadované hodnoty. (Roesslein 2022)

Nástroje API zásadně změnily způsob, jakým vývojáři vytváří dnešní aplikace. Zavedli zcela novou vertikálu softwarových společností „platforma jako služba“. Nástroje založené na API jsou důvodem, proč je možná integrace dat mezi základním obchodním softwarem. API se stala jednou ze základních funkcí mnoha komerčních aplikací. (Park 2022)

API je zkratka pro „Application Programming Interface“, tedy aplikační programovací rozhraní. API je v podstatě sada pravidel, která určují, jak spolu dva stroje mluví. Některé příklady interakcí založených na rozhraní API zahrnují cloudovou aplikaci komunikující se serverem, servery, které si vzájemně pingují, nebo aplikace komunikující s operačním systémem. Kdykoli dojde ke spuštění většiny aplikací

v telefonu, nebo na počítači či k přihlášení na Twitter nebo Facebook, v zákulisí probíhá komunikace mezi několika různými rozhraními API. Téměř všechny podniky, které používají jakýkoli druh moderní technologie, používají na určité úrovni rozhraní API k získávání dat nebo interakci s databází, kterou mohou zákazníci používat. (Park 2022)

Komunikační protokol definovaný rozhraním API umožňuje vývojářům vytvářet, připojovat a integrovat aplikace rychle a ve velkém měřítku. Jako příklad lze uvést slavný mandát Jeffa Bezose z roku 2002. Změna směru Amazonu ukazuje, jak mu API pomohla pohybovat se rychleji než jeho konkurenti, a je údajně důvodem, proč je Amazon tak úspěšný. Bezos nařídil všem svým týmům, aby komunikovaly a vystavovaly data a funkce prostřednictvím rozhraní služeb, tedy rozhraní API. Jakmile byly API a infrastruktura na místě, týmy Amazonu mohly fungovat mnohem efektivněji. Spuštění této nové infrastruktury umožnilo vytvoření webových služeb Amazon, které se od té doby staly největším zdrojem příjmů Amazonu. (Park 2022)

Aplikační programovací rozhraní je sada pravidel, která definují, jak spolu mohou počítače, aplikace nebo stroje komunikovat. Typické uživatelské rozhraní je určeno pro použití člověkem, zatímco API jsou určena pro použití aplikací nebo počítačem. (Park 2022)

Většina webových rozhraní API je umístěna mezi aplikací a webovým serverem. Uživatel zahájí volání API, které aplikaci řekne, aby něco udělala, a poté aplikace pomocí API požádá webový server, aby něco udělal. API je prostředníkem mezi aplikací a webovým serverem a volání API je požadavek. A pokaždé, když má software komunikovat s jiným softwarem nebo online webovými servery, je využívána API k vyžádání informací, které jsou požadovány. (Park 2022)

Je důležité si uvědomit, že ačkoli jsou webová rozhraní API nejběžnější, nejsou omezena pouze na web. Prakticky pro každý stroj nebo systém, který očekává interakci s jinými stroji nebo systémy, existují rozhraní API. (Park 2022)

8.3.1. REST API

REST API je jedním ze čtyř typů API, které se dnes při vývoji aplikací využívá. Více než 70 % všech veřejných API používá REST kvůli jeho rychlému výkonu, spolehlivosti a schopnosti škálovat opětovným použitím modulárních komponent, aniž by to ovlivnilo systém jako celek. (Park 2022)

REST neboli „přenos reprezentativního stavu“ je typ softwarového návrhu, který umožňuje přístup k datům („webovým zdrojům“) pomocí jednotné a předem definované sady operací. Užitná zátěž – data, která mají být doručena – definovaná v samotném požadavku, jsou naformátována v jazyce, jako je HTML, JSON nebo XML. Sada operací jsou metody dostupné pro HTTP, což je základní protokol pro to, jak prohlížeče získávají webové stránky ze serverů. (Park 2022)

8.3.2. *Modul Tweepy*

Tweepy je knihovna Pythonu pro přístup k Twitter API. Je skvělá pro jednoduchou automatizaci a vytváření twitterových botů. *Tweep*y má mnoho funkcí, jako například:

- Získání přístupu k tweetům z časové osy přihlášeného účtu
- Vytváření a mazání tweetů
- Nastavení odběrů od ostatních uživatelů

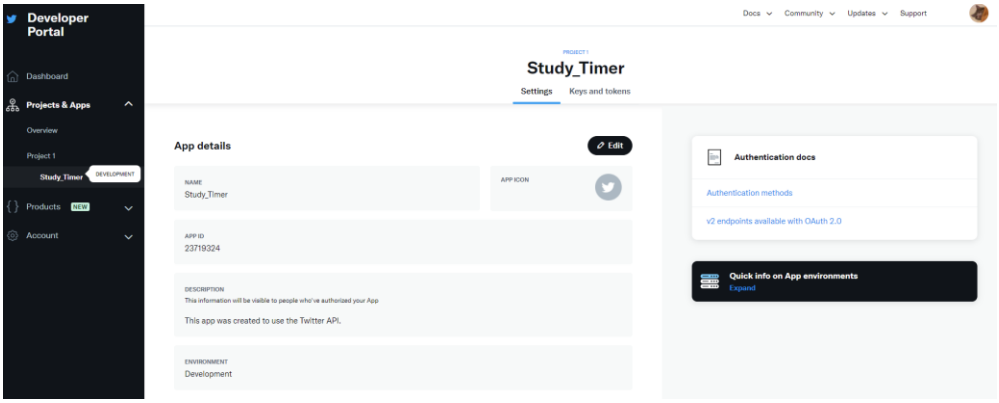
(Rigden, 2018)

Jednou z výhod *Tweepy* je jeho široká dokumentace, tedy jeho uživatel má přístup k veškerým informacím i kódům, Twitter také umožňuje využití podpory v případě jakýchkoliv potíží, nebo jako alternativa může posloužit fórum na oficiálních stránkách, kde si uživatelé navzájem vyměňují své zkušenosti a poznatky.

Další výhodou *Tweepy* je jeho jednoduchost, tedy je možné skrze několik řádků kódu vytvořit efektivní funkce, které aplikaci dokážou oživit. Pro přidání *Tweepy* do aplikace je nutné stáhnout modul v příkazovém řádku pomocí jednoduchého příkazu, poté je nutné naimportovat knihovnu pomocí jednoduché syntaxe přímo ve Visual Studio Code.

Podmínkou využívání modulu *Tweepy* je založení vývojářského účtu na Twitteru, kde je následně nutné sdělit, jak bude modul a data díky němu získána, využíván. Po schválení těchto zadaných údajů, je uživateli přiděleno 5 klíčů, které propojí aplikaci s Twitterem a umožní přidávat příspěvky (neboli tweety), filtrovat je, zobrazovat je v aplikaci a podobně.

Na obrázku níže je uživatelská zeď developerského účtu na Twitteru. Sociální síť poskytuje uživatelům přesný přehled o tom, jaký tarif využívají, volnou kapacitu dat, kterou mohou čerpat a odkazy k dokumentaci, kterou má uživatel celou k dispozici.



Obr. 3 - Developerské prostředí Twitteru

Zdroj: Autor

8.4. Knihovna *Datetime*

V Pythonu nejsou datum a čas vlastním datovým typem, ale lze importovat modul s názvem *Datetime*, aby pracoval s datem i časem. Modul. *Datetime* je zabudován do Pythonu, takže jej není třeba instalovat externě. (GeeksforGeeks 2022)

Modul *Datetime* poskytuje třídy pro práci s datem a časem. Tyto třídy poskytují řadu funkcí pro práci s daty, časy a časovými intervaly. Datum a čas jsou objekty v Pythonu, takže když v případě manipulace dochází k manipulaci s objekty, nikoliv s textem nebo časovými razítky. (GeeksforGeeks 2022)

Modul *Datetime* je rozdělen do 6 hlavních tříd –

- datum – Idealizované naivní datum, za předpokladu, že aktuální gregoriánský kalendář vždy byl a vždy bude platný. Jeho atributy jsou rok, měsíc a den.
- čas – Idealizovaný čas, nezávislý na konkrétním dni, za předpokladu, že každý den má přesně 24*60*60 sekund. Jeho atributy jsou hodina, minuta, sekunda a mikrosekunda.
- datetime – Jedná se o kombinaci data a času spolu s atributy rok, měsíc, den, hodina, minuta, sekunda a mikrosekunda.
- timedelta – Trvání vyjadřující rozdíl mezi dvěma instancemi data, času nebo data a času na mikrosekundové rozlišení.

(GeeksforGeeks 2022)

Modul je v aplikaci využívaný k zápisu záznamů o studiu a získávání dalších údajů.

8.4.1. Třída *date*

Instance třídy date jsou používány k získání datumů. Tyto instance se řídí formou gregoriánského kalendáře. Instance date mají tři celočíselné atributy pouze pro čtení: rok, měsíc a den. Objekty je možné porovnávat mezi sebou, odečítat apod. (Martelli, Ravenscroft, & Holden 2017)

V aplikaci je tato třída využívána při zápisu studijního záznamu.

8.5. Knihovna *Time*

Tento modul umožňuje zpracovávat úlohy zpracovávající, či využívající čas. Tento modul je nutné nejdříve importovat, nicméně není nutné stahovat speciální balíček. Nabízí funkce pro získávání aktuálního času, pro odložení vykonání úloh a dalších složitějších funkcí. V aplikaci je tato knihovna využívána pro odpočet času. (Python time Module 2022)

9. Realizace a zdrojový kód vyvíjené aplikace

9.1. Vytváření tabulek v databázi

9.1.1. Tabulka uživatele

Pro vytvoření tabulky uživatele je nutné importovat knihovnu *sqlite3*, která umožní propojit kód s databází a následně s ní manipulovat. Před jakoukoliv akcí je nutné definovat propojení pomocí následující syntaxe:

```
conn = sqlite3.connect('user_database')
c= conn.cursor()
```

Zdrojový kód 1 - definování propojení s databází

Zdroj: Autor

Pomocí jména knihovny, integrované metody *connect* a jména databáze dochází k propojení s požadovanou databází a následně dochází k vytvoření proměnné, díky které bude možné zadávat SQL dotazy. Vlastnosti tabulky definované v závorkách tvoří jednotlivé sloupce v tabulce, dohromady vytváří takzvané řádky.

