Дизајн и архитектура на софтвер

Домашна работа 2

1. Концептуална архитектура

Концептуалната архитектура всушност го претставува високо-нивовското функционирање на апликацијата. Нашата апликација користи **Pipe-and-Filter архитектура** и ги вклучува следниве компоненти:

- **Влезни податоци:**
  - Веб-страницата на Македонската берза претставува изворот од каде што се преземаат сите податоци за компаниите.
- **Филтри:**
  - **fil1:** Ги филтрира листите на имиња на компании од HTML-структурата на веб-страницата.
  - **fil2:** Додава информации за последниот датум на ажурирање за секоја компанија.
  - **fil3:** Ги скрапира новите податоци за компаниите и ја комбинира постоечката и ново добиената информација.
- **Обработка на податоци:**
  - Главната функција (data_processing) ги повикува филтрите еден по еден, трансформирајќи ги податоците чекор по чекор.
- **Резултат:**
  - Податоците се зачувуваат во CSV датотеки за секоја компанија.

Цел: Целта на концептуалната архитектура е да се прикаже како различните компоненти соработуваат за да се обработат податоците. Ова ќе го преставиме со помош на дијаграм на податочен проток (DFD). И овој тип на дијаграм се користи за да се прикаже како податоците течат низ различни делови од системот и кои процеси ги обработуваат.

Елементи на дијаграмот:

1. **Влезни податоци:**
   - Претставено со правоаголник. Во овој случај, „Веб-страница“ е изворот на податоците (влезната точка).
2. **Процеси (Филтри):**
   - Претставени со правоаголници со заоблени агли или стандарден правоаголник.
   - Овие процеси ги обработуваат податоците:
     - fil1 — Филтрирање имиња на компании.
     - fil2 — Проверка на последни ажурирања.
     - fil3 — Преземање и спојување на податоци.
3. **Излезни податоци:**
   - Претставено со правоаголник. Во овој случај, CSV датотеки се крајниот излез

4. **Стрелки:**
    - Ги прикажуваат насоката и протокот на податоците помеѓу елементите.

Во продолжение е даден дијаграмот:

2. Извршна архитектура

Извршната архитектура го опишува начинот на извршување на апликацијата во инфраструктурата.

- **Главни компоненти:**
    - **Локален сервер или работна станица:** Кодот е наменет за извршување на Python средина.
    - **Веб-извори:** Податоците се преземаат од веб-страницата на Македонската берза.
    - **Паралелно извршување:**
        - Користи **ThreadPoolExecutor** за паралелна обработка на податоци за повеќе компании.
- **Папки и структура:**
    - **all_data/**: Чува CSV-датотеки со податоци за секоја компанија
- **Контејнеризација:**

- Апликацијата може да се контејнеризира со Docker за полесна дистрибуција и скалабилност.

Цел: Целта на извршната архитектура е да се прикаже протокот на податоци и интеракција помеѓу различните компоненти на системот.

Елементи на дијаграмот:

Овој дијаграм ги прикажува основните елементи и нивната комуникација преку следниот редослед:

- **Локална машина**:

- **Python скриптите** се извршуваат на **локалната машина**. Овие скрипти се одговорни за обработка на податоци, како и за комуникација со надворешните извори на податоци (веб-страници и локални фајлови). Python скриптите ги иницираат процесите за преземање, филтрирање и складирање на податоци.

- **Надворешна веб-страница**:

- Локалната машина се поврзува со **Надворешната веб-страница** (во овој случај, веб-страницата за Македонската берза). Оваа компонента обезбедува податоци кои се извлекуваат преку Python скриптите.
- Податоците од веб-страницата се обработуваат, филтрираат и претставуваат во потребен формат за понатамошна обработка.

- **Локален фајл систем (CSV)**:

- Како последен чекор во извршната архитектура, податоците што се обработуваат се зачувуваат во **CSV датотеки** на локалниот фајл систем. Овие датотеки можат да се користат за понатамошни анализи, чување на историјат на податоци или пренос на податоци во други делови на системот.

- **Стрелки** – покажуваат проток на податоци од еден елемент кон друг, Пример:

- Стрелка од **Python скрипти** до **Надворешна веб страница** покажува дека скриптите ги преземаат податоците од веб страницата
- Стрелка од Надворешна веб страница до Локален фајл систем (CSV) покажува дека преземените податоци се зачувуваат локално.

Во продолжение е даден дијаграмот:

3. Имплементациска архитектура

Имплементациската архитектура го опишува кодот, алатките и технологиите користени за развој.

- **Јазик:** Python.
- **Библиотеки:**
  - **pandas:** За манипулација со податоци и чување во CSV формат.
  - **BeautifulSoup:** За парсирање на HTML содржина.
  - **requests:** За преземање на податоци преку HTTP.
  - **concurrent.futures:** За паралелно извршување.
- **Модуларност:**
  - Кодот е организиран во модули (filters.py, data_processing.py), што овозможува повторна употреба и лесно одржување.
- **Фолдерска структура:**
  - **data_pipeline.py:** Главна програма.
  - **filters.py:** Функции за филтрирање.
  - **data_processing.py:** Логика за обработка на податоци.
- **Зачувување на податоци:** Податоците се зачувуваат како CSV-датотеки, што овозможува лесна анализа со други алатки.

Цел: Целта на имплементациската архитектура е да се прикаже како се имплементира системот на техничко ниво, со фокус на конкретни компоненти, модули, и нивните интеракции во рамките на апликацијата. Таа е важен дел од архитектурниот дизајн на апликацијата, бидејќи ги покажува начините на кои се спроведуваат концептуалните и извршните архитектури.

Елементи на дијаграмот:

Овој дијаграм е **UML Класен дијаграм** или **Дијаграм на имплементација** кој прикажува како различни компоненти на апликацијата (класи или модули) се поврзуваат и комуницираат меѓусебно.

- **data_pipeline.py:**

- Претставува главната компонента на апликацијата.
- Тоа е модул кој ги повикува функциите од други модули.
- Главната задача на **data_pipeline.py** е да иницира обработка на податоците преку **filters.py** и **data_processing.py**.

- **filters.py:**

- Овој модул ги содржи функциите **fil1**, **fil2**, и **fil3**.
- Неговата задача е да обезбеди филтрирање на податоците, проверка на последни ажурирања и собирање на потребни податоци.

- **data_processing.py:**

- Овој модул се грижи за обработката на податоците.
- Тоа е место каде што се врши основната логика на апликацијата за обработка и спојување на податоците од различни извори.

- **Стрелки:** стрелките помеѓу блоковите покажуваат повикување на функции или преносс на податоци помеѓу модулите.
  - **data_pipeline.py** повикува функции од **filters.py**, што значи дека тој ги иницира операциите за филтрирање на податоците.
  - **filters.py** комуницира со **data_processing.py**, пренесувајќи податоци кои треба да бидат обработени.

Во продолжение го имаме имплементацискиот дијаграм: