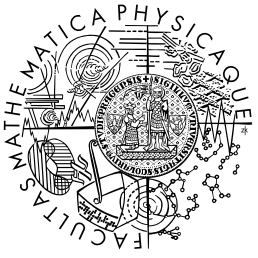


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Pavol Rajczy

Aplikace pro vytváření osobního rozvrhu

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Klára Pešková, Ph.D.

Studijní program: Informatika (B0613A140006)

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád by som sa poďakoval svojej vedúcej práce, Mgr. Kláre Peškovej, Ph.D., za čas, ochotu a trpezlivosť, bez ktorej by táto práca nemohla vzniknúť. Taktiež i rodine a priateľom, že so mnou vydržali i keď som sa v posledných mesiacoch premenil na hobita.

Název práce: Aplikace pro vytváření osobního rozvrhu

Autor: Pavol Rajczy

Department: Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Klára Pešková, Ph.D., Katedra softwaru a výuky informatiky

Abstrakt: Táto práca predstavuje Windows aplikáciu vyvinutú na zvýšenie zamerania a efektívnosti užívateľov prostredníctvom efektívnych stratégií riadenia času a rozvrhu. Softvér integruje zavedené metódy ako Eat That Frog!, Eisenhower Matrix, 80-20 Rule a upravenú metódu Eat That Frog! spolu s Pomodoro Technique na pomoc používateľom pri vytváraní a dodržiavaní optimalizovaných rozvrhov. Po úspešnom zvládnutí aktivity v rozvrhu má užívateľ možnosť zdieľať svoj výsledok s inými užívateľmi, čo taktiež zvyšuje motiváciu. Štúdia podčiarkuje úlohu aplikácie pri podpore zlepšených pracovných návykov a koncentrácie. Proces vývoja, od návrhu až po implementáciu, uprednostňuje praktické funkcie pri tvorbe rozvrhu.

Klíčová slova: Pomodoro, Eat That Frog!, Eisenhower Matrix, 80-20 Rule, časový manažment, stavanie rozvrhu

Title: Application for Time Management

Author: Pavol Rajczy

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Klára Pešková, Ph.D., Department of Software and Computer Science Education

Abstract: This work presents a Windows application developed to increase user focus and efficiency through effective time and schedule management strategies. The software integrates established methods such as Eat That Frog!, the Eisenhower Matrix, the 80-20 Rule, and the modified Eat That Frog! along with the Pomodoro Technique to help users create and stick to optimized schedules. After successfully mastering the activity in the schedule, the user has the opportunity to share his result with other users, which also increases motivation. The study underscores the app's role in promoting improved work habits and concentration. The development process, from design to implementation, prioritizes practical functions in the creation of a schedule.

Keywords: Pomodoro, Eat That Frog!, Eisenhower Matrix, 80-20 Rule, time managment, construction of timetable

Obsah

Úvod

V dnešnej dobe rýchleho tempa a neustáleho technologického pokroku sa efektívne riadenie času stáva kľúčom k úspechu v škole, ale aj v osobnom živote. S nárastom študijných požiadaviek sa študenti čoraz viac obracajú na digitálne nástroje a aplikácie, ktoré by im mohli pomôcť zlepšiť ich produktivitu a efektívnosť. Pre študentov, ktorí prevažne používajú počítač pri výuke avšak aplikácie na manažment času neponúkajú metódy na tvorbu rozvrhu alebo sú príliš komplikované na použitie bez hlbšieho porozumenia základným princípom riadenia času a taktiež majú mnohé z nich väčšinu funkcií zamknutých za platobnou bránou.

Práve táto skutočnosť ma motivovala k vývoju tohto projektu, kde sa zameriame na vývoj aplikácie pre operačný systém Windows, ktorá zvyšuje sústredenie a efektivitu užívateľov prostredníctvom implementácie efektívnych stratégií riadenia času a rozvrhu. Taktiež naším cieľom bude integrovať metódy riadenia času, ako sú Eat That Frog! (Tracy [1]), Eisenhower Matrix (Clear [2]), 80-20 Rule (Koch [3]), zároveň pridať upravenú metódu Eat That Frog! a techniku Pomodoro (Cirillo [4]). Tieto nástroje pomôžu užívateľom vytvárať rozvrhy. Ďalším cieľom je poskytnúť nástroje na zdieľanie úspechov a výsledkov medzi užívateľmi, čo prispieva k vzájomnej motivácii a posilňovaniu pracovných návykov (Galinsky a Schweitzer [5]). Taktiež sme chceli otestovať na užívateľoch či aplikácia funguje a porovnať ich výsledky pri používaní aplikácií a pri ich samostatnej práci. Tým aj poukázať na jej prínos pri zlepšení pracovných návykov a koncentrácie užívateľov.

Práca podčiarkuje dôležitosť užívateľsky orientovaných riešení, ktoré umožňujú užívateľovi plynulo pridať do rozvrhu všetky aktivity, ktoré sa môžu vyskytnúť počas týždňa. Či už dlhodobé alebo denné ciele, čo môže významne prispieť k zvýšeniu produktivity a spokojnosti užívateľov.

Pri vypracovaní práce sme vychádzali hlavne zo zahraničnej literatúry. Dôležitá bola hlavne pre správne pochopenie metód na časový manažment a aby sme vedeli správne čitateľa oboznámiť s úlohou, ktorú hrajú pri správnom fungovaní aplikácie.

Prvá kapitola práce predstavuje a popisuje metódy riadenia času, ktoré sú v práci používané. Druhá sa priamo pozerá na užívateľskú stranu programu a jeho všetky funkcionality. V nasledujúcej kapitole sa pozrieme na implementáciu metód a tvorbu rozvrhu. Posledná kapitola hovorí o testovaní, nazbieraných dátach, ktoré dopodrobna rozoberie taktiež sa pozerá na výsledky testovania a na splnené ciele. Na záver sa spätne pozrieme na prácu, zhodnotíme výsledky a navrhneme budúce kroky projektu.

1 Iné aplikácie na časový manažment

V tejto kapitole sa pozrieme na aplikácie, ktoré sú funkcionalitou podobné nášmu projektu. Najprv sa pozrieme na Clockify a Asana, keďže patria v súčasnosti medzi najpopulárnejšie a mám s nimi osobne skúsenosť. Potom sa pozrieme na aplikáciu Strava, keďže nápad na tento projekt začal vznikať pri jej používaní.

1.1 Clockify a Asana

Medzi najznámejšie aplikácie na časový manažment patria napríklad Clockify a Asana. Obom avšak chýba, pre mňa najdôležitejšia funkcia, možnosť generovať rozvrh cez špecifický algoritmus. Clockify síce poskytuje možnosť vytvárania špecifického rozvrhu ale má 2 nevýhody; rozvrh musí byť vytvorený človekom a na odomknutie tejto funkcie je treba platiť za software. Jediná z ponúkaných metód je Pomodoro, ale tú poskytuje väčšina aplikácií, keďže nie je zložitá.

Pri pohľade na iné funkcionality môžeme pozorovať prvky, ktoré som použil aj v mojom projekte. Možnosť zdieľať svoje projekty a to na čom užívateľ v súčasnosti pracuje. Taktiež tieto aplikácie ponúkajú i možnosť výkazov o výkone užívateľa, čo moja aplikácia ponúkať neplánuje.

1.2 Strava

V tejto sekcii sa pozrieme na aplikáciu Strava. Ide síce o aplikáciu zameranú na fyzickú aktivitu, dala mi však nápad na možnosť zdieľania výsledkov medzi užívateľmi. Pri jej používaní som si všimol, že som motivovanejší keď sa "pretekám" s kamarátmi, ktorí aplikáciu tiež používajú. To ma inšpirovalo pozrieť sa na to, čo nás motivuje a po prečítaní Friend & Foe [5] som sa rozhodol túto funkciu zaviesť i do môjho projektu.

2 Použité metódy

Pred tým ako sa zameriame na samotnú aplikáciu si predstavíme metódy časového manažmentu, ktoré boli použité v práci, aby sme pochopili ich aplikovanie v projekte.

2.1 Pomodoro

Technika Pomodoro je metóda riadenia času, ktorú vyvinul Francesco Cirillo koncom osemdesiatych rokov minulého storočia [4]. Technika využíva časovač na rozdelenie práce do intervalov, tradične v dĺžke 25 minút, oddelených krátkymi prestávkami. Tieto intervaly sa označujú ako „pomodoros“, talianske slovo pre paradajky, inšpirované kuchynským časovačom v tvare paradajok, ktorý Cirillo používal ako študent univerzity.

Ako to funguje

1. **Vyberte si úlohu:** Musíte mať jasnú predstavu o tom, na akej úlohe alebo sérii úloh budete pracovať.
2. **Nastavenie časovača Pomodoro:** Časovač je nastavený na 25 minút a na úlohe sa musí pracovať výlučne počas tohto času.
3. **Práca na úlohe:** Venujte pozornosť úlohe bez akýchkoľvek prerušení. Bez kontroly e-mailov, bez používania telefónu, bez rozptyľovania.
4. **Ukončite prácu, keď časovač zazvoní:** Keď časovač zhasne, začiarknite políčko na kus papiera, čím označíte dokončenie jednej relácie.
5. **Urobte si krátku prestávku:** Po zazvonení časovača si urobte krátku prestávku (zvyčajne 5 minút), aby ste si oddýchli a zresetovali svoj duševný stav.
6. **Každé štyri pomodorá si dajte dlhšiu prestávku:** Keď dokončíte štyri pomodorá, urobte si dlhšiu prestávku (15 až 30 minút). Pomáha to dobiť energiu a udržať sústredenie na dlhšie obdobia.

Dĺžka sústredenia, krátkej a dlhej prestávky sa môže meniť v závislosti na používateľovi.

Predpokladané výhody

- **Zlepšené sústredenie a koncentrácia:** Vďaka krátkym, sústredeným intervalom práce sa minimalizuje rozptýlenie a maximalizuje sa koncentrácia.
- **Jasnejšie nastavenie cieľa:** Táto technika vás núti pracovať s časom, a nie proti nemu, a stanovuje jasné fázy pre prácu a odpočinok.
- **Znížená šanca vyhorenia:** Pravidelné prestávky zaisťujú, že myseľ zostáva svieža a stres je minimalizovaný.

- **Efektívne riadenie času:** Môže pomôcť efektívnejšie riadiť čas rozdelením práce na zvládnuteľné intervaly.

2.2 80/20 rule

Pravidlo 80/20, známe aj ako Paretov princíp, je koncept, ktorý vyvinul taliansky ekonóm Vilfredo Pareto koncom 19. storočia [3]. Pareto si všimol, že 80% talianskeho bohatstva vlastní 20% populácie, čo je fenomén, o ktorom neskôr zistil, že je prítomný v rôznych iných krajinách a kontextoch. Princíp bol odvtedy prijatý v kontexte podnikania, ekonomiky a produktivity, čo naznačuje, že 80% účinkov pochádza z 20% príčin.

Ako to funguje

1. **Identifikujte kľúčové úlohy:** Na začiatok je dôležité uviesť zoznam všetkých úloh a činností, ktoré sa týkajú vášho života, práce alebo konkrétneho projektu.

2. **Analyzujte príspevky:** Posúďte, ktoré úlohy najviac prispievajú k požadovaným výsledkom. To zahŕňa čo najpresnejšie meranie účinnosti každej úlohy alebo činnosti z hľadiska jeho výsledkov.

3. **Zamerajte sa na činnosti s vysokým dopadom:** Uprednostnite a zamerajte viac zdrojov na 20% činností, ktoré preukázateľne prinášajú 80% výsledkov. To môže znamenať vyčlenenie väčšieho množstva času, energie alebo kapitálu na tieto kľúčové aktivity.

4. **Znížte dôraz alebo odstráňte aktivity s nízkym dopadom:** Identifikujte 80% aktivít, ktoré prispievajú len k 20% výsledkov. Zvážte zníženie zdrojov určených na tieto účely alebo ich úplné odstránenie, aby ste zefektívnili úsilie a zvýšili efektivitu.

5. **Priebežné hodnotenie:** Pravidelne kontrolujte a upravujte vstupy a výstupy, aby sa zabezpečilo efektívne uplatňovanie zásady, keďže situácie a výsledky sa môžu časom vyvíjať.