Pomocí syntaxe *„IF NOT EXISTS“* dochází ke kontrole při spuštění kódu, tedy program nejdříve zkontroluje, zda se v databází tabulka se stejným názvem již nenachází. Není možné, aby databáze obsahovala více tabulek se stejným názvem, tedy pokud by příkaz takovou podmínku neobsahoval, aplikace by nefungovala správně.

První tabulka je určena pro uchování dat uživatelů, jako je jejich identifikační číslo (ID), které má číselný datový typ a omezení *PRIMARY KE* zaručuje, že se v tabulce nebudou nacházet uživatelé se stejným ID, zároveň dochází k automatické inkrementaci, kdy každý nový uživatel automaticky získá vlastní identifikační číslo. Následně příkaz definuje uživatelské jméno, které je datového typu *text*, tedy umožní uživateli, aby se jeho uživatelské jméno skládalo z písmen i čísel. Pomocí omezení *UNIQUE* dochází vždy v případě nového elementu v databázi ke kontrole, zda se již v databázi stejné uživatelské jméno nenachází, podobně jako v předchozím případě. Na rozdíl od *PRIMARY KEY* ale omezení *UNIQUE* neumožňuje automatickou

inkrementaci, pouze zajišťuje jedinečnost uživatelského jména. Heslo žádná omezení nemá, jelikož není následně v kódu využíváno k získávání uživatelských údajů.

Datový typ celkového součtu času stráveného učením je číslo a pomocí omezení *DEFAULT* bude v tabulce výchozím záznamem vždy 0, kvůli následnému zobrazení statistických dat. Množství zakoupených předmětů v aplikaci je uchováváno v posledním sloupci a je uchovávána jako číselný datový typ.

Pro provedení akce, která je zapsána v příkazu je nutné zavolat metodu *commit*, při které dojde k vytvoření tabulky v databázi. Obecně se metoda *commit* využívána pro vykonání předem stanoveného SQL dotazu.

Pro úsporu operační paměti je nutné také zavolat metodu *close*, která uzavře spojení mezi databází a kódem.

```
def create_database():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS users (ID integer PRIMARY KEY,
                                                   user_name text unique,
                                                   user_password text,
                                                   time integer DEFAULT 0,
                                                   coins integer DEFAULT 0,
                                                   property integer DEFAULT 0 )''')
    conn.commit()
    conn.close()
create_database()
```

Zdrojový kód 2 - Vytváření databáze uživatelů

Zdroj: Autor

9.1.2. Tabulka studijních záznamů

Postup pro vytvoření tabulky k uchování časových záznamů je obdobný předchozímu. Tato tabulka uchovává jméno uživatele, který je přihlášen v době zápisu času, následně sloupec pro délku času a datum provedení zápisu, který je datového typu *typestamp*, program si je tedy vědom, že v tomto sloupci bude zapsáno datum. Pro získání aktuálního datumu je následně využívána knihovna *Datetime*. Pro funkčnost SQL dotazu je nutné opět propojit kód s databází.

```
def create_time_database():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS time ( user_name text, saved_time int NOT NULL, date timestamp )''')
    conn.commit()
    conn.close()
create_time_database()
```

Zdrojový kód 3 - vytváření databáze pro uchovávání času

Zdroj: Autor

9.1.3. Tabulka Inventáře

Jelikož uživatel bude mít jistě v průběhu používání aplikace zakoupeno více předmětů, je nutné vytvořit tabulku, která bude získávat jméno uživatele, který daný předmět zakoupil a jméno předmětu, který byl uživatelem zakoupen. Následně je tato tabulka využívána pro výpis majetku uživatele v hlavním okně aplikace.

Tabulka obsahuje dva sloupce, jméno vlastníka a jméno předmětu, který vlastní. Tato tabulka je využívána v hlavním okně aplikace.

```
def create_inventory_database():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS inventory(owner text, item_name text)''')
    conn.commit()
    conn.close()
create_inventory_database()
```

Zdrojový kód 4 - tabulka inventáře

Zdroj: Autor

9.1.4. Tabulka předmětů

Tato tabulka obsahuje jméno předmětu ve formě textu a jeho cenu, jejíchž datový typ je číslo. Následně je tabulka vyplněna předměty pomocí funkce *insert_item,* jejíchž parametry jsou *name* a *cost*, které jsou následně vloženy do tabulky.

Objekty budou následně zobrazeny v okně *shop*, kde si je bude moci uživatel pomocí mincí zakoupit.

```
def create_item_database():

    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS item_table (item_name text, item_cost integer NOT NULL)''')
    conn.commit()
    conn.close()

create_item_database()

def insert_item(name, cost):
    conn = sqlite3.connect('user_database')
    c= conn.cursor()

    sqlite_insert_query = '''INSERT INTO item_table (item_name, item_cost) VALUES(?,?)'''

    data_tuple = (name, cost,)

    c.execute(sqlite_insert_query, data_tuple)
    conn.commit()

insert_item("soda", 2)
insert_item("peach", 5)
insert_item("phone", 15)
insert_item("rock", 5)
insert_item("imaginary friend", 25)
insert_item("icecubes", 10)
insert_item("apple", 5)
insert_item("dog", 20)
insert_item("cat", 15)
insert_item("hamster", 20)
insert_item("hair tie", 7)
insert_item("gem", 30)
```

Zdrojový kód 5 - tabulka předmětů

Zdroj: Autor

9.2. Tvorba uživatelského rozhraní

Aplikace po spuštění nabízí uživateli možnost registrace do databáze a možnost přihlášení. Okno je voláno pomocí funkce, která využívá modul *Tkinter*, díky kterému mohu nastavit velikost okna, přidat nadpisy a tlačítka, ke kterým následně přiřazuji jednotlivé funkce, ke kterým přistupuji jako k jednotlivým objektům, které jsou již nadefinované v knihovně. Z těchto objektů následně vytvářím instance, které mají jedinečné vlastnost, jako velikost, barvu, typ písma, v případě tlačítek i přiřazené metody, které se jejich stisknutím spouští.

```
def login_screen():
    screen= Tk()
    screen.geometry("300x250")
    screen.title("first you need to log in/register")
    login_label= Label(screen, text = "please login/register", width = "300", height = "2 ", font = ("calibri", 13)).pack()
    login_button=Button(screen, text = "login", command =  login).pack()
    register_button= Button(screen, text = "register", command=register).pack()
    close = Button(screen, text = "close" ,width= 35, borderwidth=5, command=screen.destroy).pack()

    screen.mainloop()
```

Zdrojový kód 6 - Tvorba grafického uživatelského rozhraní

Zdroj: Autor

9.3. Registrace uživatele

Funkce registrace se spustí po stisknutí tlačítka v úvodním okně. Oknu je nataveno jméno, které se následně objeví v horní liště a jeho velikost. Tyto vlastnosti jsou nastaveny pomocí předdefinovaných funkcí knihovny *Tkinter*.

Následně upravuji přístup k proměnným, které v tomto okně využívám, tedy uživatelské jméno a heslo, pomocí klíčového slova *global* bude hodnota této proměnné přístupná i pro ostatní metody. Pomocí funkce *StringVar* určuji datový typ proměnných, které získávám ze vstupu od uživatele. Program s nimi následně bude nakládat jako s textem.

Poslední řádka slouží pro vytvoření tlačítka, které v sobě uchovává metodu pro zapsání zadaných údajů do databáze.

Funkce *pack*, kterou využívám pro jednotlivé objekty, umožňuje jejich zobrazení v okně a je součástí předdefinovaných funkcí *Tkinter*. Jedná se o nejprimitivnější metodu pro zobrazení objektů, jelikož přesně nedefinuji, kde přesně se mají zobrazit.

```
def register():
    global register_screen
    register_screen = Toplevel()
    register_screen.title("register")
    register_screen.geometry("400x500")

    global username
    global password
    global name_entry
    global password_entry
    username=StringVar()
    password=StringVar()

    name_entry = Entry(register_screen, width= 35, borderwidth=5, text = "your username", textvariable=username)
    name_entry.pack()
    password_entry = Entry(register_screen, width= 35, borderwidth=5, text = "your password", textvariable=password)
    password_entry.pack()
    login_button=Button(register_screen, text = "register", command =  register_user).pack()
```

Zdrojový kód 7 - Okno registrace pro získání registračních údajů

Zdroj: Autor

Po stisknutí tlačítka pro registraci se volá metoda *register_user*, která vloží zadané údaje do databáze pomocí modulu *sqlite3*. Uživateli se v okně následně zobrazí zpráva o úspěšném uložení údajů do databáze.

```python
def register_user():
    username_info=username.get()
    password_info=password.get()
    conn = sqlite3.connect('user_database')
    c= conn.cursor()

    c.execute("INSERT INTO users2 (user_name, user_password) VALUES (?,?)" , (username_info, password_info))
    conn.commit()
    conn.close()

    succes= Label(register_screen, text="registration succesfully finished", fg= "green").pack()
```

