Документација на Архитектурниот Дизајн на TuneIn

Концепцискиот, извршниот и имплементацискиот поглед

Содржина на документацијата:

Концепски поглед **1**

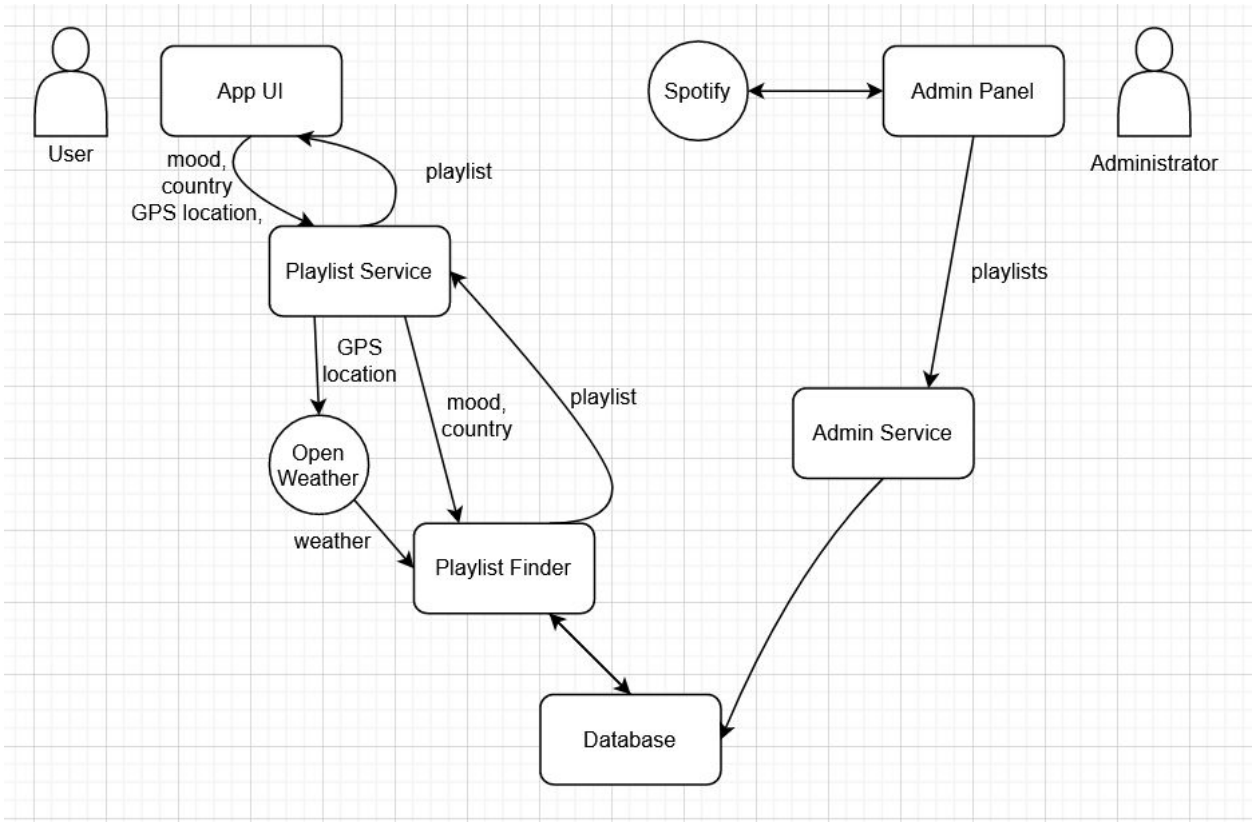
Извршен поглед **3**

Имплементациски поглед **4**

1. Концепски поглед

Концепцискиот поглед на софтверската архитектура на една апликација е првиот поглед што се дизајнира. Тој ги рефлектира желбите на стејкхолдерите. Првите итерации на овој дизајн беа посветени на функционалните барања. Тоа е основата на апликацијата. Потоа се имплементираат нефункционалните барања. Финесите на една веќе функционална апликација.