Predpokladané výhody

- **Zvýšená efektivita:** Identifikovaním a zameraním sa na činnosti, ktoré prinášajú najvýznamnejšie výsledky, môžete dosiahnuť viac s menším úsilím.

- **Vylepšené rozhodovanie:** Pochopenie, ktoré oblasti uprednostňovať, pomáha pri prijímaní informovaných rozhodnutí, ktoré maximalizujú vplyv.

- **Optimalizácia zdrojov:** Pridelenie zdrojov (čas, peniaze, pracovná sila) najvplyvnejším činnostiam zaisťuje, že sa využívajú najefektívnejšie.

- **Vylepšená produktivita:** Zefektívnenie zamerania na to, čo generuje najlepšie výsledky, môže viesť k vyššej produktivite.

2.3 Eat That Frog!

Eat That Frog! je populárna metóda riadenia času a produktivity, ktorú predstavil Brian Tracy vo svojej knihe s rovnakým názvom [1]. Táto fráza je inšpirovaná citátom, ktorý sa často pripisuje Markovi Twainovi, ktorý naznačuje, že ak prvá vec, ktorú každé ráno urobíte, je zjesť živú žabu, môžete celý deň prežiť s uspokojením, keď viete, že toto je pravdepodobne tá najhoršia vec, ktorá sa stane celý deň.

Ako to funguje

1. **Identifikujte svoju žabu:** Tento krok zahŕňa určenie najnáročnejšej a najdôležitejšej úlohy, ktorú máte na svojom zozname úloh – tej, s ktorou budete s najväčšou pravdepodobnosťou otáľať. Vaša „žaba“ je úloha, ktorá bude mať najväčší vplyv na dosiahnutie vašich cieľov a často tá, ktorá môže spôsobiť úzkosť, kým sa nesplní.

2. **Zjedzte žabu ako prvú vec ráno:** Zvládnite túto úlohu ako prvú vec ráno, keď je vaša energia a sústredenie na vrchole. Tým sa zabezpečí, že najväčšiu a najdôležitejšiu úlohu dokončíte čo najskôr.

3. **Pripravte si noc vopred:** Naplánujte si úlohy na predchádzajúci večer, aby ste presne vedeli, čo je vaša žaba na ďalší deň. Tento prípravok pomáha začať deň s jasným zameraním.

4. **Aplikujte pravidlo 80/20:** Paretov princíp alebo pravidlo 80/20 naznačuje, že 20 % vašich aktivít bude predstavovať 80 % vašich výsledkov. Identifikujte tieto náročné úlohy a zamerajte sa na ne ako na svoje žaby.

Predpokladané výhody

- **Zvyšuje produktivitu:** Tým, že najskôr dokončíte najdôležitejšiu úlohu, zvýšite svoju produktivitu hneď od začiatku dňa.

- **Zlepšuje prioritizáciu:** Táto metóda vás núti uprednostňovať úlohy na základe ich vplyvu, čo vám pomáha sústrediť sa na to, na čom skutočne záleží.

- **Zvyšuje motiváciu:** Včasné dokončenie významnej úlohy vytvára pozitívny impulz a pocit úspechu, ktorý poháňa produktivitu po zvyšok dňa.

- **Znižuje prokrastináciu:** Tým, že sa najprv zameriate na tú najnáročnejšiu úlohu, obmedzíte prokrastináciu a úzkosť, ktorú so sebou prináša.

- **Zlepšuje time management:** "Zjedzte tú žabu!"podporuje strategické plánovanie a disciplinované vykonávanie, čo zlepšuje celkové riadenie času.

2.4 Eisenhower matrix

Eisenhowerova matica, tiež známa ako Eisenhowerova škatuľka alebo matica naliehavosti a dôležitosti, je nástroj na riadenie času pripisovaný Dwightovi D. Eisenhowerovi, 34. prezidentovi Spojených štátov amerických [2]. Bol známy svojou neuveriteľnou schopnosťou udržiavať produktivitu organizovaním svojich úloh do jednoduchých, ale účinných kategórií. Matica pomáha uprednostňovať úlohy na základe ich naliehavosti a dôležitosti, čo vedie k lepšiemu rozhodovaniu o tom, kam zamerať čas a energiu.

Ako to funguje

1. **Zoznam úloh:** Začnite zoznamom všetkých úloh, ktoré musíte dokončiť.
2. **Rozdeľte úlohy do štyroch kvadrantov:**
    - **Kvadrant 1: Naliehavé a dôležité (Urobte ako prvé):** Úlohy, ktoré si vyžadujú okamžitú pozornosť a majú významné dôsledky (napr. termíny, krízy, naliehavé problémy).
    - **Kvadrant 2: Dôležité, nie naliehavé (Plán):** Úlohy, ktoré sú dôležité, ale nevyžadujú okamžitú akciu. Tu sa hodí plánovanie, rozvoj a upevňovanie vzťahov.
    - **Kvadrant 3: Naliehavé, nie dôležité (delegát):** Úlohy, ktoré je potrebné urobiť čoskoro, ale sú menej dôležité. Toto sú často úlohy, ktoré si vyžadujú pozornosť, pretože sú naliehavé, ale nemusia nevyhnutne prispievať k dlhodobým cieľom.
    - **Kvadrant 4: Ani naliehavé, ani dôležité (eliminovať):** Činnosti, ktoré ponúkajú malú alebo žiadnu hodnotu a možno ich úplne odstrániť.
3. **Podľa toho stanovte priority:** Použite maticu ako pomôcku na stanovenie priorít a efektívne riadenie úloh. Zamerajte sa nielen na naliehavé úlohy, ale zabezpečte aj naplánovanie a dokončenie dôležitých, nie naliehavých úloh.
4. **Kontrola a úprava:** Pravidelne kontrolujte kategorizáciu podľa toho, ako sa úlohy a priority časom vyvíjajú. V prípade potreby upravte, aby ste udržali produktivitu a efektivitu.

Predpokladané výhody

- **Jasné stanovenie priorít:** Poskytuje jasný rámec pre stanovenie priorít úloh na základe úrovne ich naliehavosti a dôležitosti.
- **Znižuje preťaženie:** Kategorizáciou úloh môžete ľahšie zvládnuť pracovné zaťaženie a znížiť pocity preťaženia.
- **Zvyšuje produktivitu:** Zameriava vašu energiu na to, na čom skutočne záleží, čím zvyšuje produktivitu a efektivitu.
- **Zlepšuje riadenie času:** Pomáha rozlišovať medzi tým, čo je potrebné urobiť okamžite, a tým, čo možno naplánovať alebo delegovať.

2.5 Upravená Eat that Frog!

Mnou upravená metóda Eat that Frog! jej hlavný cieľ je rovnaký ako v prípade pôvodnej metódy najdôležitejšie je tiež splniť najdôležitejšiu úlohu ako prvú ale potom dávame dôraz i na deadline keďže pôvodná metóda nám hovorí, že by sme mali uprednostniť danú úlohu i keď nám niečo iné treba súrne spraviť. Táto metóda je dobrá hlavne pri začiatku, keď sa snažíme zvyknúť si na rozvrh a keď sme s rôznymi úlohami pozadu a musíme všetko dobiehať.

3 Užívateľská príručka

Aplikácia má módy a to pre anonymného užívateľa a pre registrovaného užívateľa. Rozdelenie je závislé na tom, či užívateľ má alebo nemá v aplikácií účet. Čo sa týka funkcionalít programu, anonymný užívateľ nemá prístup k práci s inými užívateľmi a jeho rozvrh sa ukladá len lokálne (na jeho počítací) takže oňho môže prsť.

3.1 Zo začiatku zľahka

Otvorenie aplikácie

Po otvorení aplikácie uvidíte tlačidlo na vytvorenie rozvrhu a side bar, ktorý môžete rozkliknúť a pomocou neho sa v aplikácie presúvať.

Side Bar

Má prekliky na orientáciu. Pre rozšírené vysvetlenie je potreba kliknúť na tri čiarky. Na ňom môžte nájsť cestu k svojmu rozvrhu, nastaveniam, príspevkom iných užívateľov, správe prihlásenia a nakoniec i k obrazovke, ktorá slúži na pokračovanie v aktivitách užívateľa,

3.2 Ako vytvoriť vlastný rozvrh a správne ho použiť

Na začiatok sa pozrieme na už vytvorený rozvrh a rozoberieme všetky jeho časti.

Obrázek 3.1 Ukážka vytvoreného rozvrhu

Môžme si všimnúť, že rozvrh pozostáva z dvoch typov aktivít a to z aktivít a denných aktivít. Poďme sa pozrieť bližšie na ich funkciu v rozvrhu.

Denná aktivita je časový úsek v rozvrhu, ktorý sa vysťahuje na špecifický deň v týždni alebo každý deň v týždni. Tento časový úsek spolu s jeho názvom je presne zadaný užívateľom. Ako príklad môžme uviesť Fitko z obrázka 3.1. Aplikácia negeneruje tento typ aktivity ale v rozvrhu ho len vyznačí ako svetlomodrý časový úsek. Čo sa týka generovania rozvrhu tento typ aktivity môžme vnímať ako prekážky v rozvrhu užívateľa, okolo ktorých my musíme vytvoriť rozvrh.

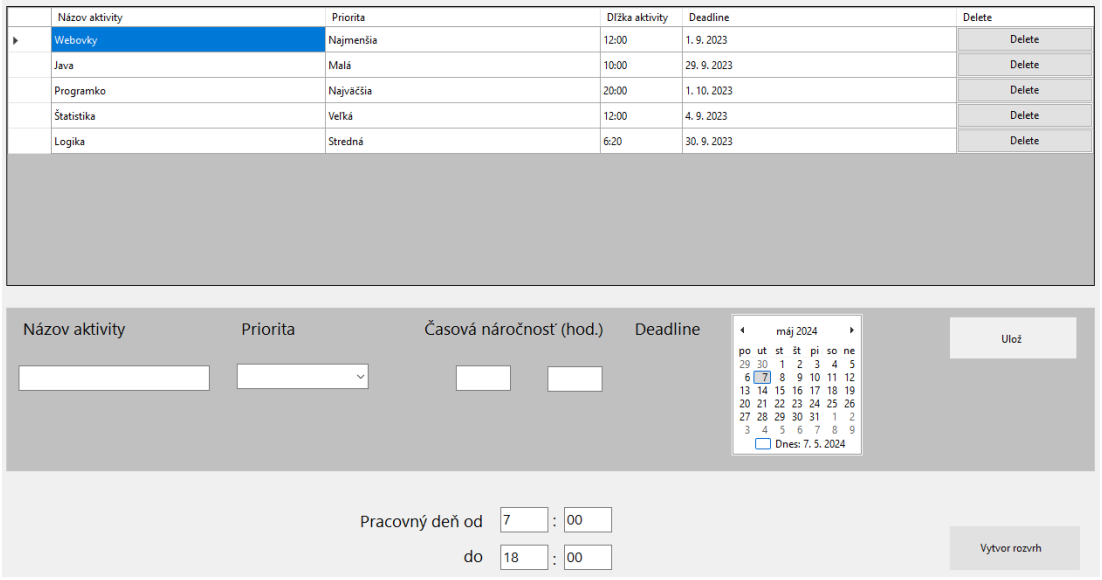
Aktivita taktiež spomínaná i ako normálna aktivita, je časový úsek v rozvrhu, ktorý je generovaný aplikáciou. Sú tu aktivity, na ktoré sa užívateľ chce zamerať, rozdelené na úseky, tak aby nepresahovali do denných aktivít a aby sa odohrávali len v užívateľom nastavených pracovných hodinách, čo je interval od kedy do kedy sa má rozvrh vytvoriť napríklad od 9:00 do 17:00. Môžme pozorovať, že aktivity majú možnosť troch farieb a to: zelená, červená a šedá. Každá z týchto farieb niečo znamená; zelená farba signalizuje, že danú aktivitu užívateľ úspešne dokončil a mal pritom zapnutú aplikáciu, červená hovorí, že užívateľ danú aktivitu nesplnil alebo nemal zapnutú aplikáciu a šedá farba znamená, že koniec danej aktivity ešte nenastal a môže prísť buď v nasledujúcom dni alebo ešte v priebehu súčasného dňa, pred jeho koncom.