Zdrojový kód 8¨- Uložení registračních údajů do databáze

Zdroj: Autor

9.4. Přihlášení uživatele

Po registraci se může uživatel opět z úvodního okna přihlásit, a to jednoduchým vyplněním přihlašovacích údajů, které zadá do zobrazovaných polí. Opět je upraven vzhled okna a nastavena přístupnost k proměnným. V tomto okně jsou vytvořena vstupní pole, do kterých uživatel zadá přihlašovací údaje. Následně budou tyto údaje uloženy do proměnných, pomocí klíčového slova *textvariable*, které se následně budou porovnávat s údaji v databázi.

```
def login():
    global login_screen
    login_screen = Toplevel()
    login_screen.title("login")
    login_screen.geometry("400x500")

    global username_verify
    global password_verify
    global login_name_entry
    global login_password_entry
    username_verify=StringVar()
    password_verify=StringVar()

    login_name_entry = Entry(login_screen, width= 35, borderwidth=5, text = "your username", textvariable=username_verify)
    login_name_entry.pack()
    login_password_entry = Entry(login_screen, width= 35, borderwidth=5, text = "your password", textvariable=password_verify)
    login_password_entry.pack()
    login_button=Button(login_screen, text = "login", command =  login_user).pack()
```

Zdrojový kód 9 - Okno pro přihlášení uživatele

Zdroj: Autor

V následující funkcí probíhá porovnávání zadaných údajů uživatelem s údaji uloženými v databázi uživatelů. Pomocí funkce *get* je získáván přístup k datům uložených v proměnných ze vstupu. Poté je nutné získat pomocí SQL dotazu údaje o uživatelích z databáze. Opět je nutné volat funkci *execute,* která daný dotaz vykoná. Pomocí podmínky *if* jsou porovnávány údaje z proměnné *data* se zadanými údaji, pokud se v databázi najde shoda, uživatel je přihlášen do aplikace a okno pro přihlášení se pomocí metody *destroy* zavře. V případě, že podmínka není splněna a v databázi se nenachází žádná data, která by odpovídala datům ze vstupu, uživateli je podána zpráva o neúspěšném přihlášení.

```
def login_user():
    global username_login
    username_login=username_verify.get()
    password_login=password_verify.get()
    global login_screen

    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute("SELECT user_name, user_password FROM users")
    data = c.fetchall()
    for row in data:
        if (username_login in row[0]):
            if (password_login in row[1]):
                start_app()
                login_screen.destroy()
        else:
            fail=Label(login_screen, text="invalid info", fg="red")
            fail.pack()
    conn.close()
```

Zdrojový kód 10 - Přihlášení uživatele

Zdroj: Autor

9.5. Hlavní okno aplikace

Po spuštění hlavního okna se uživateli zobrazí jeho uživatelské jméno a ve vstupech může začít zadávat dobu, kterou chce strávit studiem. Tyto vstupy jsou rozděleny na minuty a sekundy, každá část je zadávána do svého vstupního pole. Odpočet je zahájen stisknutím tlačítka s nápisem *Start the timer*. Je možné odpočet ukončit pomocí tlačítka s nápisem *Stop the Timer*.

V tomto okně se také nacházejí tlačítka pro seznam ostatních uživatelů, které se nacházejí v databázi. Uživatel má přístup k údajům jako celkový čas strávený studiem a počet zakoupených předmětů. Dále se v tomto okně nachází tlačítko pro zveřejňování tweetů. Dále je zde tlačítko pro zobrazení seznamu studijních záznamů a obchodu.

Tlačítka jsou vytvářena pomocí knihovny Tkinter, je nutné stanovit, kterému oknu tlačítko patří, jeho text pro přehlednost a následně je nutné přiřadit ke každému tlačítku funkci, kterou bude spouštět. Tlačítka rozmisťuji do okna pomocí funkce *grid*, kdy určuji přesnou polohu v okně, pomocí klíčových slov *column* a *row*.

```
global minuteString
global secondString
minuteString = StringVar()
secondString = StringVar()

minuteString.set("00")
secondString.set("00")

minuteTextbox = Entry(root, width=3, font=("Calibri", 20, ""), textvariable=minuteString)
secondTextbox = Entry(root, width=3, font=("Calibri", 20, ""), textvariable=secondString)

minuteTextbox.place(x=155, y=100)
secondTextbox.place(x=205, y=100)

button1 = Button(root, text= "Start the timer", command=runTimer)
button1.grid(row=2, column = 1)

button3 = Button(root, text= "Stop the timer", command=stopTime)
button3.grid(row=2, column = 2)

button4 = Button(root, text= "Shop", command=shop)
button4.grid(row=0, column = 9, columnspan= 2, sticky = W+E )
```

Zdrojový kód 11 - Hlavní okno aplikace

Zdroj: Autor

9.6. Zobrazení stavu mincí

Pro zobrazení aktuálního stavu mincí získaných díky studiu je vytvořena metoda jménem *view_coins*. Ta je následně volána v hlavním okně aplikace a dochází k volání i z metody *runTimer* po uplynutí nastaveného času, aby měl uživatel vždy aktuální záznam. Pokud by docházelo k volání metody pouze v hlavním okně aplikace, stav by se neobnovil po vypršení času, i přes to, že by se záznam v databázi již změnil. Pro aktualizovaný záznam by byl uživatel nucen odhlásit se a následně se znovu přihlásit.

Metoda získává pomocí SQL dotazu záznam z databáze, který je podmíněný uživatelským jménem, tedy je získáván jen záznam aktuálně přihlášeného uživatele. Tento údaj je následně ukládán do proměnné jménem *num*, která musí být následně konvertována z datového typu n-tice, do číselné podoby. Jelikož se jedná vždy o pouze jedno číslo, je možné použít ke konvertování předdefinovanou metodu *sum*, která funguje na principu součtu hodnoty všech čísel obsažených v n-tici. Následně je hodnota z proměnné zobrazována v okně aplikace pomocí labelu (štítku).

```
global view_coins
def view_coins():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    current=c.execute("SELECT coins FROM users WHERE user_name=(?)", [user])
    global num
    num=current.fetchone()
    num = sum(num)
    coin_label=Label(root, text='coins: '+ str(num))
    coin_label.grid(row = 9 , column = 1)
```

Zdrojový kód 12 - metoda pro zobrazení aktuálního stavu mincí

Zdroj: Autor

9.7. Inventář v hlavním okně

Funkce je spouštěna v hlavním okně aplikace. Jsou získávány informace z tabulky inventáře. SQL dotaz hledá veškeré záznamy, které patří momentálně přihlášenému uživateli. Veškeré předměty zakoupené přihlášeným uživatelem jsou uloženy do proměnné jménem *user_inventory* pomocí funkce *fetchall*. Následně je pomocí smyčky *for* vytvářen pro každý předmět *štítek* v hlavním okně. Pro správné zobrazení je upraveno zobrazování v okně a každý *štítek* je tedy vytvářen pod **štítek** předchozí.

Pomocí předdefinované metody *len* je získáván počet předmětů, které uživatel vlastní. Pokud je počet roven nule, je v okně aplikace zobrazen *štítek*, který o takovém stavu uživatele informuje. Podmínka je zajištěna pomocí syntaxe *if*.

Propojení s databází je ukončeno syntaxí *conn.close*.

```
global show_inventory
def show_inventory():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    c.execute("SELECT item_name FROM inventory WHERE owner=?", (user,))
    conn.commit()
    global user_inventory
    user_inventory=c.fetchall()
    print("----------------")
    print(user_inventory)
    sum_inventory=Label(root, text= "Your inventory")
    sum_inventory.grid(row=9, column=6)

    j=0
    x = 13
    object_label = []
    for object in user_inventory:
        object_label.append(Label(root, text = object))
        object_label[j].grid(row=x, column = 6)
        j+=1
        x+=1

    lenght_user_inventory = len(user_inventory)
    if lenght_user_inventory == 0:
        inventory_label=Label(root, text="your inventory is empty")
        inventory_label.grid(row = 11 , column = 6)
    else:
        pass
```

Zdrojový kód 13 - metoda show_inventory

Zdroj: Autor

9.8. Okno *friendslist*

Okno nabízí seznam ostatních uživatelů, jejich celkového času stráveným učením a jejich stavu sloupce *coins*. Tyto údaje jsou zobrazovány uživatelovi pomocí metody *Treeview,* která je předdefinována v knihovně *Tkinter*. Umožňuje přehledné zobrazení dat v libovolném počtu sloupců. Jeho nastavení je rychlé a snadné. Následně jsou data do *Treeview,* nahrávány pomocí *for* smyčky, která postupně prochází databázi uloženou do proměnné *records*. Jednotlivé prvky databáze následně vkládá do okna pomocí předdefinované metody *insert.*

```
friends_tree= ttk.Treeview(friends, selectmode="extended")
friends_tree['columns'] = ("Username", "Time", "Coins")
friends_tree.column("#0", width=120, minwidth=25)
friends_tree.column("Username", anchor=W, width=120)
friends_tree.column("Time", anchor=CENTER, width=120)
friends_tree.column("Coins", anchor=W, width=120)

friends_tree.heading("#0", text = "Label", anchor=W)
friends_tree.heading("Username", text = "Username", anchor=W)
friends_tree.heading("Time", text = "Time", anchor=CENTER)
friends_tree.heading("Coins", text = "Coins", anchor=W)

c.execute("SELECT * FROM users1")
records = c.fetchall()
global count
conn.commit()

count=0
for record in records:
        friends_tree.insert(parent='', index='end', iid = count, text = '', values=(record[0], record[2], record[3]))
        count+=1

friends_tree.pack()
conn.close()
```

Zdrojový kód 14 - Zobrazení údajů ostatních uživatelů v databázi

Zdroj: Autor

9.9. Okno *records*

Okno *records* funguje obdobně jako okno *friendslist*, které zobrazuje údaje o uživatelích, tedy k zobrazení časových údajů o studiu je užíván *Treeview* a smyčka *for*. V okně je zobrazen datum provedení zápisů do databáze i s počtem minut strávených učením. Toto okno umožní uživateli získat přesný přehled o množství stráveného času studiem, zároveň zápis času studenta motivuje, zabraňuje prokrastinaci díky stanovení jasného cíle.