Еве го концепцискиот поглед на TuneIn апликацијата:



Компоненти и нивните одговорности:

- Admin Panel
    - Филтрирање на побараните информации (т.е. новите листи од песни)
- Admin Service
    - Додавање/ ажурирање на листи на песни за држави
    - Додавање/ ажурирање на листи на песни за нови расположенија
    - Додавање/ ажурирање на листи на песни за нови временски прогнози
- App UI
    - Избор на тип на листа на песни
    - Земање соодветни параметри(држава/ расположение /координати)
    - Прикажување на листата песни
    - Прикажување на пуштената песна во формат на music player
- Database
- External System 1 (Leaflet, Mapbox)
    - Зема координати од локацијата на корисникот
    - Враќа во која држава се наоѓа корисникот
- External System 2 (Open Weather)
    - Зема држава
    - Враќа каква е временската прогноза во државата
- External System 3 (Spotify)
    - Враќа готови листи од песни (нова држава/ расположение)

- Playlist Service
    - Насочува кориснички параметри кон следната дестинација(екстерен систем или соодветната компонента)
- Playlist Finder
    - Зема кориснички параметри
    - Соодветно на избраниот параметар, враќа листа од песни

Конекторите го означуваат текот на информации помеѓу активните компоненти. Стрелките покажуваат од изворот на информации кон нивната дестинација. Кога ќе ја достигнат дестинацијата тие или се обработуваат или се зачувуваат.

2. Извршен поглед

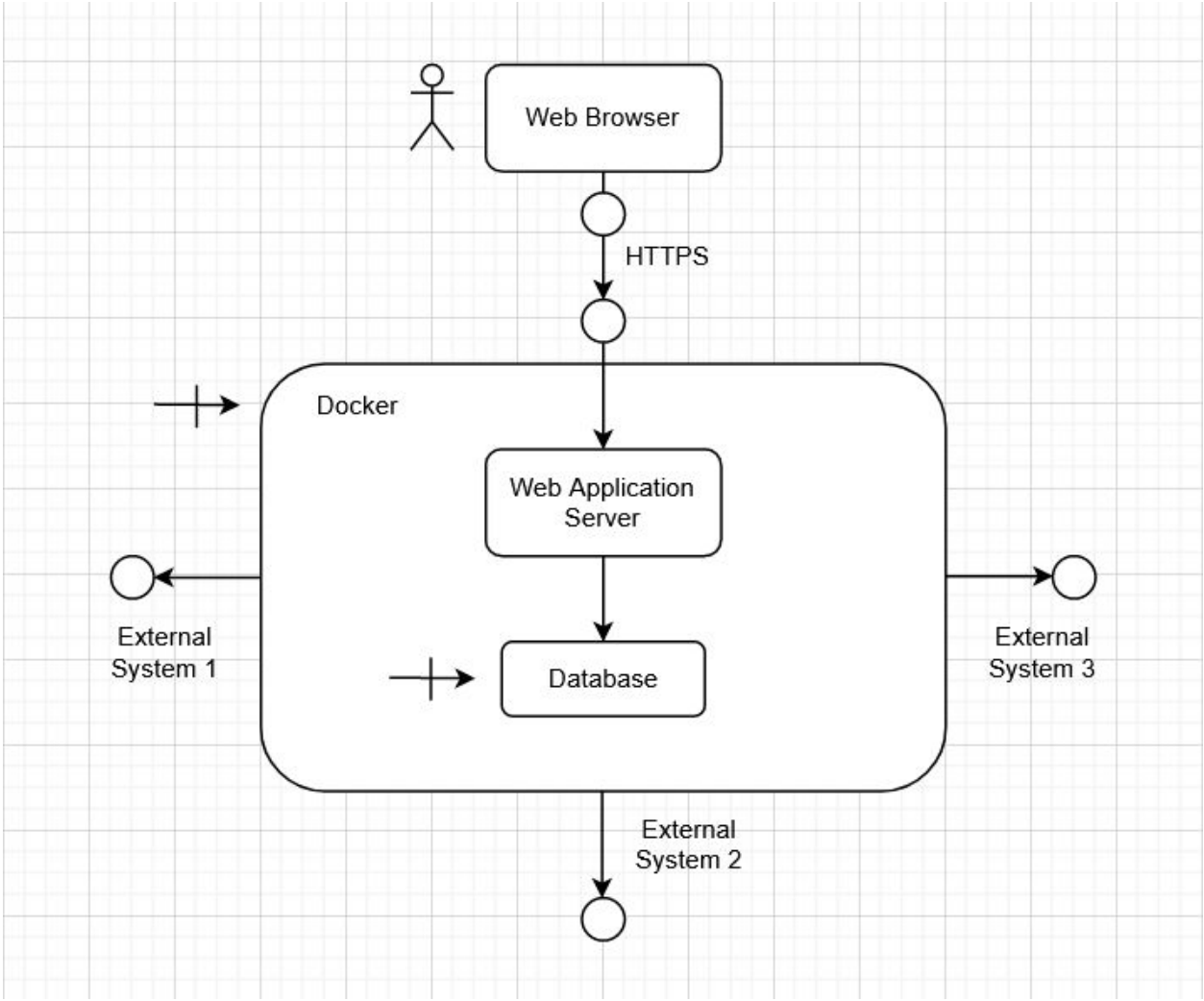
Концепцискиот поглед најмногу се фокусираше на функционалните барања. Соодветно потребен е поглед за нефункционалните барања. Тоа е извршниот поглед од архитектурата, кој се фокусира на времето на извршување. Целта е да се има што пократко време, а тоа значи брзи хардверски елементи и системи, и ефикасни процеси и нишки. Исто како и концепцискиот поглед и овде се присутни компонентите и конекторите.

Компонентите се:
- Хардверски елементи
- База на податоци
- Процеси
- Нишки

Брзината на хардверот е зависна од бројот на јадра во процесорот. Колку повеќе јадра толку повеќе паралелни процеси може да си извршуваат. Обратното е точно за процесите и нишките. Колку помалку од нив, толку подобро време. Освен бројот важна е и ефикасноста.

Конекторите означуваат повици од еден компонент кон друг. Комуникацијата може да биде синхрона, асинхрона и callback. Меѓутоа, дизајнот на нашата апликација дозволува само синхрона комуникација. Причината за ова е високото ниво на интерактивност со клиентот. Исклучително синхроната комуникација ќе го зголеми времето на извршување. Дополнително ќе се зголеми времето со повиците кон трите екстерни системи. Ова се дел од ограничувањата кои не може да се надминат, така што посветено е внимание да се оптимираат другите аспекти од апликацијата.

Ова најдобро е илустрирано на следната слика.



3. Имплементациски поглед

Концепцискиот и извршниот поглед ги покрија функционалните и нефункционалните барања. Останува да се претстави како ќе се изгради оваа апликација, а за тоа е имплементацискиот поглед. Подетално, овој поглед се фокусира на сите технички неопходни елементи (софтверски пакети, библиотеки, …) и на нефункциските барања кои не беа покриени во извршниот поглед.

Кај компонентите има поделба на апликациски и инфраструктурни. Апликациските компноненти во поголем дел се поклопуваат со концепциските компоненти. Нема потреба од повторување, па се продолжува на инфраструктурните компоненти. Тие овозможуваат

системот да работи, но не се поврзани со функционалните барања. Таков компонент во апликацијата е базата на податоци.

Вклучени се и неколку контејнер компоненти. Одлучено е да бидат повеќе со цел да нема една точка на пад. Има главен контејнер, тоа е Docker. Во него имаме уште 3 помали контејнери, за базата на податоци, за скриптите и за веб апликацискиот сервер. Откако сите контејнери ќе се поврзат и ќе завршат со нивните фунции се добива низа од 10 модели со параметри - линк, име, артист. Овој финален продукт се враќа на корисникот како листа од песни и тој може да ги слуша. Ова е претставено на илустрацијата подолу.