3.2.1 Vytvorenie rozvrhu

Ako prvý krok pri vytváraní rozvrhu je **pridanie Aktivít**. Na ich pridanie je potrebné vyplniť všetky polia pre vytvorenie Aktivity (názov aktivity, priorita,

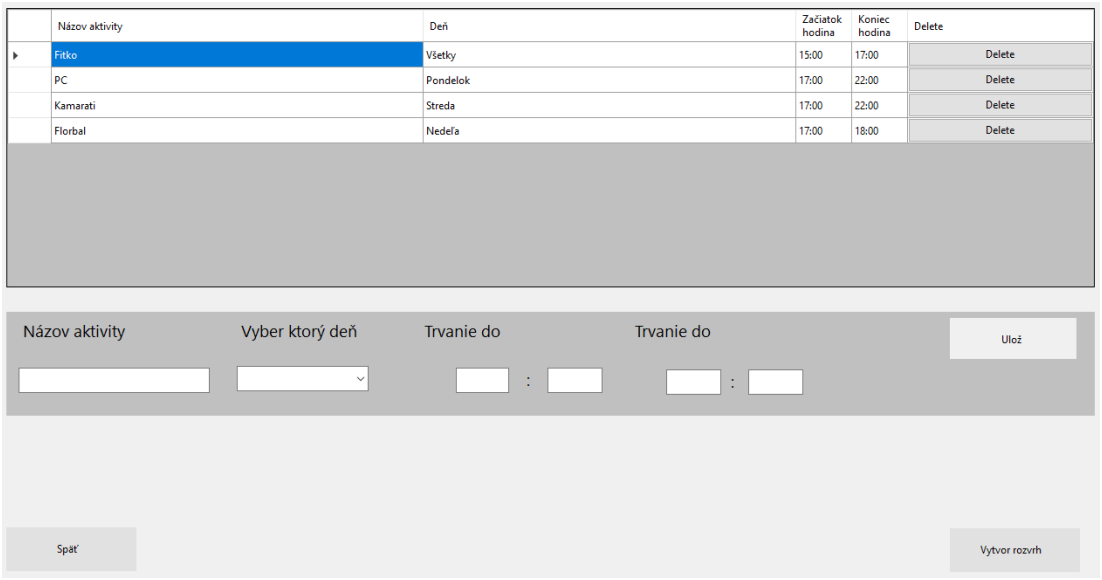
časová náročnosť a deadline) potom kliknúť ULOŽIŤ, aktivita sa zobrazí v tabuľke. Odkiaľ vie byť odstránená po kliknutí na tlačidlo DELETE. Všetku polia musia byť vyplnené správnymi hodnotami: deadline musí byť pred dneškom aby bola aktivita pridaná do rozvrhu, čas musí byť číslo a minutý musia byt menšie ako 60.

Na tejto obrazovke sa nachádza i možnosť zmeniť pracovné hodiny, čo je interval od kedy do kedy sa má rozvrh vytvárať napríklad od 9:00 do 17:00. Potom čo je užívateľ spokojný s aktivitami musí stlačiť tlačidlo *Vytvor rozvrh* a pokračuje zadávaním denných aktivít.



Obrázek 3.2 Ukážka obrazovky pridania aktivity

Pridanie Denných aktivít. Ako prvé treba vyplniť všetky polia platnými hodnotami pre vytvorenie Dennej aktivity (názov aktivity, deň, čas od, čas do) potom kliknúť ULOŽIŤ. Aktivita sa zobrazí v tabuľke odkiaľ vie byť odstránená po kliknutí na tlačidlo DELETE. Tu je potrebné zarezervovať dni, počas ktorých nechceme aby sa nám vytváral rozvrh. Taktiež je potrebné mať na vedomí čas na prepravu na denné aktivity (napr. do školy dochádzam 30 minút). Potom čo je užívateľ spokojný s dennými aktivitami musí stlačiť tlačidlo *Vytvor rozvrh* a pokračuje výberom algoritmu na vytvorenie rozvrhu.



Obrázek 3.3 Ukážka obrazovky pridania dennej aktivity

Výber algoritmu. Ako na stránke uvidí užívateľ výber 4 metód s krátkym popisom ako sú aktivity radené. Krátky popis podľa môjho názoru povie užívateľovi viac ako keby tam bol veľký popis presnej funkcionality metódy, ktorú by mal už poznať po prečítaní tejto práce.

- **Eat That Frog!:** Veľké aktivity na začiatku, priorita > časová náročnosť, na deadline sa vôbec nepozerá.
- **80/20 rule:** Deadline > priorita > časová náročnosť.
- **Upravená Eat That Frog!:** Deadline > časová náročnosť > priorita.
- **Einsenhower:** Algoritmus vyradí najmenej potrebné aktivity (aktivity s najnižšou prioritou) a potom zostaví rozvrh ako pri 80/20 rule. Deadline > priorita > časová náročnosť.

Finálne zobrazenie. Po vybratí metódy na tvorbu rozvrhu sa užívateľovi zobrazí finálny rozvrh, ktorý sa súčasne uloží lokálne do súboru. Odporúčam vytvorenie rozvrhu viacero krát ziterovať až pokiaľ nie je užívateľ spokojný. Lebo odhadnutie správneho počtu hodín a náročnosti aktivít je relatívne náročne a väčšinou sa nepodarí vytvoriť rozvrh, s ktorým je užívateľ spokojný, na prvý pokus. Po vytvorení rozvrhu, s ktorým je spokojný ho môže užívateľ používať celý týždeň. Iterovanie rozvrhu je podporované zapamätaním si aktivít v tabuľkách, z ktorých užívateľ vytvoril rozvrh. Takže ak užívateľ chce zmeniť iba jednu aktivitu nie je potrebné opätovné zadanie všetkých aktivít.

3.2.2 Pokračovanie v aktivitách

Ak má už užívateľ vytvorený rozvrh tak môže pokračovať v práci na svojom pláne. V Sidebare klikne na tlačidlo Pokračovať. Tam sa mu zobrazí názov aktivity a od kedy do kedy jeho aktivita, ktorú má teraz vykonávať. Po spustení aktivity ju môže užívateľ zastaviť. Nato aby sa aktivita započítala ako splnená, musí

byť spustená na konci časovej doby. Po úspešnom dokončení aktivity sa aktivita zobrazí v rozvrhu nazeleno ak užívateľ aktivity nespravil tak je červená. Pokiaľ na aktivitu ešte neprišiel čas alebo je užívateľ akurát v procese jej plnenia je šedá. Jedinou výnimkou sú Denné aktivity tie sú vždy svetlomodré, keďže pri niektorých aktivitách u seba užívateľ nemá notebook (napr. fitko).

Ak sme v nastavenia povolili Pomodoro tak počas spustenej aktivity sa môže užívateľ riadiť týmto systémom. Po úspešnom splnení celej aktivity sa užívateľovi zobrazí možnosť zazdieľať svoj úspech v podobe príspevku, avšak na to musí byť užívateľ prihlásený (viac v časti 3.3).

3.2.3 Nastavenia

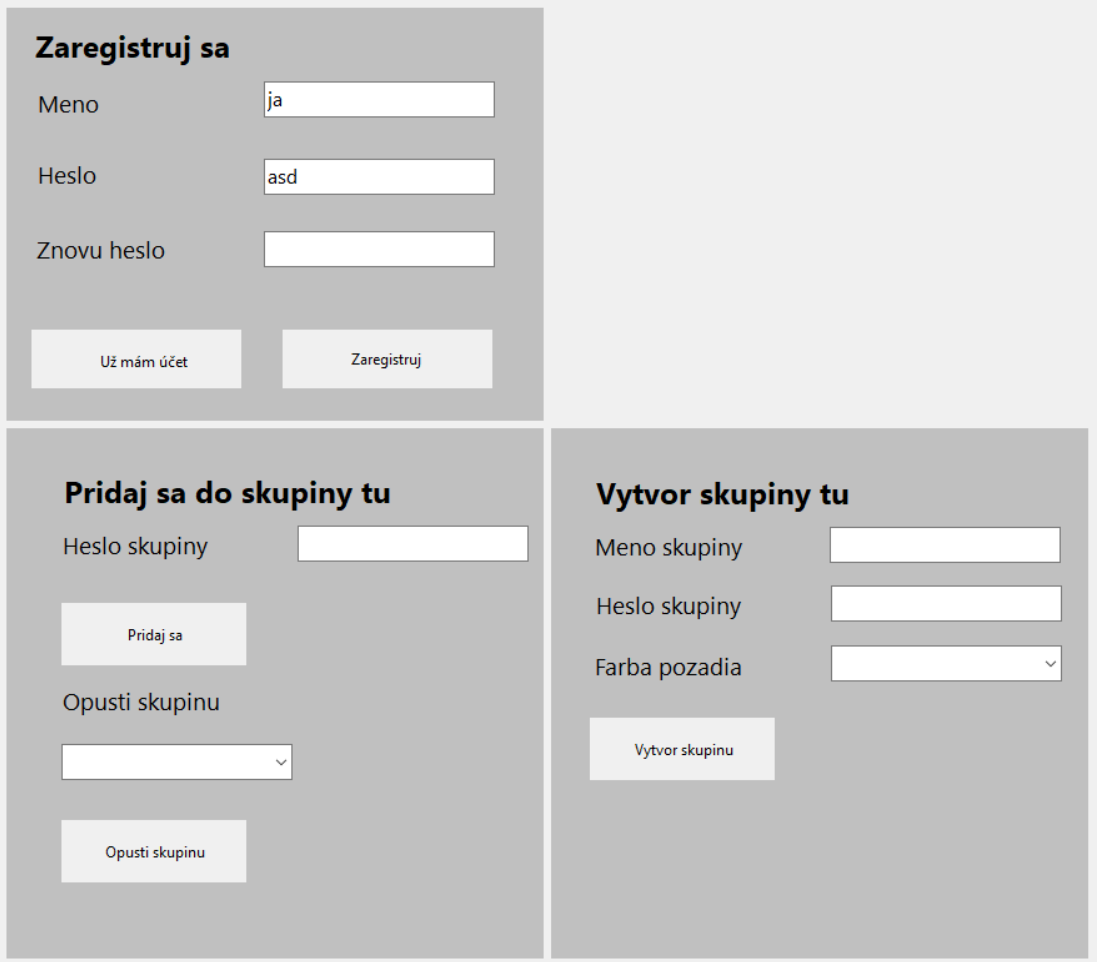
V nastaveniach môže užívateľ povolil Pomodoro a nastaviť časové intervaly na Pomodoro. Taktiež si tam vie pozrieť i súčasné hodnoty Pomodora.

3.3 Práca s skupinami a inými užívateľmi

V tejto sekcii sa zameriame na časť aplikácie, ktorá dovoľuje užívateľovi interagovať s inými užívateľmi. Na to aby sme to užívateľovi dovolili sa najskôr musí užívateľ prihlásiť alebo si vytvoriť účet. Po tom ako je užívateľ prihlásený otvorí sa mu viacero nových funkcií a to: vytvorenie skupiny, pridanie sa už do vytvorenej skupiny a zobrazovanie príspevkov. Teraz sa pozrime bližšie na tieto funkcionality.