Okno *records* zároveň zobrazuje další dva statistické údaje – průměrná délka jednotlivých studijních úseků a celkový čas studia za aktuální den. Tyto data umožňují uživateli přesný přehled o jeho délce studia a student díky nim ví, zda je jeho čas věnovaný učivu adekvátní.

```
def records():
    conn = sqlite3.connect('user_database')
    c= conn.cursor()

    records = Toplevel()
    records.title("records")
    records.geometry("500x500")

    records_tree= ttk.Treeview(records, selectmode="extended")
    records_tree['columns'] = ("Username", "Time", "Date")
    records_tree.column("#0", width=0, minwidth=0)
    records_tree.column("Username", anchor=W, width=120)
    records_tree.column("Time", anchor=CENTER, width=120)
    records_tree.column("Date", anchor=W, width=120)

    records_tree.heading("#0", text = "Label", anchor=W)
    records_tree.heading("Username", text = "Username", anchor=W)
    records_tree.heading("Time", text = "Time", anchor=CENTER)
    records_tree.heading("Date", text = "Date", anchor=W)

    global username_verify
    user= username_verify.get()

    data = c.execute("SELECT * FROM time WHERE user_name=?", (user,))
    fetched_records= data.fetchall()
    global count
    conn.commit()

    count=0
    for record in fetched_records:
            records_tree.insert(parent='', index='end', iid = count, text = '', values=(record[0], record[1], record[2]))
            count+=1
```

Zdrojový kód 15 - okno records

Zdroj: Autor

9.9.1. Průměrná délka

Tento údaj je získáván pomocí údajům z databáze. Je nutné z tabulky uživatelů získat jeho celkovou dobu studia, tento údaj je následně uložený do proměnné *sum_time* pomocí předdefinované metody *fetchone*. Údaj je opět datového typu n-tice, tedy je nutné ho, obdobně jako u předchozích případů, převézt na číselný datový typ.

Druhý údaj, který je využívaný k získání průměrné délky studia je počet záznamů o studiu, které má pod svým uživatelským jménem student uložen v tabulce *time_table*. Počet záznamů je získán pomocí metody jménem *len*, která vrací počet údajů nacházejících se v n-tici. Údaj je ukládán do proměnné *len_all_data*.

Proměnnou *sum_time* je nutné vydělit beze zbytku proměnnou *len_all_data*. Výsledek je poté zobrazen uživateli v okně *records*, pomocí štítku, kombinací textu a proměnné *average*.

```
c.execute("SELECT time FROM users WHERE user_name=?", (user,))
conn.commit()
sum_time = c.fetchone()
sum_time = sum(sum_time)
c.execute("SELECT saved_time FROM time WHERE user_name=?", (user,))
count_time = c.fetchall()
len_all_data = int(len(count_time))
average= sum_time//len_all_data
average_sum = Label(records, text='your average time is: '+str(average))
average_sum.pack()
```

Zdrojový kód 16 - Průměrná délka studijního času

Zdroj: Autor

9.9.2. Denní součet času

Do proměnné *records_date* je uložen aktuální datum pomocí funkce *today*, která je získávána z knihovny *Datetime*. Dále je nutné získat z tabulky *time_table* časové záznamy momentálně přihlášeného uživatele z aktuálního dne. Lze využít metodu *sum,* která sečte veškeré časové záznamy nacházející se v proměnné *day_sum*, ale je nutné převést každý element v listu, jelikož je možné, že uživatel bude mít za den více než jeden záznam. V takovém případě databáze vrací list n-tic, které se musejí převádět jednotlivě a poté uložit do proměnné. Tato proměnná následně obsahuje součet délek všech studijních záznamů z daného dne a je, stejnou formou jako v předchozím případě, zobrazována uživateli v okně *records* pomocí štítku.

```
records_date = date.today()
print("curdate?", records_date )
c.execute("SELECT saved_time FROM time_try2 WHERE user_name=? AND date=?", (user, records_date))
day_sum = c.fetchall()
y=0
sum_char = 0
print("daysum:",day_sum)
for char in day_sum:
    char = sum(char)
    sum_char += char

daily_sum = Label(records, text='your daily summary of time is: '+str(sum_char) + " minutes")
daily_sum.pack()
conn.close()
```

Zdrojový kód 17- Zobrazení celkového času stráveného studiem za daný den

Zdroj: Autor

9.10. Okno obchodu

9.10.1. Vypsání předmětů z databáze

Okno obchodu je spouštěno funkcí jménem *shop*, která se volá stisknutím tlačítka v hlavním okně. Opět je nutné propojit tabulku v databázi s kódem. V tomto případě se přistupuje k údaji o množství mincí, kterým momentální uživatel disponuje.

V následujícím kroku je přistupováno pomocí SQL dotazu ke všem záznamům v tabulce předmětů k zakoupení. Tyto předměty jsou následně zobrazovány v okně obchodu, pomocí smyčky *for*, ve formě tlačítek, kterým je volána funkce jménem *buy*.

Tlačítka získávají z databáze jména předmětů a jejich cenu. Tyto údaje jsou předány pomocí funkce *lambda,* která umožní přenést jednotlivé atributy tlačítka při jeho stisknutí.

Uživatel stisknutím tlačítka potvrzuje, že si chce daný předmět zakoupit a spouští se funkce, která kontroluje, zda má na tento předmět dost mincí.

```
conn = sqlite3.connect('user_database')
c= conn.cursor()
c.execute("SELECT * FROM item_table")
conn.commit()
global items
items = c.fetchall()
item_button=[]

item_button = []
i=0
for item in items:
    item_button.append(Button(shop, text = item[0], textvariable=item[1], command = lambda name = item[0],
     cost =  item[1]:buy(name, cost), compound=RIGHT))
    item_button[i].grid(row = i, column=5, columnspan= 2, sticky = W+E )
    i+=1
```

Zdrojový kód 18 - zobrazení tlačítek z databáze

Zdroj: Autor

9.10.2. Získávání údajů z databáze

Z předešlého kroku je z tlačítka přeneseno jméno předmětu a jeho cena do parametrů funkce *buy*. Z databáze je pomocí dotazu získáno množství mincí, které má uživatel dostupné, tento údaj je uložen do proměnné *coinstatus* a opět převeden na číselný datový typ. Poté je z databáze získán počet předmětů, které již uživatel vlastní, a to do proměnné *inventorystatus*.

```
def buy(name, cost):
    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    global username_verify
    user= username_verify.get()
    c.execute("SELECT coins FROM users WHERE user_name=?", (user,))
    conn.commit()
    coinstatus = c.fetchone()
    coinstatus = sum(coinstatus)
    c.execute("SELECT property FROM users WHERE user_name=?", (user,))
    conn.commit()
    inventorystatus = c.fetchone()
    inventorystatus = sum(inventorystatus)
```

Zdrojový kód 19 - získání údajů pro zakoupení předmětu

Zdroj: Autor

9.10.3. Zakoupení předmětu

```
if coinstatus >= cost:
        coinstatus-=cost
        print("coinvar: ", coinstatus)
        c.execute("UPDATE users SET coins=? WHERE user_name=?" , (coinstatus, user,))
        conn.commit()
        inventorystatus+=1
        c.execute("UPDATE users SET property=? WHERE user_name=?" , (inventorystatus, user,))
        conn.commit()
        messagebox.showinfo('item purchased',"item succesfully purchased")
        c.execute("INSERT INTO inventory (owner, item_name) VALUES (?,?)" , (user, name ,))
        conn.commit()
else:
    messagebox.showinfo('not enough coins',"you dont have enough coins to purchase this object.")
show_inventory()
```

Zdrojový kód 20 - porovnávání údajů z databáze a proměnných

Zdroj: Autor

Po zakoupení předmětu je nutné porovnat, zda má uživatel dostatek mincí na zakoupení požadovaného předmětu. Pomocí podmínky *if* je kontrolováno, zda je cena předmětu menší nebo rovna počtu mincí uživatele. Pokud ano, je cena odečtena od

konta uživatele a tabulka uživatelů je aktualizována, konkrétně nový stav sloupce *coins* a *property*.

Do tabulky *inventory* je přidán záznam o zakoupení ve formě jména uživatele a jména produktu. Uživatel je informován o úspěšném zakoupení pomocí funkce *showinfo*, která zobrazí nové okno s danou informací.

Pokud je cena větší, než počet mincí, je nákup zamítnut a uživatel je o takovém výsledku informován.

Pro obnovení inventáře v hlavním okně aplikace je následně volána funkce *show_inventory*.

9.11. Odpočet času

Funkce je spouštěna po zadání času na časomíře v hlavním okně aplikace. Poté musí uživatel stisknout tlačítko pro spuštění času. Tlačítko spouští funkci. Po spuštění se do proměnné *holdvalue* uloží hodnota zadaná uživatelem, která je dále využívána k ukládání času do databáze. Globální proměnná *clockTime* se skládá ze vstupu od uživatele a umožňuje jeho zpracování na časovou hodnotu.

Pomocí smyčky *while* probíhá kontrola hodnoty na časomíře. Proměnné jsou následně upravovány tak, aby odpovídali času, tedy pomocí funkce *divmod* se časomíra dělí šedesáti.

Následně je výpis na časomíře upraven do časového formátu pomocí funkce *format.* Okno aplikace je obnoveno. Funkce *sleep* aplikaci určuje, jaký časový interval je mezi provedením následujícího úkonu, tedy díky ní je možné odpočítávat jednotlivé sekundy při spuštění funkce.

Tato funkce využívá knihovnu *time.*

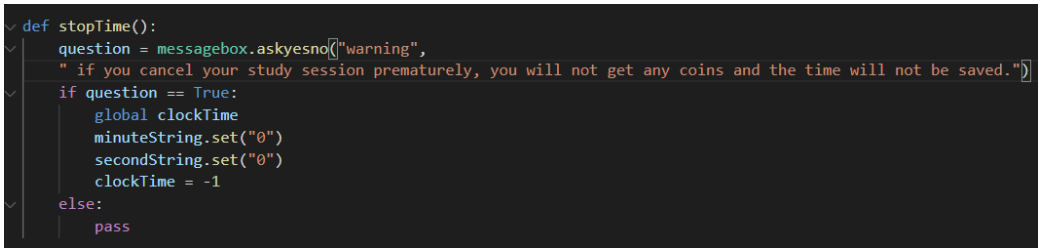
```
def runTimer():

    global holdvalue
    holdvalue= int(minuteString.get())
    global clockTime
    global running
    running = True
    try:
        clockTime = int(minuteString.get())*60 + int(secondString.get())
    except:
        messagebox.showinfo('wrong input',"this format is not supported.")

    while(clockTime > -1):
        global totalMinutes
        global totalSeconds
        totalMinutes, totalSeconds = divmod(clockTime, 60)

        minuteString.set("{0:2d}".format(totalMinutes))
        secondString.set("{0:2d}".format(totalSeconds))

        root.update()
        time.sleep(1)
```

Zdrojový kód 21 - Odpočet času

Zdroj: Autor

9.12. Ukončení odpočtu času

Po stisknutí tlačítka pro ukončení odpočtu času je uživatel upozorněn na možnou penalizaci v případě, že ukončí odpočet času předčasně, aby byl uživatel motivovaný opravdu své předem nastavené cíle splnit. Pokud opravdu ukončí odpočet, časomíra se nastaví na nulové hodnoty a odpočet se ukončí.



```
def stopTime():
    question = messagebox.askyesno("warning",
    " if you cancel your study session prematurely, you will not get any coins and the time will not be saved.")
    if question == True:
        global clockTime
        minuteString.set("0")
        secondString.set("0")
        clockTime = -1
    else:
        pass
```

Zdrojový kód 22 - předčasné ukončení odpočtu času uživatelem

Zdroj: Autor

9.13. Funkce *insert_coins*

Tato metoda je volána uvnitř metody pro odpočet času. Spouští se až v momentě, kdy čas vyprší, tato podmínka je stanovena pomocí podmínky *if*. Následně je nutné stanovit propojení s požadovanou databází pojmenovanou jako *user_database*, která obsahuje tabulky s uloženými daty uživatele.

Pomocí SQL dotazů je z dané tabulky vybrán údaj o množstvím peněz, které má momentálně daný uživatel k dispozici, tato podmínka je určena pomocí klíčového slova *WHERE*.

Daná hodnota je uložena do proměnné jménem *var_acc*, která je získána pomocí vestavené funkce *fetchone*, která umožní přístup k údaji. Nicméně tento údaj je z databáze získaný jako n-tice, což je datový typ, se kterým nelze dále manipulovat (přičíst hodnotu z proměnné a podobně). Je tedy nutné n-tici konvertovat na *integer* (datový typ pro číselné hodnoty). Tento problém je opatřený funkcí *sum*, která funguje na principu sečtení veškerých hodnot v n-tici, nicméně v tomto konkrétním případě se v ní bude nacházet vždy jen jedna hodnota, proto je možné tuto funkci použít, aniž by došlo k neúmyslnému přepsání uchovávané hodnoty.

Hodnota z časomíry je následně vydělena pěti, jelikož uživatel dostává za každých pět minut studia 1 minci. Poté co je uložena hodnota z databáze do proměnné v číselné podobě je možné k ní přičíst hodnotu, která vychází z monitoringu času. Následně aktualizovanou hodnotu je nutné uložit zpět do databáze pomocí dalšího SQL dotazu.

```
def insert_coins():
    if (clockTime == 0):
        global username_verify
        user= username_verify.get()

        conn = sqlite3.connect('user_database')
        c= conn.cursor()
        global holdvalue
        devidedValue= holdvalue//5
        var = c.execute("SELECT coins FROM users WHERE user_name=(?)", (user,))
        conn.commit()
        var_acc=var.fetchone()
        int = sum(var_acc)
        int+=devidedValue
        c.execute("UPDATE users SET coins=? WHERE user_name=?" , (int, user,))
        conn.commit()
```

Zdrojový kód 23 - převod minut na mince

Zdroj: Autor

9.14. Funkce *insert_time*

Funkce využívá knihovnu *Datetime*, díky které je možné získat aktuální datum pouze pomocí volání předdefinované funkce *today*. Získaná hodnota je následně ukládána do proměnné jménem *curdate*.

Pomocí SQL dotazu se získá hodnota z tabulky uživatelů, konkrétně hodnota celkového času studia, která je následně opět konvertována z n-tice na číselnou hodnotu, ke které je následně možné přičíst hodnoty ze vstupu. Tato hodnota je uložena do proměnné *time_fetch*.

Podobně jako v předchozích případech, je nutné převést číslo z databáze do takové podoby, kdy je s ním možné manipulovat pomocí funkce *sum*. Hodnota z databáze je následně uložena do proměnné *dat_time* a je k ní přičtena hodnota z časomíry uložená v proměnné *holdvalue*.

V předposledním kroku dochází k přidání záznamu do tabulky *time_table*, která zaznamenává aktuální datum z proměnné *curdate*, jméno uživatele a záznam času z proměnné *holdvalue*.

Poté je uložen záznam i do tabulky *users*, do této tabulky je uložen čas z databáze, ke kterému je přičten poslední záznam z odpočtu v proměnné *dat_time*.

```
def insert_time():
    if(clockTime == 0):
        conn = sqlite3.connect('user_database')
        c= conn.cursor()

        global curdate
        curdate = date.today()

        global username_verify
        user= username_verify.get()

        global holdvalue
        time_var = c.execute("SELECT time FROM users WHERE user_name=?", [user])
        time_fetch= time_var.fetchone()
        dat_time = sum(time_fetch)
        dat_time+= holdvalue

        c.execute("INSERT INTO time (user_name, saved_time, date) VALUES (?,?,?)" , (user, holdvalue, curdate,))
        conn.commit()
        c.execute("UPDATE users SET time=? WHERE user_name=?" , (dat_time, user,))
        conn.commit()
```

Zdrojový kód 24 - Funkce pro zapsání času do databáze

Zdroj: Autor

9.15. Zveřejnění tweetu

```
def share():
    API_KEY= "XSDxsOgD3L0S5brkBA9rwKHm2"
    API_KEY_SECRET = "qqCjv3DcITUrnmkxZjb22UoAGLResGSLF0xrwpUZo3tsXQnAMy"

    ACCESS_TOKEN="1364558905729306629-McUBDQ08ri9ghVH17HP90fAubyqtp9"
    ACCESS_TOKEN_SECRET = "m0FfX4Hz8bPoyCs90AqoxrdRYeNiERRQsR0XqYcDNp6VQ"

    auth = tweepy.OAuthHandler(API_KEY, API_KEY_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

    api = tweepy.API(auth)

    conn = sqlite3.connect('user_database')
    c= conn.cursor()
    global username_verify
    user= username_verify.get()

    last_record = c.execute("SELECT saved_time FROM time WHERE user_name=?",(user,))
    tweet_time = last_record.fetchall()
    lastTweet = tweet_time[-1]
    tweet_time = sum(lastTweet)
    status=('sharing with #studyTimer, today I studied for %s'%(lastTweet)+' minutes')

    try:
        api.verify_credentials()
        api.update_status(status)
        messagebox.showinfo('success',"tweet was successfully posted on Twitter.")
    except:
        messagebox.showinfo('eror',"tweet could not be posted, try again later please")
```

Zdrojový kód 25 - Zveřejnění příspěvku na Twitteru

Zdroj: Autor

Zveřejnění (neboli post) tweetu je prováděn pomocí API a modulu jménem *Tweepy*. Ten umožňuje i jiné funkce, než jen tzv. postování, například je možné skrze něj filtrovat tweety podle klíčového slova. Tyto metody ale mohou vyžadovat jiné klíče, které Twitter přiřazuje po vytvoření vývojářského účtu. V případě zveřejňování tweetů je vyžadován API klíč, API tajný klíč, přístupový token a tajný přístupový token. Tyto jednotlivé klíče jsou ukládány do proměnných a následně předány při autorizaci přístupu. Pomocí syntaxe *try/except*, která se využívá při opatřování výjimek je kontrolováno, zda je API řádně konfigurována, a to pomocí předdefinované metody *verify_credentials* v knihovně *Tweepy*. Z databáze studijních záznamů je získán poslední údaj, který je uložen do proměnné *last_time*. Následně je volána metoda pro přidání tweetu na zeď účtu pomocí syntaxe *update_status*, opět se jedná o předdefinovanou metodu knihovny. V případě chyby program není schopný přistoupit k API a uživateli je zobrazena zpráva o chybě v novém okně pomocí modulu *messagebox*.

