DOKUMENTACE PROJEKT PARALIZACE

---

1. Základní informace o projektu

Název projektu:
Paralelní zpracování CSV souborů a generování reportu

Autor:
Kryštof Málek

Kontaktní údaje:
logickrysbat@gmail.com

Škola:
SPŠE Ječná

Datum vypracování:
24.11.2025

Typ dokumentu:
Dokumentace k školnímu projektu v rámci předmětu PV.
Projekt demonstruje práci s paralelními procesy/vlákny, synchronizací a thread-safe zpracováním dat.

2. Zadání a požadavky

2.1 Funkční požadavky (Functional Requirements)

1. Aplikace načítá všechny CSV soubory ze složky `data/input`.
2. Každý soubor je zpracován pomocí paralelních vláken (worker pool).
3. Aplikace pro každý soubor určí:
    - počet řádků
    - čas zpracování
    - informace o chybách

4. Aplikace agreguje souhrnné statistiky:
    - celkový počet souborů
    - celkový počet řádků
    - průměrný počet řádků na soubor
    - nejmenší a největší soubor
5. Aplikace generuje HTML report do `data/output/report.html`.
6. Aplikace zapisuje log do `logs/app.log`.

2.2 Nefunkční požadavky (Non-Functional Requirements)

1. Aplikace musí běžet na školních PC bez potřeby instalace IDE.
2. Zpracování musí probíhat paralelně s využitím více vláken.
3. Aplikace musí být stabilní při chybách.
4. Zpracování souborů musí být thread-safe.
5. Rozhraní mezi komponentami musí být jasné a přehledné.
6. Kód musí být čitelný a modulární.
7. Aplikace musí umožnit úpravu konfigurace (počtu workerů, cesty).

3. Specifikace uživatele a Use Case

Uživatel potřebuje získat přehled nad větším množstvím CSV souborů (například interní exporty, data ze systému, senzory apod.).
Cílem je rychle získat souhrnné statistiky o datech bez potřeby manuálního zpracování.

Use Case:

Název: Zpracování CSV souborů
Aktéři: Uživatel (student, učitel, pracovník)
Popis:

Uživatel vloží CSV soubory do složky `data/input` a spustí program. Aplikace automaticky zpracuje všechny soubory paralelně, vytvoří agregovaný report a uloží jej do výstupní složky.

4. Architektura aplikace

Aplikace je rozdělena do samostatných modulů:

Loader

- Prochází složku `data/input`.
- Vkládá cesty k souborům do fronty `Queue`.

WorkerPool

- Spravuje skupinu worker threadů.
- Každý worker:
    - čte frontu úkolů,
    - zpracuje CSV soubor,
    - měří čas zpracování,
    - předá výsledek agregátoru.

Parser

- Jednoduchý CSV parser (počítá řádky).

Aggregator

- Thread-safe ukládání výsledků:
    - počty řádků,
    - počty souborů,
    - statistiky,
    - chyby.