Vytvorenie a prihlásenie užívateľa. Akonáhle sa užívateľ dostane na túto obrazovku uvidí možnosť sa prihlásiť a ak nemá účet, môže použiť tlačidlo na registráciu. Po úspešnom prihlásení sa obrazovka zmení a ponúkne mu viacero funkcionalít. A to:

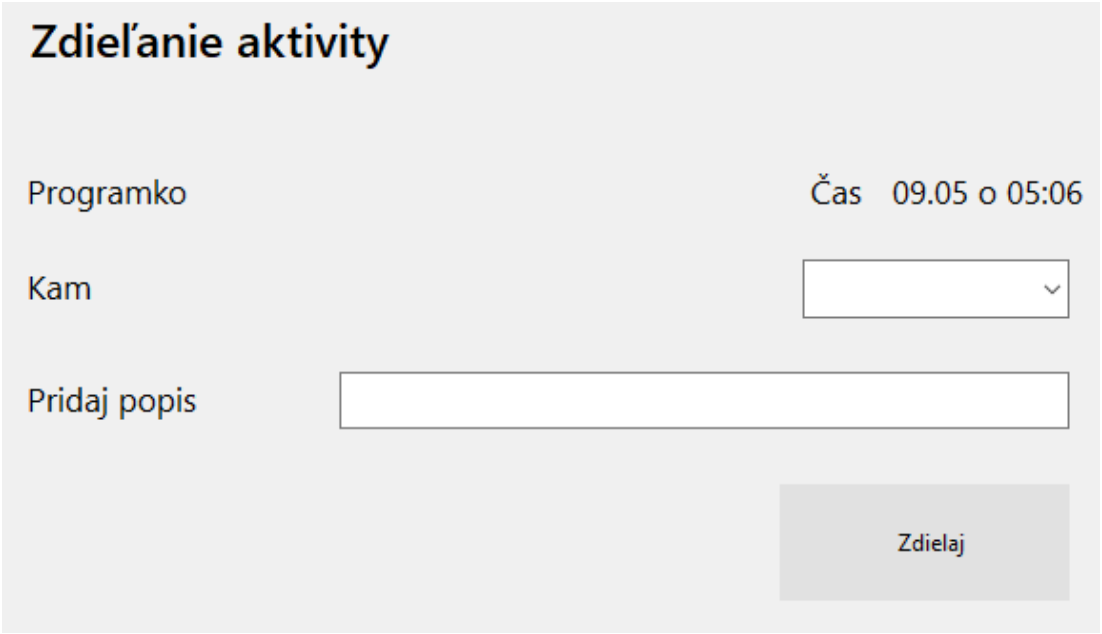
- **Vytvorenie skupiny:** Zahŕňa, názov skupiny, heslo a farbu pozadia príspevkov pre danú skupinu.
- **Pridanie sa do cudzej skupiny:** Na stránke je taktiež pole pre zadanie hesla skupiny, do ktorej sa chce užívateľ prihlásiť.
- **Možnosť opustenia skupiny:** Tole, v ktorom vie užívateľ vybrať skupinu ktorú chce opustiť.
- ***Zverejnenie príspevku:*** *Táto forma zverejnenia príspevku v finálnej verzií nebude* existuje len pre ľahkosť kontroly správania programu sa tu tiež nachádza i tlačidlo na zverejnenie príspevku. Príspevok sa dá taktiež zverejniť ako už spomínané po dokončení celej aktivity.



Obrázek 3.4 Ukážka obrazovky po prihlásení

Socials Na tejto obrazovke si užívateľ vie zobraziť, zdieľané aktivity iných užívateľov ale taktiež vidí i svoje. Príspevky sú radené časovo (najnovšie sú hore) a sú rozlíšené farbami pre lepšiu viditeľnosť na základe skupín. Avšak na to aby ich videl sa musí najskôr prihlásiť.

Zieľanie príspevku Po úspešnom dokončení aktivity, sa môže užívateľ rozhodnúť, zdieľať svoj úspech vo forme príspevku. Pri zdieľaní vie užívateľ vybrať kam chce príspevok zdieľať a s akým komentárom, z dôvodu podporenia rôznych komentárov v rôznych skupinách sa príspevok zdieľa len do vybranej skupiny. Ale užívateľ po jednom zdieľaní vie vybrať inú skupinu a napísať iný komentár a tým ju za zdieľať i inde.



Obrázek 3.5 Ukážka obrazovky pri zdieľaní

4 Implementácia programu

Najprv sa pozrieme na implementáciu algoritmu tvorby rozvrhu a ako tam zapracovávam metódy vysvetlene v predchádzajúcej kapitole. Potom vysvetlíme tvorbu grafickej aplikácie a ako implementujem jej rôzne funkcie. Na záver sa v tejto kapitole pozrieme na komunikáciu aplikácie s databázou, ktorá slúži na zdieľanie splnených úloh a jej štruktúru.

4.1 Generálna štruktúra

Celý program vieme rozdeliť do troch častí: prvé dve su napísané v C#, sú to Win-Form UI a algoritmus na vytváranie rozvrhov a posledná je napísaná v jazyku Python: server s Python knižnicou Flask, ktorý zaistuje prístup k databázi.

Teraz si ukážeme C# štruktúru kódu, skladá sa z dvoch priečinkov A.1. Priečinok **buildingTimeTable** obsahuje len jeden zdrojový kód, ktorý má na starosti algoritmus na vytvorenie rozvrhu. Priečinok **Time** obsahuje zdrojové kódy pre užívateľské rozhranie (Form1.cs, Form1.Designer.cs, Form1.resx, MyPost.cs, MyPost.Designer.cs, MyPost.resx), komunikáciu s databázou (My_SQL_comunication.cs) a triedy Activita.cs, DailyActivities.cs, FinalActivitaInTimeTable.cs a MyPostClass.cs.

- **{base directory}**
  - **buildingTimeTable**
    - TimeTableBuilder.cs
  - **Time**
    - Activita.cs
    - DailyActivities.cs
    - FinalActivitaInTimeTable.cs
    - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
    - MyPostClass.cs
    - My_SQL_comunication.cs
    - **Post**
      - MyPost.cs
      - MyPost.Designer.cs
      - MyPost.resx
    - Program.cs
    - **Properties**
    - **Resources**

4.2 Reprezentácia rozvrhu v progrmame

V aplikácií pracujeme s stringovou reprezentáciou rozvrhu. Je vhodná na ukladanie do súborov a databázy. Reprezentácie vyzerá následovne:
Formát stringového dňa:
`XXX.7.15.A.n;Fitko.15.17.O.s;Florbal.17.18.O; XXX.18.18.A.n;`

- **;** na konci aktivity aby sa dalo rozlíšiť medzi samostatnými aktivitami
- **.** rozdeľovač medzi informáciami o aktivite
- **XXX** prvá súčasť - názov aktivity
- **7.15** hodina a minúta kedy sa aktivita odohráva
- **A/O** signalizuje aktivitu alebo prekážku(dennú aktivitu)

Normálna aktivita ma ešte na koniec (pred ;) pridané ".n;"čo signalizuje že je to normálna aktivita a ešte nebola prevedená(n). Denná aktivata ma na koniec pridané ".s;"čo signalizuje, že je to prekážka(obsticle) a že sa nemá rátať medzi nesplnené ak prejde čas. Ale to je robene vždy na základe toho či to je aktivita alebo prekážka.

Tento tvar sa požíva v celom projekte na reprezentáciu rozvrhu v stringovej podobe, ukladá sa v tejto podobe i do súboru a i do databázy.

4.3 Tvorba grafickej aplikácie - Win-Form UI

Win-Form kód zodpovedá za interakciu s užívateľom. Jeho hlavnou úlohou je vytváranie obrazoviek, ktoré užívateľ vidí a logika medzi obrazovkami.

4.3.1 Používané triedy

`Activita` je trieda, v ktorej ukladám informácie o aktivitách; presnejšie názov aktivity, priorita, časová náročnosť v hodinách, časová náročnosť v minútach a deadline. Táto trieda je niekdy spomínaná ako normálna aktivita alebo len ako aktivita.

`DailyActivities` je trieda do, ktorej ukladám informácie o denných aktivitách; presnejšie názov aktivity, deň v týždni, čas od začiatku aktivity v hodinách a minútach, čas do konca aktivity v hodinách a minútach. Táto trieda je niekdy spomínaná ako prekážka alebo denná aktivita.

`FinalActivitaInTimeTable` je trieda, do ktorej ukladám info o finálnej verzie aktivít, to sú tie, ktoré sú už spracované algoritmom. Trieda obsahuje názov aktivity, čas od začiatku aktivity, čas do konca aktivity, číslo, ktoré hovorí o dni kedy je aktivita vzhľadom k dňu vytvorenia rozvrhu a premennú, ktorá nám ukazuje či je aktivita hotová, nie je hotová alebo ešte na ňu nenastal čas.

Tieto triedy neobsahujú žiadne metódy slúžia len na ukladanie údajov.

Ale `Activita` a `DailyActivities` sú prepojené s dailyActivitiesBindingSource a activitaBindingSource, čo nám dovolí zobrazovať aktivity v príslušnom dataGridView.

4.3.2 Vstup do programu

Hlavným vstupom do programu je konstruktor `Form1()`, ktorý inicializuje všetky komponenty, ktoré boli použité v FormDesigneri. Táto inicializácia nám zavolá `Form1_Load()` nastaví prvú obrazovku a taktiež vloží prvotné dáta do activitaBindingSource a dailyActivitiesBindingSource. Taktiež sa pokúsi načítať zo súboru s názvom "time table", deň jeho vytvorenia a pracovné hodiny.

Teraz sa pozrieme na konkrétne metódy v Form1.cs. Veľa metód je kóde reagujú na akcie užívateľa. Spomeniem tu tie najpodstatnejšie.

4.3.3 Ukladanie údajov do súborov

Pri používaní programu si užívateľ vytvára rozvrh, ale aby ho nemusel vytvárať pri každom zapnutí aplikácie, údaje o rozvrhu sa ukladajú vždy lokálne na počítač. A potom pri opätovnom otvorení aplikácie sa údaje načítajú. Tieto funkcie zaručujú, že sa korektne uloží rozvrh a pre neho potrebné údaje aby sa dali zobraziť i po opätovnom otvorení aplikácií. Všetky tieto metódy sa volajú ako reakcia na vstupy od užívateľa.

Prvá metóda sa používa pri prvotnom vytvorení rozvrhu, keď je potreba uložiť všetky informácie.

Program 1 SaveTimeTable: Uloží informácie o rozvrhu do súboru bez zmeny dňa vytvorenia rozvrhu a pracovných hodín. Volá sa napríklad po dokončení aktivity kedy sa mení stav aktivity na dokoncenú.

```
private void SaveTimeTable(string result);
```

Táto metóda sa používa ak užívateľ splní aktivitu a treba iba zmeniť časť súboru.

Program 2 SaveChangesToFile: Táto metóda sa volá ak je potreba uložiť rozvrh lokálne do súboru. Pri tomto ukladaní sa zmení i deň vytvorenia rozvrhu a i pracovné hodiny.

```
private void SaveChangesToFile();
```

Formát uložených dát:

```
+7+1.7.15.A;Fitko.15.17.O;Florbal.17.18.O;
1.18.18.A;+8+1.7.11,033334.A;.....
```

- **+7+** rozdeľuje dni s indikátorom, ktorý deň sa má aktivita odohrávať
- **;** oddeľuje aktivity v jednom dni navzájom od seba
- **.** rozdeľuje informácie v aktivite navzájom od seba

4.3.4 Zobrazenie rozvrhu

Je zaistené mnohými funkciami, ktoré spracujú rozvrh z stringového vstupu a zobrazia ho. Rozdelenie stringu na aktivity sa prevedie len raz pre každú

aktualizáciu vstupu, alebo po spustení aplikácie ak už existuje rozvrh a aplikácia ho načíta z súboru.

Program 3 Vytvorí verziu rozvrhu v activitiesInTimeTable, čo je list plnený inštanciám triedy FinalActivitaInTimeTable, z globálneho stringu timeTable. Tento string rozdelí na časti a postupne z nich vytvára jednotlivé inštancie FinalActivitaInTimeTable, ktoré vkladá do activitiesInTimeTable.

```
private void GeneratingTimeTable();
```

Program 4 Z vopred vytvoreného rozvrhu v podobe activitiesInTimeTable vytvára postupne komponenty typu lable a tie pokladá na obrazovku.

```
private void ShowTimeTable();
```

4.3.5 Kontroly pre prebiehajúcu aktivitu

Kontroly sú zaručené prevažne nasledujúcimi metódami, ktoré kontrolujú či prebiehajúca aktivita je splnená a ak je Pomodoro zvolené v nastaveniach tak sa skontroluje i aká fáza Pomodora má byť spustená.

Program 5 checkForCompletion_Tick: Táto metóda sa zavolá vždy keď je na pozadí spustený timer *checkForCompletion* a ak je aktivita spustená tak skontroluje či nie je čas skončiť v intervale 1 milisekunda. Tu sa taktiež zavolá i kontrola Pomodora (pomodoroCheck). Ak aktivita úspešne skončí tak jej nastaví premennú "checker"v triede FinalActivitaInTimeTable na “d“ čo znamená, že je úspešne ukončená a uloží zmeny. Potom zobrazí obrazovku s ďalšou aktivitou.

```
private void checkForCompletion_Tick(object sender, EventArgs e);
```

Program 6 pomodoroCheck: Skontroluje či je Pomodoro zvolené v nastaveniach. Ak je tak rozhodne aká fáza Pomodora je súčasne v procese, nastaví text, ktorý sa zobrazí a skontroluje či už fáza nemá skončiť.

```
private void pomodoroCheck(float currentTime);
```

4.3.6 Kontrola aktivít rozvrhu

Pozostáva hlavne z dvoch funkcií. Kontrolujú či aktivity boli splnené a ak nie tak im priradí červenú farbu. Tieto kontroly sa spúšťajú pri zobrazení rozvrhu a pri spustení aplikácie. V tejto sekcii sa taktiež vyskytujú metódy na určenie súčasnej aktivity, ktoré sa používajú v celej aplikácií.

Program 7 GetCurrentActivity: Po zavolaní vráti súčasnú aktivitu, ktorú získa prechodom cez aktivity pričom začína pri poslednou známou súčasnou aktivitou.

```
private FinalActivitaInTimeTable GetCurrentActivity();
```

Program 8 CheckForRed: prejde cez všetky aktivity z activitiesInTimeTable a tie, ktoré sú pred súčasnou aktivitou a nemajú zatiaľ žiadne označenie (“n“) označí "x", čo signalizuje, že aktivita nebola splnená.

```
private void CheckForRed();
```

4.3.7 Príprava na tvorbu rozvrhu

Následné funkcie, sú zodpovedné za reakciu na voľbu metódy užívateľom a za predpripravenie dát pre algoritmy tvorby rozvrhu.

Program 9 Všetky tieto metódy reagujú na užívateľskú voľbu systému pre vytvorenie rozvrhu.

```
private void choose_system_Einsenhower_Click(object sender,
                              EventArgs e);
private void choose_system_MyOwn_Click(object sender,
                              EventArgs e);
private void choose_system_80_20_Click(object sender,
                              EventArgs e);
private void choose_system_EatTheFrog_Click(object sender,
                              EventArgs e);
```

Najprv spracujem aktivity a denné aktivity, vytvorím príslušnú triedu (EatTheFrog, Pareto, MyOwn a Eisenhower) a potom zavoláme .Run(Activities, DailyActivities, workhours) Tuto metódu budem bližšie rozoberať neskor. Potom už sa iba uloží výsledok a zavolajú sa metódy na vykreslenie rozvrhu na obrazovku.

Program 10 ProcessActivities a ProcessDailyActivities:
Tieto metódy spracujú aktivity a denné aktivity z BindingSourceov. Spracuje ich do formátu vhodného pre algoritmy.

```
private string ProcessActivities();
private string ProcessDailyActivities();
```

4.3.8 Zobrazenie prvkov na obrazovkách

Tieto dve funkcie skrývajú prvky na obrazovkách, aby sme mali vždy čisté okno pred tým ako začneme zobrazovať prvky, ktoré sú potrebné pre každú obrazovku. Prvky sú zoskupené podľa obrazoviek a dynamicky generované elementy sa pri tomto procese rovno ničia.

Program 11 diasableAll: Schová všetky inicializované Komponenty.

```
private void diasableAll();
```

Program 12 disableCreatingActivity: Schová komponenty, ktoré sú zobrazené pri zadávaní aktivít.

```
private void disableCreatingActivity();
```

4.3.9 Kontroly polí pri zadávaní aktivit

Na kontrolu vyplnenia polí a či sú v nich platné argumenty slúžia tieto funkcie. Sú to funkcie, ktoré reagujú na stlačenie tlačidla užívateľom a pri zadaní neplatných argumentov sa zobrazí užívateľovi upozornenie.

Program 13 skontroluje všetky polia v ktorých má byť informácia či tam je a či je v správnom formáte. A ak všetko spĺňa podmienky tak vytvorí Aktivity/DailyActivitu a pridá ju do príslušného BindingSourceu (dailyActivitiesBindingSource, activitaBindingSource)

```
void ulozPrekazkyVTyzdni_Click(object sender, EventArgs e);
void ContinueToDailyActivities_Click(object sender, EventArgs e);
```

4.4 Algoritmus na tvorbu rozvrhu

Táto sekcia sa pozrie na implementáciu algoritmov a ako sú v skutočnosti použité pri tvorbe rozvrhov. Taktiež sú tu znovu definované triedy Activita a DailyActivities, z dôvodu potreby čiastočne inej štruktúry ako v UI.

`Activita` je trieda do, ktorej ukladám informácie o aktivitách, presnejšie názov aktivity, priorita, časová náročnosť v hodinách, časová náročnosť v minútach a deadline. V konštruktore nastáva priradenie všetkých dat.

`DailyActivities` je trieda do, ktorej ukladám informácie o denných aktivitách, presnejšie názov aktivity, deň v týždni, čas od v hodinách a minútach, čas do v hodinách a minútach. V konštruktore nastáva priradenie všetkých dat.

Tento kód má zodpovednosť i za skutočnú tvorbu rozvrhu. Pri tvorby rozvrhu používam Template návrhový vzor, ktorý nám pomáha vytvárať rôzne štruktúry rozvrhu. Template návrhový vzor je návrhový vzor v objektovo orientovanom programovaní, ktorý definuje základný algoritmus v metóde šablóny a necháva podtriedy pretláčať niektoré kroky tohto algoritmu bez zmeny jeho štruktúry.

4.4.1 Hlavná Template metóda

Táto metóda definuje základný algoritmus a špecifický postup volaní funkcií, ako sa má rozvrh vytvárať.

Program 14 AbstractTimeTable: Template trieda, s jej premennými a metódami

```
public abstract class AbstractTimeTable
{
    protected List<Activity> activities = new List<Activity>();
    protected List<DailyAc> dailyActivities = new List<DailyAc>();
    protected string timeTable = "";
    protected List<List<DailyAc>> obsticlesByDay = new ();
    protected AlgoritmsForTTCreateation u = new ();

    protected abstract void ProcessActivities(string act);
    protected abstract void ProcessDailyActivities(string daily);
    protected abstract void CreateTimeTable(string workingHours);
    protected abstract List<Activity> SortActivitiesForAlgotithm
                              (List<Activity> activities);
    protected abstract string Disconnect();

    // The 'Template Method'
    public string Run(string act, string daily,
                              string workingHours)
    {
            ProcessActivities(act);
            ProcessDailyActivities(daily);
            CreateTimeTable(workingHours);
            return Disconnect();
    }
}
```

4.4.2 Podtriedy, ktoré dedia od Template metódy

Tu sa pozrieme na špecifické podtriedy, ktoré upravujú metódy aby vedeli správne vytvoriť rozvrh.

Špecifické metódy sú:

Program 15

```
public class Eisenhower : AbstractTimeTable
public class EatTheFrog : AbstractTimeTable
public class Pareto : AbstractTimeTable
public class EatTheFrogUpdate : AbstractTimeTable
```

Teraz pozrieme na všetky metódy, ktoré sú v základnej Template triede.

Program 16 ProcessActivities: Rozdelí vstup do tried aktivít, roztriedi a uloží ich do listu.

```
protected override void ProcessActivities(string act)
```

Program 17 ProcessDailyActivities: Rozdelí vstup do tried aktivít, roztriedi a uloží ich v dailyAcList. Taktiež rozdelí tieto aktivity podľa dní do obsticlesByDay.

```
protected override void ProcessDailyActivities(string daily)
```

Program 18 CreateTimeTable: Použie triedu AlgoritmsForTTCreateation na vytvorenie rozvrhu. Len volá: `u.CreatePlan(activities, obsticlesByDay, workingHours);` a výstup uloží do timeTable.

```
protected override void CreateTimeTable(string workingHours)
```

Program 19 Disconnect: Táto metóda iba vracia timeTable.

```
protected override string Disconnect()
```

Program 20 SortActivitiesForAlgorithm: Táto metóda ma v každej triede inú podobu ale celkovo je jej podoba rovnaká. Pomocou metód z knižnice System.Linq sa zoradí list na základe požiadaviek danej časovej metódy (napríklad Paretto: deadline > priorita > časová náročnosť)

```
protected override List<Activity> SortActivitiesForAlgotithm
                            (List<Activity> activities)
```

4.4.3 Algoritmus tvorby rozvrhu

Trieda `AlgoritmsForTTCreateation` je zodpovedná za väčšinu metód, ktoré používam na vytvorenie rozvrhu, nemá žiadne premenné. Pracuje hlavne už s utriedenými aktivitami na základe zvolenej metódy. Celkovo zodpovedá za vytvorenie rozvrhu, spracovanie dát na jeho tvorbu a taktiež i finálne odoslanie rozvrhu do grafickej aplikácie.

Jej metódy sú:

Program 21 SetNewDay: Pridá „+number of day+“ do finalneho stringu. Tento rozdeľovač signalizuje nový deň.

```
public string SetNewDay(int wk, string timeTable)
```

Program 22 SetAcctivity: Pridá celú jednu aktivitu do finálneho stringu.

```
public string SetAcctivity(string nameOfActivity,
                            float timeToStart, float timeToEnd,
                            bool isNormAc, string timeTable)
```

Program 23 CreatePlan: Vytvára už presný plán. Najprv spracuje workingHours a potom prechádza cez aktivity stále a určuje kedy v dni sa majú vyskytnúť. Tu sa veľké aktivity rozdelia na menšie. Aktivity sa potom pridávajú do rozvrhu cez `SetAcctivity()` Výsledkom je finálny string timeTable .

```
public string CreatePlan(List<Activity> activities,
                            List<List<DailyAc>> obsticlesByDay,
                            string workingHours)
```

Program 24 CreateDailyObsticles: Roztriedi denne aktivity do príslušných dní.

```
public List<List<DailyAc>> CreateDailyObsticles
                            (List<DailyAc> dailyAcList)
```

Program 25 SplitActivities: Rozdelí aktivity, keď prvý krát prídu z stringu a vráti list príslušných aktivít ako triedu.

```
public List<Activity> SplitActivities(string act)
```

Program 26 SortDailyActivitiesForAlgotithm: Zoradí denné aktivity do poradia v akom prebiehajú.

```
public List<DailyAc> SortDailyActivitiesForAlgotithm
                            (List<DailyAc> dc)
```

4.5 Databáza a komunikácia s ňou

Keďže aplikácia umožňuje užívateľom zdieľať ich úspechy medzi sebou navzájom, tak máme potrebu to niekde ukladať. Zaručujeme to použitím databáze. Z dôvodu bezpečnosti databáze nepristupujeme na ňu priamo ale používame na to Flask, čo je python knižnica. Komunikácia medzi aplikáciou a Flaskom prebieha pomocou REST API.

REST API je sada pravidiel, ktoré umožňujú rôznym softvérovým aplikáciám komunikovať medzi sebou prostredníctvom internetu. Umožňuje jednej aplikácii požiadať o údaje alebo služby od inej aplikácie a získať ich vo formáte, ktorý je ľahko spracovateľný napríklad JSON, ktorý používame my. REST využíva štandardné metódy HTTP, ako sú GET, POST, PUT a DELETE, na manipuláciu s dátami.

Všetky možné dotazy v API:

- `/`
  - **Funkcia:** Táto cesta slúži na testovanie pripojenia k serveru.
  - **Vráti:** Textový reťazec "Hello from Flask!".
- `/getGroupName/id_group/<string:id_group>/`
  - **Funkcia:** Získava názvy skupín na základe ID skupiny.
  - **Vráti:** JSON reprezentáciu názvov skupín pre dané ID skupiny.
- `/joinGroup/meno/<string:jmeno>/pass/<string:passw>/`

  - **Funkcia:** Pridáva používateľa do skupiny na základe mena a hesla skupiny.
  - **Vráti:** "added" ak pridanie bolo úspešné, inak chybový kód.

- `/noDoublePost/meno/<string:jmeno>/comment/<string:comment>/activityName/<string:activityName>/`
  - **Funkcia:** Kontroluje, či používateľ už nepridal daný príspevok k danému podujatiu.
  - **Vráti:** "noDoublePost" s hodnotou "n", ak príspevok už existuje, inak "y".

- `/addPost/meno/<string:jmeno>/group/<string:group>/when/<string:when>/actName/<string:actName>/comment/<string:comment>/`
  - **Funkcia:** Pridáva príspevok do databázy a priradzuje ho k určitej skupine.
  - **Vráti:** "create" ak pridanie bolo úspešné, inak chybový kód.

- `/leaveGroup/meno/<string:nameG>/userName/<string:userName>/`
  - **Funkcia:** Odstraňuje používateľa zo skupiny na základe mena skupiny a používateľa.
  - **Vráti:** "left" s hodnotou "y", ak používateľ úspešne opustil skupinu, inak "n".

- `/noDoubleGroup/pass/<string:passw>/name/<string:nameG>/`
  - **Funkcia:** Kontroluje, či neexistuje skupina s rovnakým heslom alebo názvom.
  - **Vráti:** "noDoubleGroup" s hodnotou "n", ak skupina s rovnakým heslom alebo názvom už existuje, inak "y".

- `/createGroup/meno/<string:jmeno>/popis/<string:popis>/color/<string:color>/pass/<string:passw>/`
  - **Funkcia:** Vytvára novú skupinu s definovaným menom, popisom, farbou a heslom.
  - **Vráti:** "create" ak vytvorenie bolo úspešné, inak chybový kód.

- `/getGroup/meno/<string:jmeno>/`
  - **Funkcia:** Získava všetky skupiny, ktorých je používateľ členom.

  - **Vráti:** JSON reprezentáciu všetkých skupín, ktorých je používateľ členom.

- `/getPost/meno/<string:jmeno>/`
  - **Funkcia:** Získava všetky príspevky používateľa.
  - **Vráti:** JSON reprezentáciu všetkých príspevkov používateľa.

- `/login/heslo/<string:heslo>/meno/<string:jmeno>/`
  - **Funkcia:** Overuje prihlasovacie údaje používateľa.
  - **Vráti:** "log" s hodnotou "y", ak sú prihlasovacie údaje platné, inak "n".

- `/noDoubleUser/meno/<string:jmeno>/`
  - **Funkcia:** Kontroluje, či používateľ s daným menom už existuje.
  - **Vráti:** "noDoubleUser" s hodnotou "n", ak používateľ s daným menom už existuje, inak "y".

- `/register/heslo/<string:heslo>/meno/<string:jmeno>/sch/<string:sch>/`
  - **Funkcia:** Registruje nového používateľa s definovaným menom, heslom a rozvrhom.
  - **Vráti:** "reg" ak registrácia bola úspešná, inak chybový kód.

- `/get/sched/meno/<string:jmeno>/`
  - **Funkcia:** Získava rozvrh používateľa.
  - **Vráti:** JSON reprezentáciu rozvrhu používateľa.

- `/set/sched/meno/<string:jmeno>/sch/<string:sch>/`
  - **Funkcia:** Nastavuje rozvrh používateľa.
  - **Vráti:** Textový reťazec "sch set".

4.5.1 SQL databáza

Najskôr si predstavíme SQL databázu a tabuľky, ktoré obsahuje.

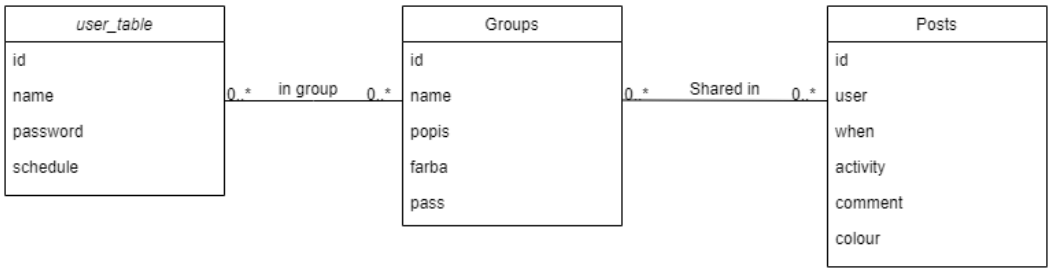
`user_table` Tabuľka určná na ukladanie údajov o užívateľoch. Ukladám tam jeho užívateľské meno, heslo a jeho rozvrh.

`Groups` V tejto tabuľke mám uložené informácie o skupinách a to presnejšie ich meno, popis, farbu a taktiež i heslo.

`Posts` Táto tabuľka je určená na ukladanie príspevkov, obsahuje meno užívateľa, ktorý príspevok vytvoril, kedy ho vytvoril, názov aktivity, farbu pozadia príspevku a popis, ktorý dal používateľ príspevku.

`in_group` Obsahuje prepojenia medzi užívateľmi a skupinami.

`shared_in` Tu sú uložené prepojenia medzi postami a skupinami.



Obrázek 4.1 UML model databáze

4.5.2 Python Flask

Na komunikáciu môjho programu s databázou som použil použil python a na ňom knižnicu Flask, ktorý beží na serveri a tým mi umožňuje komunikovať s databázou.

Čo sa týka kódu v pythone zachytávam http requesty podľa URI a na základe toho, ktorý príde spravím na databáze príslušné kroky.

Príklad kódu v pythone:

Program 27 join_group: Tu napríklad je pridanie užívateľa do skupiny. Flask pomocou mechanizmu dekoratorov predá hodnoty z URI (jmeno, ...) parametrom funkcie, ktorá daný request spracuje. Potom sa zistí pomocou SQL queries id užívateľa a skupiny a obe sa ako pár predaju funkcii, ktorá ich pridá do príslušnej tabuľky.

```
@app.route('/joinGroup/meno/<string:jmeno>/pass/<string:passw>/')
def join_group(jmeno,passw):
    try:
        cur = mysql.connection.cursor()
        # get id of a user
        cur.execute("SELECT id FROM groups WHERE
                            pass = %s", (passw,))
        rv = cur.fetchall()
        group_id = rv[0][0]
        cur.execute("SELECT id FROM user_table WHERE
                            name = %s", (jmeno,))
        rv2 = cur.fetchall()
        user_id = rv2[0][0]
        cur.close()
        add_user_to_group_connection(user_id,group_id)
        return jsonify("added","succes")
    except Exception as e:
        return jsonify({'error': str(e)})
```

4.5.3 My_SQL_comunication.cs

Tento súbor je súčasťou `Time` namespace. A obsahuje len jednu triedu a to `My_SQL_comunication`, ktorá má všetky potrebné metódy na posielanie http requestov na server na ktorom beží Flask aplikácia a ona spracuje konkrétne requesty. Jej všetky metódy sú asynchrónne aby sa zaručila správna funkcionalita programu.

Táto trieda obsahuje nasledujúce metódy:

Program 28 Táto metóda je čo sa týka komunikácie najdôležitejšia, keďže väčšina ostatných ju volá aby dostali odpoveď. Metóda začne vytvorením HttpClientu a potom pošle request, na ktorý asynchrónne čaká a potom odpoveď prevedie na list stringov.

```
internal async Task<string[]> getResponceBaseOnUrl(string url)
{
  using (HttpClient client = new HttpClient())
  {
    // Send a GET request to the specified Uri
    HttpResponseMessage response = await client.GetAsync(url);
    response.EnsureSuccessStatusCode();

    // Read and output the response body
    string responseBody = await response.Content
                              .ReadAsStringAsync();

    string[] myArray = JsonConvert.DeserializeObject
                              <string[]>(responseBody);
    return myArray;
  }
}
```

Program 29 Táto metóda taktiež dostáva ako parameter url, ale jej úlohou je prísť na to či neexistuje v špecifickej databáze prvok (riadok) s rovnakými hodnotami, takže zaručuje, že v databáze nebudú 2 rovnaké hodnoty naraz.

```
internal async Task<bool> boolTestIfIs_NOT_InDatabase(string url)
{
  using (HttpClient client = new HttpClient())
  {
    // Send a GET request to the specified Uri
    HttpResponseMessage response = await client.GetAsync(url);
    response.EnsureSuccessStatusCode();

    // Read and output the response body
    string responseBody = await response.Content
                              .ReadAsStringAsync();

    string[] myArray = JsonConvert.DeserializeObject
                                <string[]>(responseBody);
    Debug.WriteLine(responseBody);

    if (myArray[1] != "y")
    {
      return false;
    }
    else
    {
      return true;
    }
  }
}
```

Ako príklad použitia API a volania metódy GetResponseBasedOnURL uvedieme metódu join group.

Program 30 join_group: funkciou tejto metódy je pridať užívateľa do skupiny. Metóda pozostáva väčšinou z kontrol a upozornení, ktoré sa zobrazia užívateľovi. Po vytvorení url zavoláme už vopred spomínanú `getResponceBaseOnUrl()` a taktiež na základe odpovede oboznámime užívateľa o výsledku.

```
internal async Task<bool> join_group(string username,string pass)
{
  if (pass=="")
  {
    string message = "Joining failed \nInvalid fields";
    string title = "Error";
    MessageBox.Show(message, title);
    return false;
  }
  string url = "http://195.113.21.131:5580/joinGroup/meno/"
                              + username + "/pass/" + pass;

  string[] myArray = await getResponceBaseOnUrl(url);

  if (myArray[1] != "succes")
  {
    string message = "Joining failed";
    string title = "Error";
    MessageBox.Show(message, title);
    return false;
  }
  else
  {
    string message = "Joining succesful";
    string title = "Welcome";
    MessageBox.Show(message, title);
    return true;
  }
}
```

5 Testovanie a získané dáta

V prvej sekcii tejto kapitoly si predstavíme spôsob testovania aplikácie a otázky formulára. V nasledujúcej sa pozrieme na odpovede užívateľov a ich skúsenosť s aplikáciou.

Aplikácia bola otestovaná na 5 dobrovoľníkoch (študentoch), ktorí ju používali po dobu jedného týždňa. Potom ich výsledky porovnali s ich prácou bez nej, kedy si mali značiť počet a náročnosť splnených aktivít na čas. Aktivity v tom obdoby boli podobne náročne ako pri používaní aplikácie a denné aktivity boli rovnaké v rovnakých časových intervaloch.

5.1 Ako sme testovali aplikáciu

Traja užívatelia si najskôr otestovali aplikáciu a potom si skúsili všetko riadiť sami. Zvyšní dvaja to mali opačne najskôr boli usmernení si rozvrh vytvoriť sami a riadiť sa ním. A až v druhom týždni sa riadili aplikáciou. Čím chceme ukázať, že výsledky sú nezávislé na tom, či mali užívatelia predstavu ako má rozvrh vyzerať. Presnejšie užívatelia č. 1, 2 a 4 mali najskôr aplikáciu a užívatelia č. 3 a 5 pracovali najskôr samostatne.

Oba tímy boli na začiatku oboznámené s cieľom a funkcionalitou aplikácie. Taktiež si ju skúsili použiť prvý krát pod mojím dohľadom. Keďže testovanie nebolo zamerané na prácu s skupinami nemali k dispozícií daný modul, jedným z dôvodov bolo i to, že užívatelia sa navzájom nepoznali. Taktiež boli usmernený aby si vyhradili každý deň čas na obed. A aby do denných aktivít, kde musia cestovať zahrnuli i čas.

Na získanie výsledkov bol použitý formulár a na konci i diskusia aký mali pocit z aplikácie a či mali nejaké pozorovania.

Otázky na formulári:

1. **Zvýšila táto aplikácia podľa vášho vašu produktivitu?**
   Na škále od 1 (Vôbec) do 10 (Maximálne).

2. **Dosiahli ste ciele, ktoré ste si stanovili?**
   Na škále od 1 (Vôbec) do 10 (Maximálne).

3. **Stretli ste sa pri používaní aplikácie s bugmi?**
   Možnosti: Áno/Nie

4. **Ak áno s akými?**
   Ak v predchádzajúcej otázke odpovedali áno.

5. **Ako hodnotíte výber metód tvorby rozvrhu?**
   Možnosti: veľmi dobrý, dobrý, neutrálny, zlý, veľmi zlý.

6. **Ako veľmi ste využívali Pomodoro techniku?**
   Na škále od 1 (Vôbec) do 5 (Stále).

7. **Prišla vám aplikácia užitočná?**
   Na škále od 1 (Vôbec) do 10 (Veľmi).