Metoda *update_status* získává opět záznam z tabulky *time_table*, konkrétně čas aktuálně přihlášeného uživatele, který je uložen do proměnné. Tato proměnná je následně vložena do tweetu:

```
status=('sharing with #studyTimer, today I studied for %s'%(tweet_time))
```

Zdrojový kód 26 - syntaxe pro zveřejnění tweetu

Zdroj: Autor

9.16. Získávání tweetů s klíčovým slovem

Pro zobrazení tweetů s hashtagem aplikace (#studyTimer), se kterým jsou postované tweety z funkce *share,* je využíván modul Tweepy. Pro propojení se sociální sítí je nutné opět propojit aplikaci pomocí API, k tomuto slouží funkce *OAuthHandler* společně s funkcí *set_access_token.*

V proměnné *keyword* je následně uloženo klíčové slovo. Tweety, které klíčové slovo obsahují jsou následně získávány a uloženy do proměnné *fetch_tweets.* Je také možné určit si limit tweetů, které budou následně zobrazeny, maximální počet je tedy uložen do *limit.*

Pomocí metody *search_tweets* od knihovny Tweepy jsou následně získávány tweety splňující zadané požadavky. V okně aplikace jsou zobrazovány pomocí smyčky *for* a pro každý tweet bude vytvořen štítek.

```
def get_tweets():

    tweets = Toplevel()
    tweets.geometry("900x500")
    tweets.title("studyTimer tweets")

    API_KEY= "XSDxsOgD3L0S5brkBA9rwKHm2"
    API_KEY_SECRET = "qqCjv3DcITUrnmkxZjb22UoAGLResGSLF0xrwpUZo3tsXQnAMy"
    ACCESS_TOKEN="1364558905729306629-McUBDQ08ri9ghVH17HP90fAubyqtp9"
    ACCESS_TOKEN_SECRET = "m0FfX4Hz8bPoyCs90AqoxrdRYeNiERRQsR0XqYcDNp6VQ"

    auth = tweepy.OAuthHandler(API_KEY, API_KEY_SECRET)
    auth.set_access_token(ACCESS_TOKEN,ACCESS_TOKEN_SECRET)
    api = tweepy.API(auth)

    keyword= "#studyTimer"
    limit = 10

    fetch_tweets = api.search_tweets(keyword, count= limit)
    print(fetch_tweets)

    for tweet in fetch_tweets:
        print(tweet.text)

    k=0
    row=0
    tweet_label=[]
    for tweet in fetch_tweets:
        tweet_label.append(Label(tweets, text = tweet.text, textvariable = tweet.text))
        tweet_label[k].grid(row=row, column=2)
        k+=1
        row+=1
```

Zdrojový kód 27 - Získávání Tweetů pomocí modulu Tweepy

Zdroj: Autor

9.17. Odhlášení uživatele

Funkce se spouští po stisknutí tlačítka v okně aplikace. Po stisknutí se uživateli zobrazí okno díky knihovně *messagebox,* které se ho ptá, zda si opravdu přeje se odhlásit a vyčkává na jeho reakci. Pokud uživatel stiskne možnost „ANO“, aplikace se ukončí a časomíra se nastaví na hodnotu 0, v případě, že funkce *runTimer* je právě spuštěna, ukončí se společně s aplikací.

Pokud stiskne tlačítko „NE“, aplikace zůstane spuštěna.

```
def log_out():
    areyousure = messagebox.askyesno("Are you sure? ", " Do you really want to quit? ")
    if areyousure == True:
        global clockTime
        minuteString.set("0")
        secondString.set("0")
        clockTime = -1
        root.destroy()
    else:
        pass
```

Zdrojový kód 28 - odhlášení uživatele

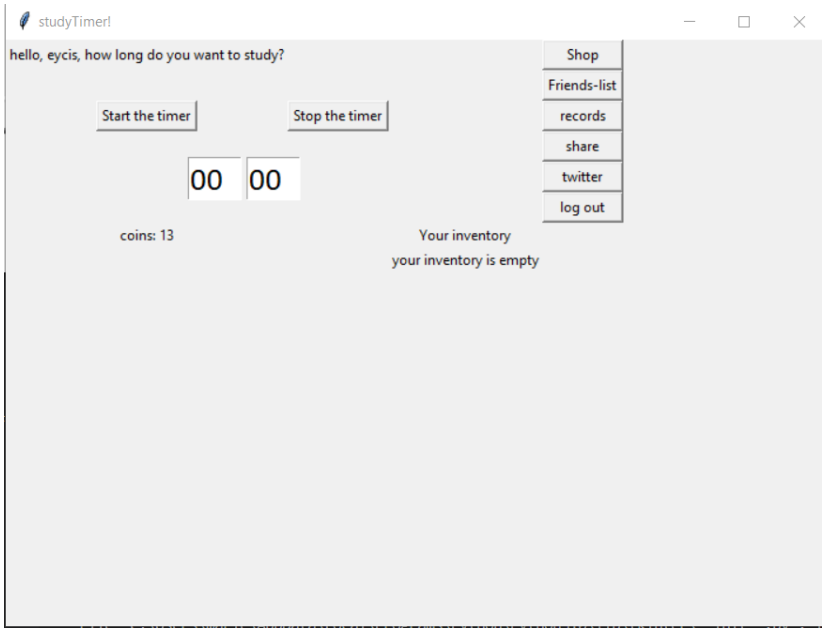
Zdroj: Autor

10. Aplikace

Aplikace následně zobrazuje v grafickém rozhraní veškeré funkce popsané na předchozích stránkách. Na levé straně lze nalézt tlačítka která, po rozkliknutí, zobrazí jednotlivá okna s dalšími funkcemi a možnostmi. Zároveň hlavní okno aplikace zobrazuje aktuální stav mincí a inventář uživatele s jeho zakoupenými předměty.

Hlavní součástí okna je časomíra, zde si uživatel nastavuje čas a může odpočet začít a ukončit. Čas je po spuštění odpočítáván v hlavním okně.

Pod časomírou je vypsán aktuální stav mincí, který se aktualizuje v případě spuštění aplikace, i v případě ukončení odpočtu.



Obr. 4 - Hlavní okno aplikace s uživatelským rozhraním

Zdroj: Autor

10.1. Testování

Testování probíhalo ve dvou etapách, tedy při vývoji aplikace a poté po dokončení základních funkcí. Následně byla aplikace otestována třemi dobrovolníky, kteří se podíleli na fázi testování a bylo možné díky tomu opravit případné nedostatky v kódu. Jednalo se osoby od věku 18 až 22 let. Účastníci měli za úkol stanovit si dobu vyhrazenou pro studium a následně čas využít. Dobrovolníci konstatují, že stanovení cíle a jeho zapsání mělo za následek vyšší motivaci a odhodlání využít čas na

maximum. Zároveň oceňují, že záznamy jsou uchovávány a následně využity pro získání statistických údajů.

Aplikace byla hodnocena převážně kladně, hlavně díky možnosti interakce se sociální sítí Twitter.

Fáze testování jsou popsány v tabulce testování (viz. příloha A).

11. Závěr

Základní stanovené cíle bakalářské práce byly splněny. Byla vytvořena aplikace pro monitoring času, který student věnoval učení. Zároveň je aplikace propojena se sociální sítí Twitter. Využití API při tvorbě aplikací je aktuálně velmi populární a dá se očekávat daleko jednodušší přístup ke knihovnám umožňující propojení, stejně jako širší využití při neziskovém vývoji aplikací. Tento fakt může vést i k vyšší poptávce po širší propojenosti aplikací.

Aplikace má zavedené základní prvky gamifikace, které mohou uživatele motivovat k častějšímu studiu. Všechny záznamy o učení jsou uchovávány v databázi a k těmto údajům má uživatel přístup přímo v aplikaci.

Na závěr je nutné zmínit, že je aplikace tvořena základními funkcemi, tedy existuje zde potenciál pro následný vývoj. Nové funkce by mohly zahrnovat například širší využití propojení s Twitterem a databázi umístěnou na serveru pomocí SQL Server, která by umožnila uživatelům širší propojením mezi sebou. Dále by aplikace mohla obsahovat seznam aktivit, které chce uživatel během odpočítávání času vykonat. Tyto aktivity by po ukončení mohl postupně odškrtávat a tyto záznamy by mohly vést k dalším statistickým údajům o produktivitě uživatele zobrazené v aplikaci.

Abstrakt

Theoretical part of this work talks about desktop applications, their historical development, contribution to society and how it effects our day-to-day life. It also introduces the most important names of software development, how they get into it and their contribution to it. The work also presents the most used programming languages of today and an pplications without which we can not imagine modern life. The goal of this thesis is to develop a desktop application, that works as a counter for students, checking how much time they spent on learning. The application is connected with social media, so the student are able to see the time their friends spent with studying, which should motivate each student to study even more. This application uses Python and its libraries as a programming language.

Keywords: python and its libraries, software development, social media, history, desktop application

ZDROJE

KNIŽNÍ ZDROJE

Dawson, M. D. (2010). *Python Programming for the absolute Beginner* (3rd ed.). Boston, USA: Course Technology.

Evans, C. L. E. (2018). Broad Band: The Untold Story of the Women Who Made the Interne. London, UK: Portfolio.

Hally, M. H. (2005). *Electronic Brains: Stories from the Dawn of the Computer Age*. Washington D.C., USA: Joseph Henry Press.

Hammerman, R. H., & Russel, A. L. R. (2015). *Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age*. San Rafael, USA: Morgan & Claypool Publishers.

Lubanovic, B. L. (2019). *Introducing Python: Modern Computing in Simple Packages* (2nd ed.). Sebastopol, USA: O'Reilly Media.

Martelli, A. M., Ravenscroft, A. R., & Holden, S. H. (2017). *Python in a Nutshell* (3rd ed.). Sebastopol, USA: O'Reilly Media, Inc.

Werbach, K. W., & Hunter, D. H. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business*. Philadelpia, USA: Wharton Digital Press.

Pecinovský, R. (2020). *Začínáme programovat v jazyku Python*. Praha, Česká republika: Grada.

OSTATNÍ ZDROJE

Codeinstitute.net. (2022). *7 Popular Software Programs Written in Python.* Dostupné z: https://codeinstitute.net/global/blog/7-popular-software-programs-written-in-python/

Ivt.mzf.cz. (2022). *14. Programovací jazyky*. Dostupné z: http://www.ivt.mzf.cz/seminar/14-programovaci-jazyky/

Cs.wikipedia.org. (2022). *Aplikační software.* Dostupné z: https://cs.wikipedia.org/wiki/Aplika%C4%8Dn%C3%AD_software

Čápka, D. Č. (2022). *Lekce 1 - MySQL krok za krokem: Úvod do MySQL a příprava prostředí.* Dostupné z: https://www.itnetwork.cz/mysql/mysql-tutorial-uvod-a-priprava-prostredi

Computer Hope. (2021). *GUI.* Dostupné z: https://www.computerhope.com/jargon/g/gui.htm

Prg.estranky.cz. (2022). *Dělení programovacích jazyků.* Dostupné z: https://prg.estranky.cz/

Engeto.cz. (2022). *Programuju, 2. díl: Jaký programovací jazyk si vybrat?* Dostupné z: https://engeto.cz/blog/programovani/programuju-2-jaky-programovaci-jazyk-si-vybrat/

Fiala, J. F. (2019). *Gamifikace ve výuce.* Dostupné z: https://spomocnik.rvp.cz/clanek/21961/GAMIFIKACE-VE-VYUCE.html

Terrahunt.cz. *(2022) Gamifikace a její využití ve vzdělávání.* Dostupné z: https://www.terrahunt.cz/blog/gamifikace-a-jeji-vyuziti-ve-vzdelavani

GeeksforGeeks.org. (2020). *Python GUI – tkinter*. Dostupné z: https://www.geeksforgeeks.org/python-gui-tkinter/

GeeksforGeeks.org. (2021). *Structured Query Language (SQL)*. Dostupné z: https://www.geeksforgeeks.org/structured-query-language/?ref=gcse

GeeksforGeeks.org. (2022). *Python datetime module*. Dostupné z: https://www.geeksforgeeks.org/python-datetime-module/

GeeksforGeeks.org. (2022). *Python Programming Language*. Dostupné z: https://www.geeksforgeeks.org/python-programming-language/

GeeksforGeeks.org. (2022). *SQL Tutorial.* Dostupné z: https://www.geeksforgeeks.org/sql-tutorial/?ref=gcse#basics

Harper, D. H., & Stockman, L. M. S. ,(2022). *The History of FORTRAN*. Dostupné z: https://www.obliquity.org/computer/fortran/history.html

Vidabytes.com. (2022). *History of UNIX How has it affected other systems?* Dostupné z: https://vidabytes.com/cs/historia-de-unix/

Koďousková, B. K. (2021). *Co je webová a desktopová aplikace a jaký je mezi nimi rozdíl?* Dostupné z: https://www.rascasone.com/cs/blog/desktop-web-aplikace

Kumar, A. K. (2020). *https://qr.ae/pvK66P [Quora Post]*. Dostupné z: https://www.quora.com/What-large-applications-have-been-built-using-C

Code.visualstudio.com. (2022). *Visual Studio Code.* Dostupné z: https://code.visualstudio.com/docs

Park, A. P. (2022). *How do APIs work?* Dostupné z: https://tray.io/blog/how-do-apis-work

Pedamkar, P. P. (2022). *What is GUI.* Dostupné z: https://www.educba.com/what-is-gui/

Javascript.jecool.net. *(2022). Programovací jazyky.* Dostupné z: http://www.javascript.jecool.net/teorie.html

Programiz.com. (2022). *Python time Module*. Dostupné z: https://www.programiz.com/python-programming/time

Remetei, D. R. (2022). *Co je to OOP a proč se to mám učit?* Dostupné z: https://engeto.cz/blog/programovani/co-je-to-oop/

Rigden, J. R. (2018). *Tweepy: a Python Library for the Twitter API*. Dostupné z: https://medium.com/@jasonrigden/tweept-a-python-library-for-the-twitter-api-9d0537dcebd4

Roesslein, J. R. (2022). *TWITTER API V1.1 REFERENCE*. Dostupné z: https://docs.tweepy.org/en/stable/getting_started.html#

Svoboda, R. S. (2022). *Nejoblíbenější programovací jazyky současnosti a jejich budoucí trendy.* Dostupné z: https://club.coolpeople.cz/nejoblibenejsi-programovaci-jazyky-soucasnosti-a-jejich-budouci-trendy/1372.html

Devtopics.com. (2008). *Top 10 Software Innovators of All Time*. Dostupné z: https://www.devtopics.com/top-10-software-innovators-of-all-time/

Tse, D. T. (2020). *How Claude Shannon Invented the Future*. Dostupné z: https://www.quantamagazine.org/how-claude-shannons-information-theory-invented-the-future-20201222/

History-computer.com. (2022). *VisiCalc of Dan Bricklin and Bob Frankston Guide: History, Origin, and More*. Dostupné z: https://history-computer.com/visicalc-of-dan-bricklin-and-bob-frankston-guide/

Cs.strephonsays.com. (2022). *What is the difference between a markup language and a programming language*. Dostupné z: https://cs.strephonsays.com/what-is-the-difference-between-markup-language-and-programming-language

Yost, M. Y. (2018). *A Brief History of Software Development*. Dostupné z: https://medium.com/@micahyost/a-brief-history-of-software-development-f67a6e6ddae0

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

ZDROJOVÝ KÓD

PŘÍLOHY

A. Testovací scénáře

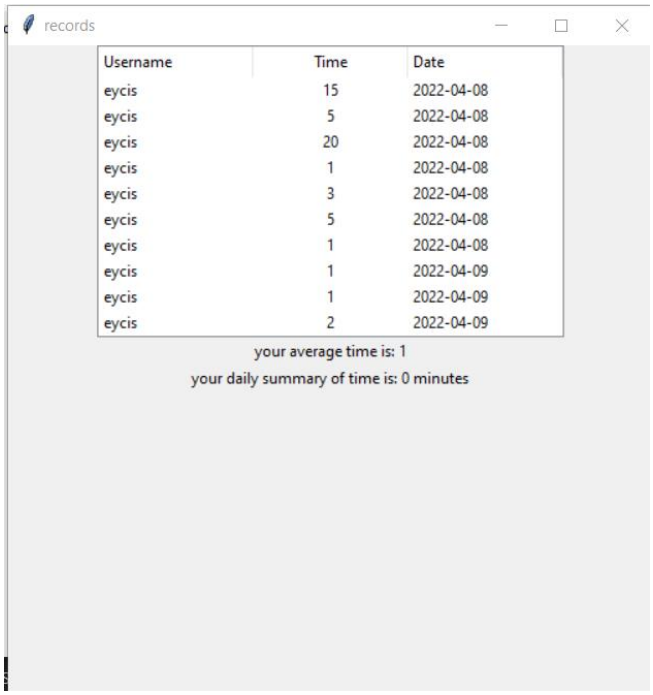
Akce	**Předpoklady**	**Instrukce**	**Očekáváný výsledek**
Registrace	Unikátní uživatelské jméno a heslo	Zadání unikátního uživatelského jména a hesla do určených polí a potvrzení volby tlačítkem *register*.	Uživateli se vytvoří účet a jím zadaná data jsou uložena do tabulky v databázi. Uživatel je o úspěšném vytvoření informován v okně aplikace.
Přihlášení	Uživatel je úspěšně registrován, databáze je propojená s kódem pro porovnání zadaných údajů.	Vyplnění polí pro uživatelské jméno a heslo a potvrzení volby tlačítkem *login*	Uživatel je úspěšně přihlášen a hlavní okno aplikace je zpřístupněno uživateli.
Spuštění odpočtu času	Uživatel je přihlášeny a připraven studovat.	Zadání plánované doby studia uživatelem a stisknutí tlačítka pro spuštění odpočtu	Uživatelem zadaná doba se začne odpočítávat.
Předčasné ukončení odpočítávání	Uživatel je přihlášený a čas se odpočítává.	Stisknutí tlačítka pro ukončení odpočtu a potvrzení volby.	Časomíra je vynulována a uživatel je o stavu a penalizaci informován.
Plánované ukončení odpočtu	Uživatel je přihlášený a čas se odpočítává	-	Po uplynutí zadaného času je odpočet ukončen, stopky vynulované a uživatel je informován o ukončení odpočtu.

Uchování záznamů o studiu v tabulce uživatelů	Uživatel je přihlášen a dokončil minimálně jednu studijní periodu	-	Po ukončení odpočtu jsou záznamy o délce studia připsány k celkovému součtu doby.
Připsání mincí na konto uživatele	Uživatel je přihlášen a dokončil minimálně jednu studijní periodu, která byla delší, než 5 minut	-	Uživateli se po vypršení zadaného času přepočítají minuty na mince a jeho stav mincí v tabulce uživatelů je aktualizován. Aktuální stav je zobrazován v okně aplikace.
Zakoupení předmětů	Uživatel má dostatek mincí na požadovaný předmět	Stisknutím tlačítka požadovaného předmětu v okně obchodu uživatel vybere požadovaný předmět.	Předmět je zakoupen, množství zakoupených předmětů v tabulce uživatelů je aktualizováno, stejně jako tabulka inventář. Následně je předmět zobrazen v inventáři v hlavním okně aplikace.
Zobrazení veškerých studijních záznamů a statistických údajů	Uživatel má více než jeden studijní záznam.	Stisknutím příslušného tlačítka v hlavním okně aplikace je otevřeno okno se záznamy	Uživateli je zobrazen seznam s jeho studijními záznamy. Také může vidět průměrnou délku studia a jeho denní součet doby strávenou studiem.