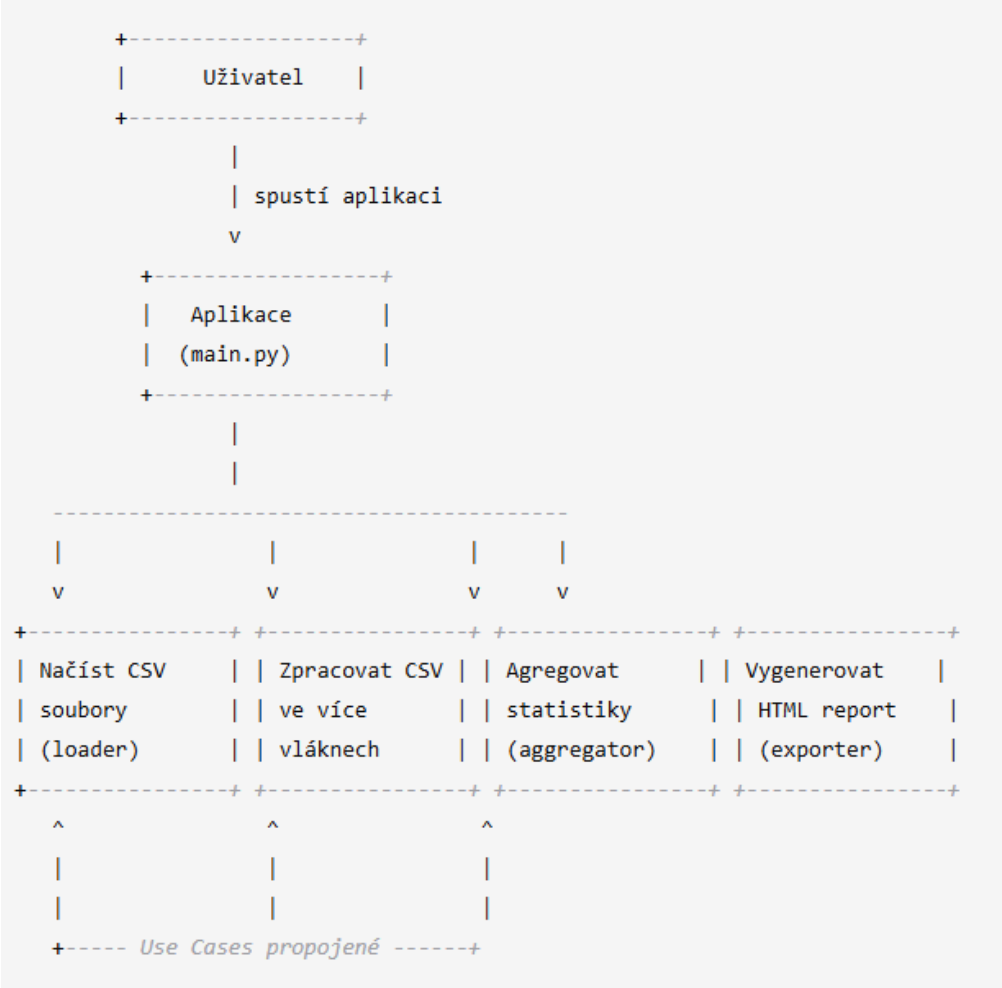
Exporter

- Vygeneruje HTML report obsahující:
    - souhrny,
    - čas zpracování,
    - seznam souborů,
    - chyby.

---

5. UML diagramy

5.1 Class Diagram (zjednodušený)



5.2 Activity Diagram – průběh aplikace

```
Start
  ↓
Načíst konfiguraci
  ↓
Inicializovat WorkerPool
  ↓
Loader vloží soubory do fronty
  ↓
Workeři paralelně zpracovávají soubory
  ↓
Aggregator ukládá data
  ↓
Po dokončení WorkerPool skončí
  ↓
Exporter vygeneruje report
  ↓
Konec
```

6. Běh aplikace (popis)

1. Aplikace se spustí v `main.py`.
2. Načte konfiguraci (INPUT_DIR, OUTPUT_DIR, NUM_WORKERS).
3. Inicializuje se agregátor, parser a worker pool.
4. WorkerPool spustí všechna vlákna.
5. Loader přidá všechny CSV soubory do fronty `file_queue`.
6. Workeři paralelně:
    - vytahují soubory z fronty,
    - měří čas,
    - počítají řádky,

    - ukládají výsledek do agregátoru.
7. Po ukončení práce WorkerPool odešle STOP signály.
8. Exporter vygeneruje výstupní report.

7. Použitá rozhraní a knihovny

Interní knihovny Pythonu:

- `threading` – práce s vlákny
- `queue` – thread-safe fronta
- `csv` – načítání CSV
- `time` – měření časů
- `os` – práce se souborovým systémem
- `logging` – logování

Žádné třetí strany – projekt je 100% kompatibilní se školními PC.

8. Konfigurace programu

V souboru `config.py`:

- `INPUT_DIR` – vstupní složka s CSV
- `OUTPUT_DIR` – kam se uloží report
- `LOG_DIR` – logování
- `NUM_WORKERS` – počet worker threadů

Uživatel může libovolně měnit bez zásahu do logiky aplikace.

9. Instalace a spuštění

1. Nainstalujte Python 3.10 nebo vyšší.
2. Ujistěte se, že složka `data/input` obsahuje CSV soubory.
3. Spusťte aplikaci:

```
python src/main.py
```

Report se vygeneruje do:

`data/output/report.html`

10. Chybové stavy

Možné chyby:

- soubor není CSV → ignoruje se
- CSV soubor je poškozený → uloží se do `Aggregator.errors`
- prázdný soubor → uloží se jako chyba
- složka `data/input` je prázdná → report bude prázdný

Způsoby řešení:

- chyby se uloží do reportu
- program nikdy nespadne (try/except)
- log obsahuje detaily

11. Testování a validace

Ve složce `test/` jsou testy pokrývající:

- CSV parser
- agregátor
- worker pool

Testy ověřují:

- správné počítání řádků
- správnou agregaci statistik
- správné dokončení workerů
- správné zpracování fronty

Aplikace splňuje požadavky zadání:

- zpracování více souborů
- paralelizace
- synchronizace
- správná generace výsledků
- stabilita při chybách

12. Verzování a známé problémy

Verzování:

Projekt je verzován pomocí Git, historie obsahuje:

- vytvoření struktury,
- implementace základní verze,
- přidání worker poolu,
- měření času,

- rozšíření reportu,
- testy,
- dokumentace.

Známé problémy:

- program nevaliduje datové typy v CSV (bere řádky jako plain text)
- nepodporuje extrémně velké CSV (100 000+ řádků)
- nepodporuje nested data nebo jiné formáty

13. Import / Export schéma

Import:

- vstupní formát: `.csv`
- povinné položky: libovolné, parser počítá řádky
- defaultní encoding: UTF-8

Export:

- HTML soubor
- obsahuje:
    - statistiky
    - informace o chybách
    - seznam souborů

14. Závěr

Projekt splňuje všechny požadavky zadání:

- paralelizace
- synchronizace vláken
- použití thread-safe struktur
- logování
- modulární architektura
- dokumentace
- testy
- reálné použití

Aplikace je vhodná pro prezentaci v rámci předmětu PV a pro rozšiřování.