Домашна работа бр.2

Опис на концептуална архитектура

Концептуалната архитектура на апликацијата ги опишува основните компоненти и нивната интеракција. Компоненти:

1. Frontend: Кориснички интерфејс за интеракција со корисниците .
2. API слој: Средишен слој кој ги пренасочува барањата од фронтенд до бекенд .
3. Backend: Логика за обработка на податоци и комуникација со базата на податоци.
4. Филтри: Обработка на податоци, како што се пребарувања или анализа.
5. База на податоци: Чување на сите податоци .
6. Кориснички интерфејс: Визуелизација на податоците (графикони, табели).

Комуникацијата помеѓу компонентите се одвива преку API барања и одговори, со обработка на податоци во бекендот и нивно прикажување во frontend.

Oпис на извршна архитектура

Извршна архитектура ја опишува динамиката на системот кога е во функција.

1. Frontend: Активен компонент кој ја иницира комуникацијата со backend преку API повици кога корисникот ќе интерактира.
2. API: Сервис кој прима барања од frontend и ги пренасочува до backend.
3. Backend: Сервис кој обработува податоци, комуницира со филтрите и базата на податоци.
4. Filters (Filter1, Filter2, Filter3): Активни компоненти кои го процесираат податокот во неколку фази (собирање, чистење, анализа).
5. Database: Сервис кој чува и обезбедува пристап до податоците за frontend и backend.

Секој од овие компоненти работи асинхроно или синхроно во зависност од природата на задачите, со цел да се обезбеди ефикасно извршување и одговорност на системот.

Опис на имплементациска архитектура

Имплементациска архитектура го дефинира конкретното техничко решение и алатките што се користат за изработка на апликацијата. Во вашиот проект:

1. Frontend: Реализиран со React за динамички интерфејс и прилагодлива корисничка интеракција.
2. API Layer: Изграден со REST API, имплементиран со Flask во Python.
3. Backend: Flask управува со бизнис логиката, комуникацијата со базата и филтрите.
4. Filters: Python модули кои обработуваат податоци.
5. Database: PostgreSQL за сигурно складирање и обработка на податоци.
6. Контенјеризација: Docker се користи за пакување на сите компоненти за лесно распоредување.

Оваа архитектура обезбедува скалабилност, одржливост и флексибилност за идни ажурирања и интеграции.

Архитектонски стилови

Aпликација ќе комбинира различни архитектонски стилови.

Pipe and filters: Овие ќе се користат за обработка на податоци во апликацијата. Секој филтер ќе биде независен и ќе работи на секвенцијален начин, што значи дека еден филтер ќе ја обработи информацијата и ќе ја предаде на следниот.

* + Филтер 1 ќе прави scraping на податоците од веб сајтот на македонска берза.
  + Филтер 2 ќе ги процесира податоците и ќе ги зачува во базата на податоци.
  + Филтер 3 ќе ги анализира обработените податоци и ќе ги подготви за визуелизација на фронтенд страната.

Слоевита архитектура: Со ова ќе ги разделиме различните делови на апликацијата на слоеви, за да биде полесно да се менуваат и управуваат деловите на апликацијата:

* + Frontend: Користиме React.js.
  + Backend: Платформа за сервер-side логика, Python и Flask, која ќе управува со API повиците и ќе ги спроведува филтрите и обработката на податоците.
  + База на податоци: Систем за складирање податоци во кој користиме SQL кој ќе чува податоци за акциите, корисниците и извештаите.

Дистрибуирана архитектура со микросервиси: Секој дел од апликацијата ќе се развива како независен сервис. На пример:

* Филтерите се имплементирани како микросервиси кои ќе комуницираат преку API со бекендот.
* API слојот ќе биде основната комуникациска точка за сè, каде фронтендот и другите компоненти ќе испраќаат барања.
* Контејнеризација: Користиме Docker за да ги пакуваме различните делови на апликацијата во контеинери, што ќе овозможи лесна имплементација и скалабилност на апликацијата. Секој микросервис (филтер, бекенд, база на податоци) ќе може да се распакува во свој контенер.

Како ќе ги примените архитектонските стилови?

* Pipe-and-Filter: Овие филтри се применуваат за обработката на податоци. Секој филтер ќе работи со податоците, додека не дојде до последниот филтер кој ќе ги генерира резултатите за визуализација.
* Layered Architecture: Со слоевитата архитектура, ќе имаме одделени различни аспекти на апликацијата:
* Frontend: Користење на React.js за да создадеме динамични страници.
* Backend: Бекенд со Python и Flask кој ќе прима барања од фронтенд и ќе управува со податоци.
* Database: Складирање на податоци во база на податоци (PostgreSQL).
* Distributed Microservices: Секој филтер ќе работи како микросервис кој може да се разгори или биде повторно поставен независно од другите компоненти. Ова овозможува независна еволуција и развој на секоја компонента.
* Containerization: За секој микросервис и дел од апликацијата ќе користиме Docker контейнери. Ова ќе ни овозможи да ја поставиме апликацијата во било кој облак и да ја управуваме поедноставно.