Zobrazení údajů o ostatních uživatelích.	V databázi je uložených více uživatelů.	Uživatel otevře příslušné okno *friendslist.*	V okně je zobrazen seznam všech uživatelů, jejich celkový čas studia a jeho počet mincí.
Sdílení studijních záznamů na Twitteru.	Uživatel má založený účet na Twitteru.	Uživatel stiskne tlačítko *share,* nacházející se v hlavním okně aplikace.	Uživatel sdílí dobu jeho posledního studia zapsaného v databázi na svém účtu na Twitteru.
Zobrazení tweetů v aplikaci	Uživatel je přihlášen.	Uživatel stiskne tlačítko *twitter,* nacházející se v hlavním okně aplikace.	Uživateli je zobrazeno okno, ve kterém jsou zobrazeny tweety s nastaveným hashtagem aplikace.
Odhlášení uživatele	Uživatel je přihlášen	Uživatel stiskne tlačítko *log out* nacházející se v hlavním okně aplikace a potvrdí svou volbu ve vyskakovacím okně.	Uživatel je odhlášen z aplikace a hlavní okno aplikace se zavře.

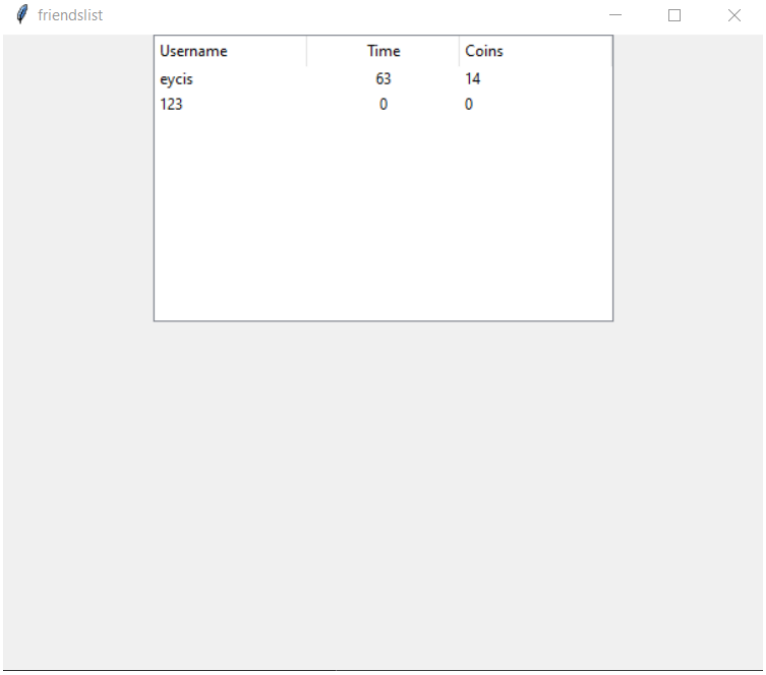
Zdroj: Autor

B. Zobrazení okna *records*



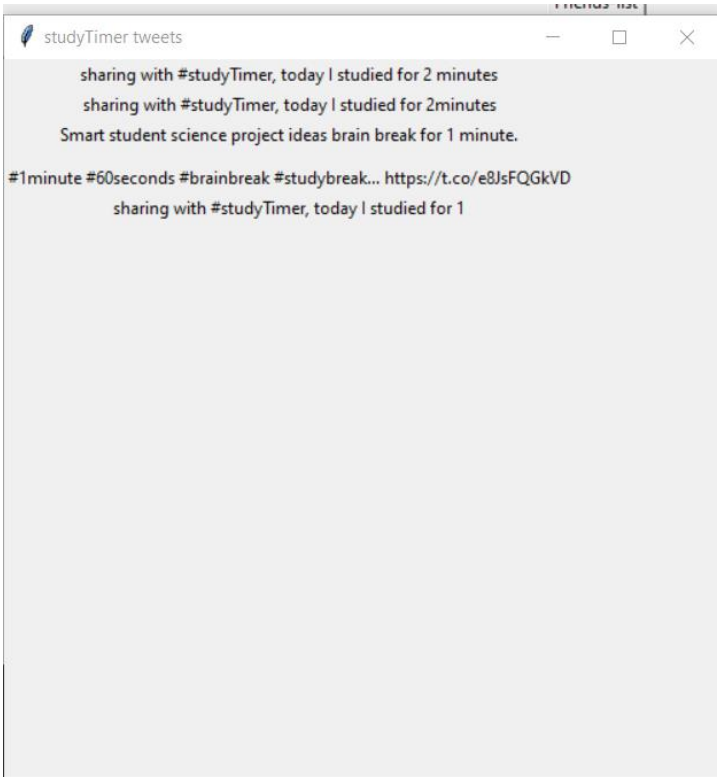
Zdroj: Autor

C. Zobrazení okna *friendslist*



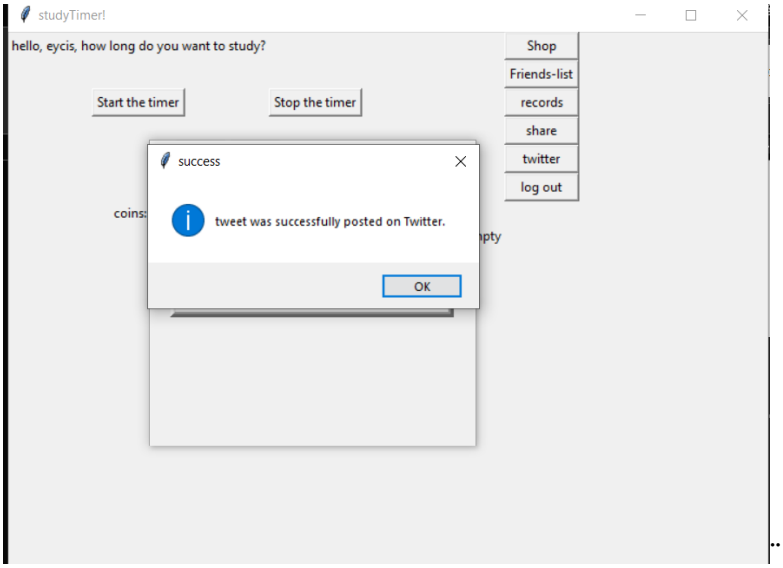
Zdroj: Autor

D. Zobrazení okna *twitter*

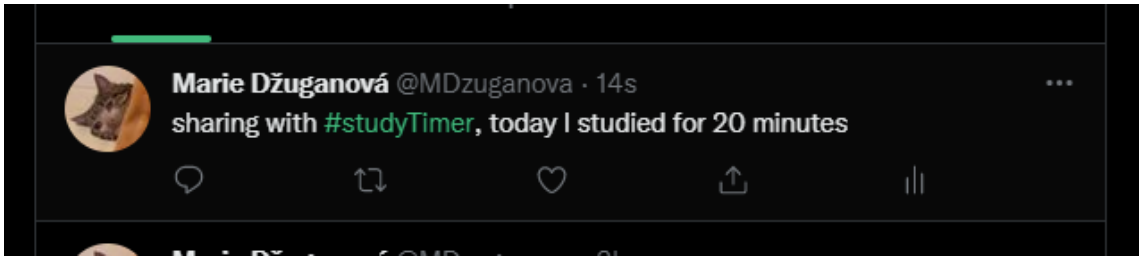


Zdroj: Autor

E. Zveřejňování tweetů na Twitteru

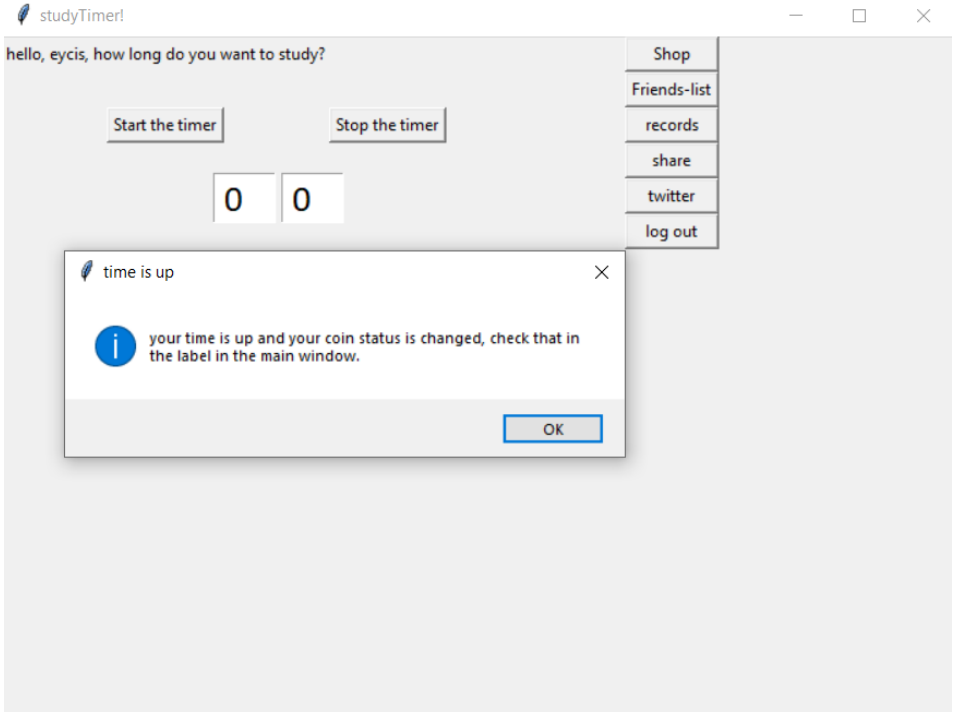


Zdroj: Autor



Zdroj: Autor

F. Vypršení času



Zdroj: Autor

G. Kód

Kód je dostupný na následujícím odkazu:

https://github.com/eycis/Marie_Dzuganova.git