8. **Plánuje aplikáciu používať naďalej?**
   Možnosti: Áno/Nie

Taktiež mi potom zaslali snímku obrazovky ich rozvrhu a poznámky k tomu ako sa im darilo počas týždňa kedy pracovali bez aplikácie.

5.2 Výsledky formuláru

Opäť si prejdeme cez otázky u každej uvediem priemerný výsledok a po prípade komentár, ktorý som k nej postal.

1. **Zvýšila táto aplikácia podľa vášho vašu produktivitu?**
   Priemerný výsledok: 7 (7,8,6,7,7).

2. **Dosiahli ste ciele, ktoré ste si stanovili?**
   Priemerný výsledok: 9.2 (9,9,8,10,10).

3. **Stretli ste sa pri používaní aplikácie s bugmi?**
   Výsledok: 4x - Nie, 1x - Áno.

4. **Ak áno s akými?**
   "pri tvorbe rozvrhu ak si zvolím začiatočný čas v minútach iný ako násobok 5 sa začiatočné časy zle zobrazujú"

   Tento bug je už vyriešený bola to chyba v kóde pri prepočte z formátu hodiny:minúty (6:57) na hodnotu v hodinách (6.93).

5. **Ako hodnotíte výber metód tvorby rozvrhu?**
   Výsledok: dobrý, Komentár: od dvoch užívateľov som dostal k tomuto poznámku, že nemali predošlé skúsenosti s metódami riadenia času.

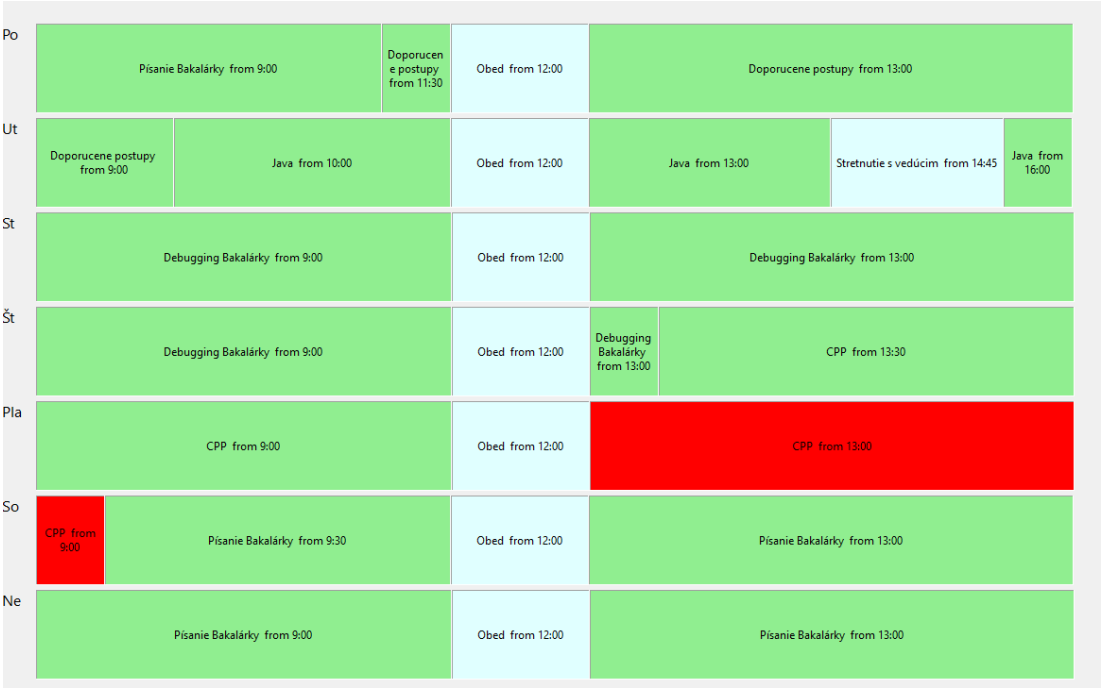
6. **Ako veľmi ste využívali Pomodoro techniku?**
   Priemerný výsledok: 8 (8,8,6,8,10).

7. **Prišla vám aplikácia užitočná?**
   Priemerný výsledok: 7 (8,7,7,7,6).

8. **Plánuje aplikáciu používať naďalej?**
   Výsledok: 2x - Nie, 3x - Áno.

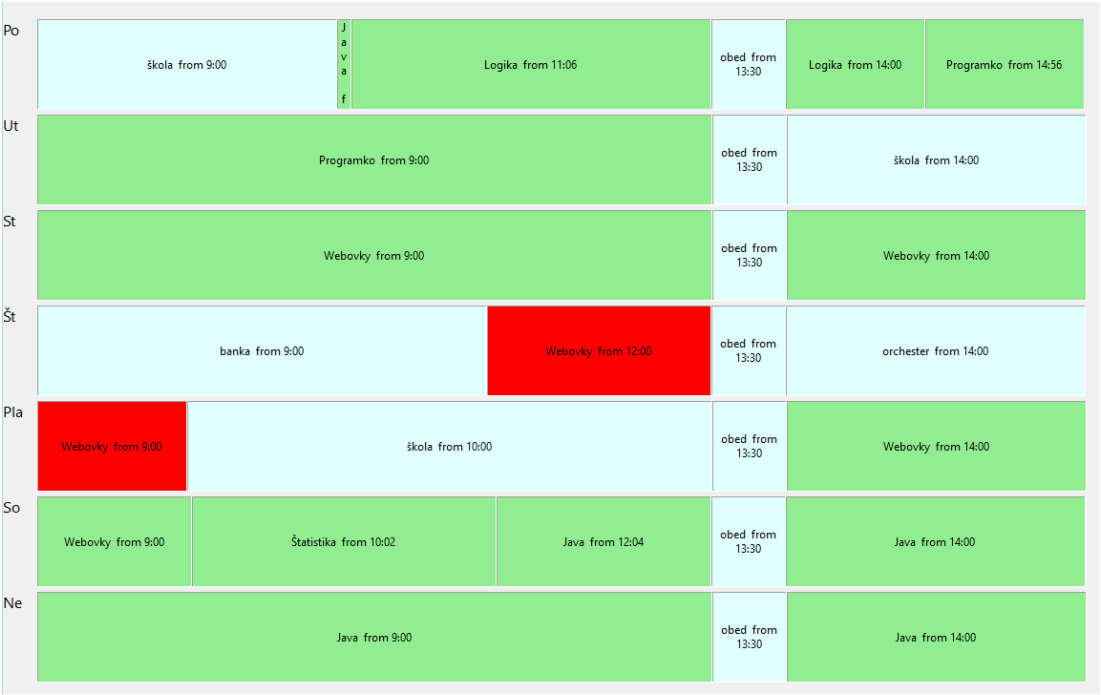
5.3 Výsledky užívateľov

Ako prvé sa pozrieme na finálne rozvrhy, ktoré si vytvorili užívatelia. Môžme pozorovať rozvrhy dvoch druhov užívatelia č. 1 a 3 testovali aplikáciu počas prázdnin a užívatelia č. 2, 4, 5 testovali aplikáciu počas semestra.

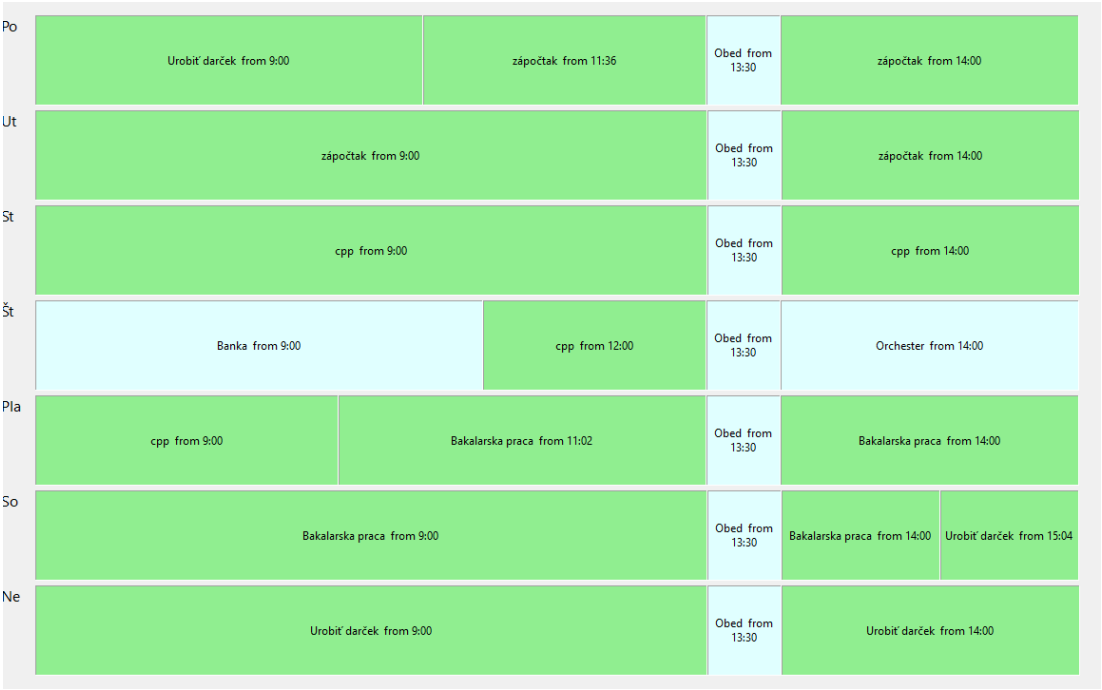


Obrázek 5.1 Rozvrh užívatelia č. 1, počas prázdnin

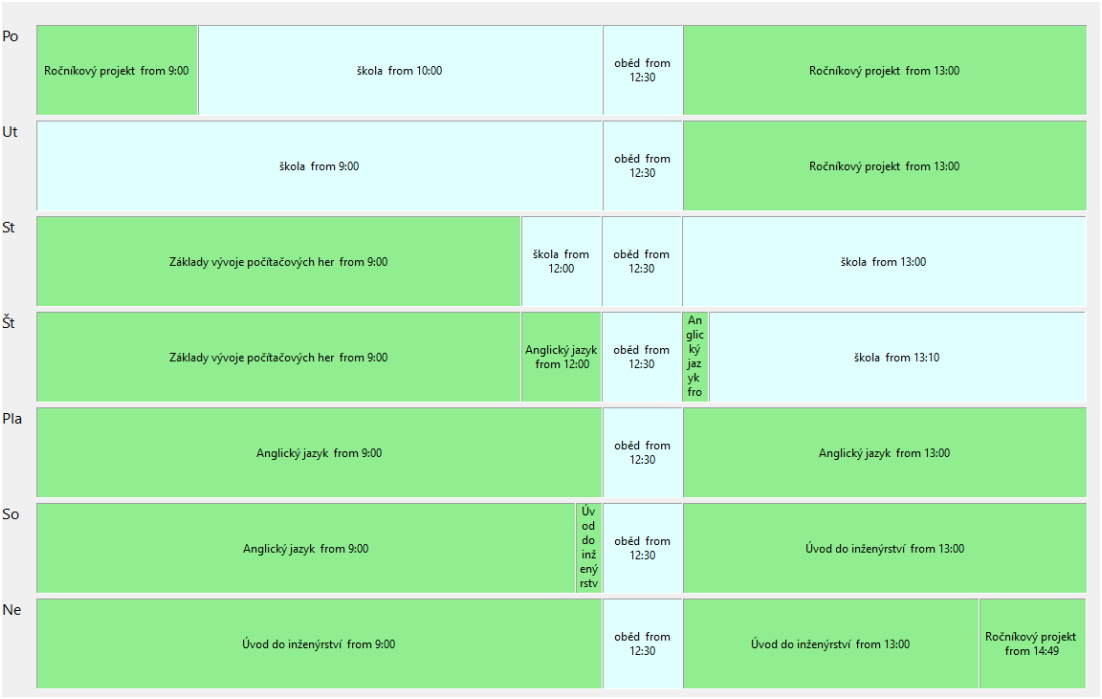
Pre pripomenutie: Po úspešnom dokončení aktivity sa aktivita zobrazí v rozvrhu nazeleno ak užívateľ aktivity nespravil tak je červená. Pokiaľ na aktivitu ešte neprišiel čas alebo je užívateľ akurát v procese jej plnenia je svetlomodrá. Jedinou výnimkou sú Denné aktivity tie sú vždy svetlomodré (napr. fitko).



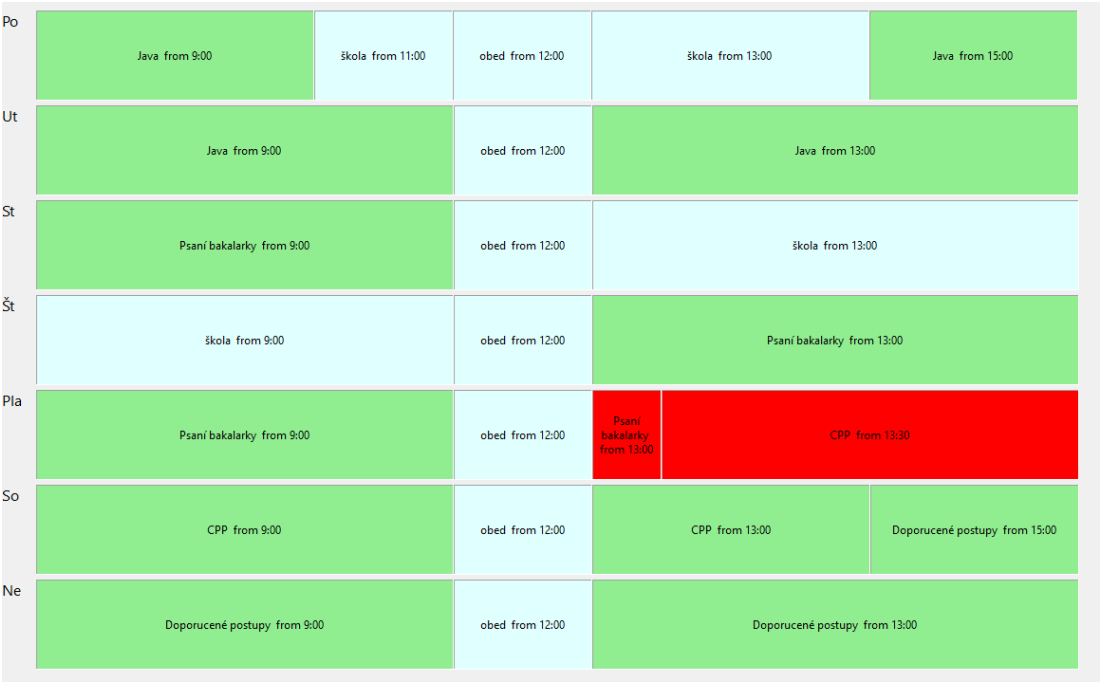
Obrázek 5.2 Rozvrh užívatelia č. 2, počas semestra



Obrázek 5.3 Rozvrh užívatelia č. 3, počas prázdnin



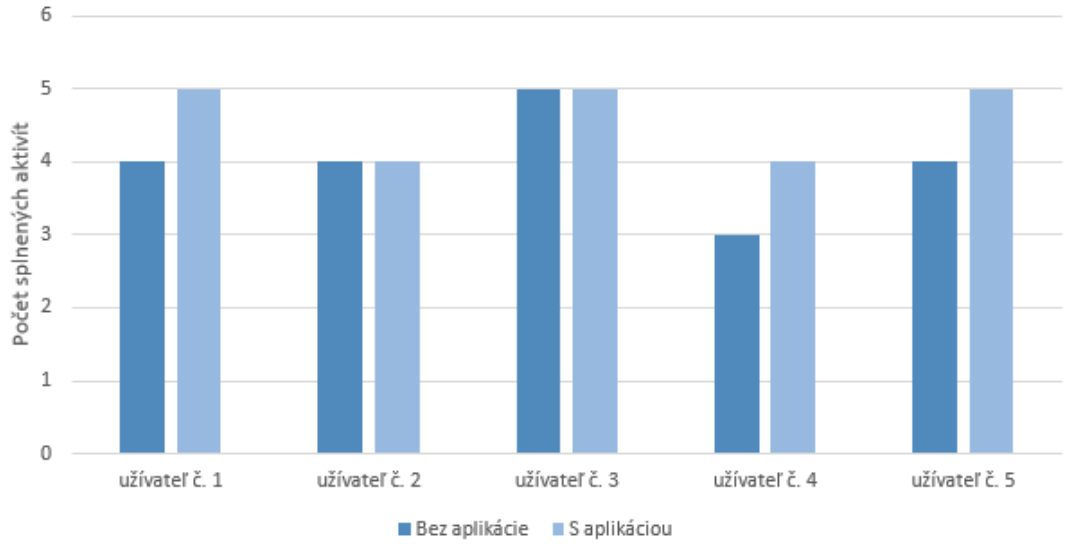
Obrázek 5.4 Rozvrh užívatelia č. 4, počas semestra



Obrázek 5.5 Rozvrh užívatelia č. 5, počas semestra

5.4 Porovnanie výsledkov užívateľov bez aplikácie a počas používania aplikácie

V tejto sekcii si ukážeme koľko aktivít zvládli užívatelia počas týždňa. Typ aktivity je napríklad `Java` z rozvrhu užívateľa č. 1 viz. Obrázek 5.1. Je to aktivita, ktorá sa v rozvrhu opakuje viac krát rozdeľujú ju len denné aktivity. Tieto aktivity boli inštruovaní dokumentovať i počas práce bez aplikácie. Toto je porovnanie počtu splnených aktivít počas používania aplikácia a bez jej použitia.



Obrázek 5.6 Užívatelia a pohľad na ich dosiahnuté ciele s a bez použitia aplikácie. Užívatelia č. 1, 2 a 4 mali najskôr aplikáciu a užívatelia č. 3 a 5 pracovali najskôr samostatne.

5.5 Diskusia s užívateľmi

Diskusia s užívateľmi a iné pozorovania: Na koniec som sa stretol s každým užívateľom a porozprával som sa o mojom projekte. Väčšina komentárov sa opakovalo nezávisle na tom či boli pozitívne alebo negatívne.

Najčastejší komentár bol, že vďaka rozvrhu nestrácali čas na nepotrebných aktivitách a prokrastinovali menej ako zvyčajne, keďže mali vždy aktivitu v rozvrhu, na ktorej majú robiť. Na druhú stranu nevýhodou aplikácie je, že je v tomto ohľade neflexibilná a ak sa náhodou objaví neočakávaná aktivita tak sa rozvrh nedá prispôsobiť alebo mu nejak naznačiť, že dôvod nezvládnutia aktivity bol podstatný a nedalo sa mu vyhnúť. To je ale funkcionalita, ktorá bola takto zamýšľaná, aby užívateľ nevedel podvádzať a potom sa chváliť svojimi výsledkami.

Často spomínaným plusom aplikácie je, že sa nemusia zamýšľať nad tvorom rozvrhu každý deň. I keď z ich skúseností tvorba rozvrhu nefungovala tak, ako som očakával. Pri vytváraní rozvrhu bolo treba prejsť viacerými iteráciami. Keď si užívateľ vytvorí prvý rozvrh nie vždy vie usúdiť správnu dĺžku alebo prioritu aktivity. Ak si zvolí, že všetky aktivity majú najväčšiu alebo 2. najväčšiu prioritu tak v podstate stratí jednu celú časť algoritmu. Iterácia tvorenia rozvrhov je dôležitá aby užívateľ pochopil ako sa rozvrh tvorí a musí to opakovať pokiaľ nie je

spokojný s rozvrhom ale akonáhle si vytvorí taký, že je s ním spokojný môže ho používať celý týždeň. Po dlhšom používaní aplikácie si vie užívateľ vybudovať cit pre tvorenie rozvrhu a vie vytvoriť dobrý rozvrh na prvý pokus.

5.6 Pohľad spätne na ciele

Síce sme aplikáciu testovali len 5 užívateľov, dalo nám to uspokojivé výsledky a na základe tohto obmedzeného testovania sa zdá, že aplikácia má pozitívny vplyv na produktivitu a organizáciu času svojich používateľov. Naznačuje to, že by bol potrebný ďalší výskum a testovania na viacerých užívateľoch.

Pozitívne ohlasy súviseli najmä s vytvorením jasného rozvrhu a s minimálnym množstvom času stráveným pri nutnosti užívateľa sa rozhodnúť, aké úlohy majú vykonať ďalej. Užívatelia ocenili i voľbu metód na časový manažment, ktoré boli implementované. Všetky tieto prvky prispeli k lepšiemu zvládaniu ich denných úloh a k vyššej spokojnosti s plnením pracovných i osobných záväzkov.

Z technického hľadiska sa v aplikácii objavilo menšie množstvo chýb, avšak boli rýchlo identifikované a opravené. V budúcnosti by sme sa mali viac zamerať na detailné testovanie aplikácie aby sa predišlo chybám.

Z pohľadu dlhodobého užívania aplikácie, z výsledkov formulára vyplýva, že väčšina užívateľov plánuje aplikáciu používať naďalej, čo ukazuje ich spokojnosť s funkcionalitou.

Na základe týchto zistení môžeme povedať, že na aplikácií sa malo pracovať naďalej s cieľom zvýšiť adaptabilitu a užívateľskú prívetivosť. Taktiež by bolo prospešné rozšíriť testovanie na väčšiu skupinu užívateľov, aby sa získali ďalšie údaje a perspektívy, ktoré by pomohli vylepšiť aplikáciu. Hlavne na skupine, ktorá sa pozná aby sa dalo pozorovať i funkcionalitu zdieľania výsledkov.

Záver

Táto práca demonštrovala praktickú aplikáciu integrácie niekoľkých techník riadenia času v aplikácií určenej na zvýšenie produktivity a efektívnosti. Začlenením metód 80/20 rule, Eisenhower Matrix, Eat That Frog!, Pomodoro Technique a i upravenú verziu Eat That Frog! ponúka vyvinutá aplikácia zmes stratégií prispôsobených tak, aby pomohla používateľom efektívnejšie riadiť svoj čas a zdieľať svoje výsledky s inými užívateľmi.

Počas tejto práce bola aplikácia podrobená testovaniu, ktoré poskytlo cenné poznatky o jej použiteľnosti a účinnosti. Síce aplikáciu testovalo len 5 užívateľov, ale sú dostatočne sľubné aby sa dalo tvrdiť, že by bol potrebný ďalší výskum. Zozbieraná spätná väzba naznačuje, že softvér zlepšuje schopnosť užívateľov zamerať sa na konkrétne aktivity a i schopnosť užívateľov plniť aktivity načas a efektívne.

Budúci vývoj by mohol zahŕňať vylepšenie algoritmu, ktorý by umožnil dynamickejšie možnosti plánovania a integráciu techník strojového učenia na predpovedanie preferencií používateľov a automatické navrhovanie priorít úloh. Tiež je potrebné zabezpečiť celé API, keďže nie je nijak zabezpečené. Navyše, rozšírenie funkčnosti aplikácie o mobilné platformy by mohlo výrazne zvýšiť jej dostupnosť a užitočnosť.

Literatura

1. Tracy, B. *Eat That Frog!: 21 Great Ways to Stop Procrastinating and Get More Done in Less Time*. Berrett-Koehler Publishers, 2017. isbn 978-1626569416.
2. Clear, J. *Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones*. Avery, 2018. First Edition. isbn 978-0735211292.
3. Koch, R. The 80/20 Principle: The Secret to Achieving More with Less. 1999. isbn 978-0385491747.
4. Cirillo, F. The Pomodoro Technique: The Acclaimed Time-Management System That Has Transformed How We Work. 2018. isbn 978-1524760700.
5. Galinsky, A.; Schweitzer, M. Friend & Foe: When to Cooperate, When to Compete, and How to Succeed at Both. 2015. isbn 978-0307720214.

A Přílohy

A.1 Elektronická príloha

Ako jediná príloha je **.zip** súbor, v ktorom sú nasledujúce priečinky: Time, app_Release, serverBackEnd a buildingTimeTable.

Štrukúru Time a buildingTimeTable sme už v práci spomínali. V priečinku serverBackEnd je zdrojový kód pre Flask a v app_Release je finála spustiteľná aplikácia.