

Дизајн и архитектура на софтвер

Апликација за анализа на Македонската берза

Стефан Ефтимов 211085

Архитектурен дизајн

1. Вовед

Во овој дел ќе биде разгледан архитектурниот дизајн на мојата апликација која има за цел приказ на дневни и историски податоци за акциите и финансиските трендови на компаниите на македонскиот пазар. Апликацијата овозможува корисниците да добијат целосен текстуален и графички преглед за компаниите и нивните акции, пребарување, филтрирање по датум и промена на валута според курсната листа.

Апликацијата е развиена со користење на различни технологии и архитектурни стилови, кои се објаснети во продолжение. Текстот е поделен на три дела: концептуална архитектура, извршна архитектура и имплементациска архитектура. Во секој дел ќе бидат опишани главните компоненти, функционалности и врските помеѓу нив.

2. Концептуална архитектура

Компонентни одговорности

App UI

- Прикажи податоци
- Прикажи дијаграми

App Logic

- Управувај со барања од корисник
- Автентифицирај корисник

Stock Info Manager

- Обработи податоци за акции
- Доведи ги во соодветен формат

User Manager

- Регистрирај корисник
- Најави корисник
- Управувај со преференци на корисник

Admin Service

- Овозможи интерфејс преку Admin Panel
- Додади, ажурирај и избриши податоци

Analysis Modul

- Генерирај прогнози

Data Storage

- Чувај податоци за корисници
- Чувај податоци за компании
- Чувај податоци за акции

External System (Македонска берза API)

- Обезбеди податоци за акции

Клучни концепти

Податок

- База на податоци
- Акции
- Компании
- Корисници
- Цени

Функција

- Пребарај
- Филтрирај
- Најави се / Одјави се
- Регистрирај се
- Пресметај

Засегната страна

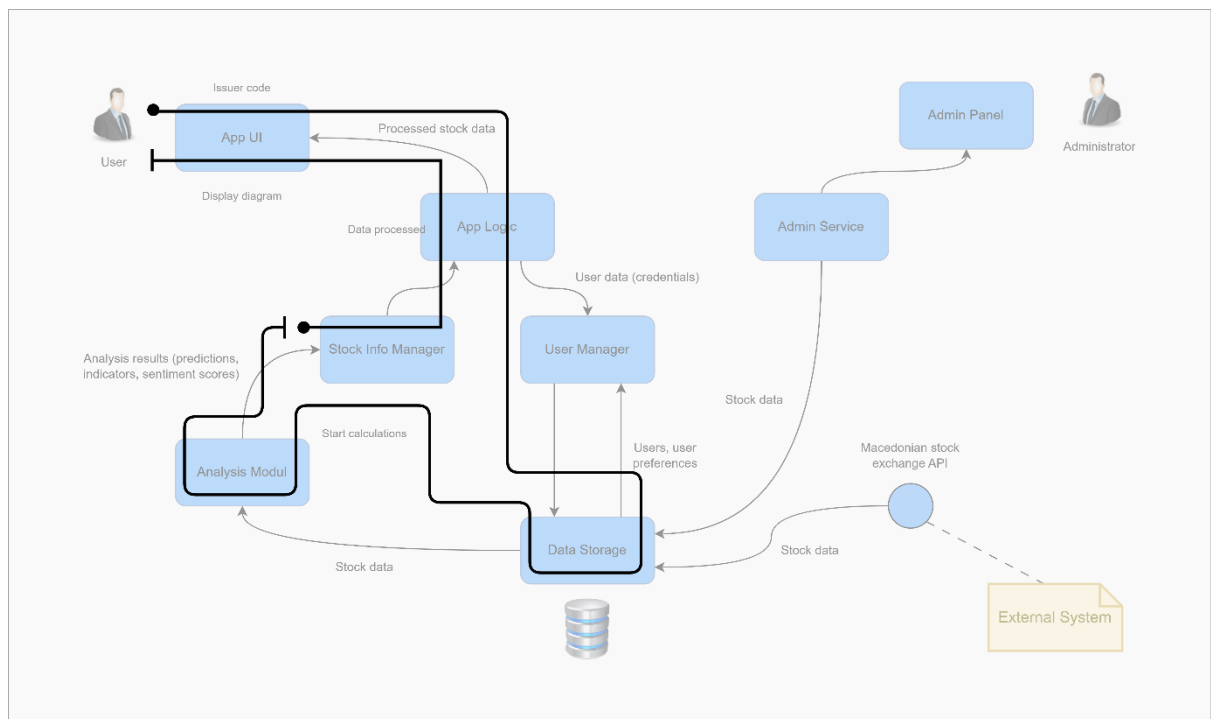
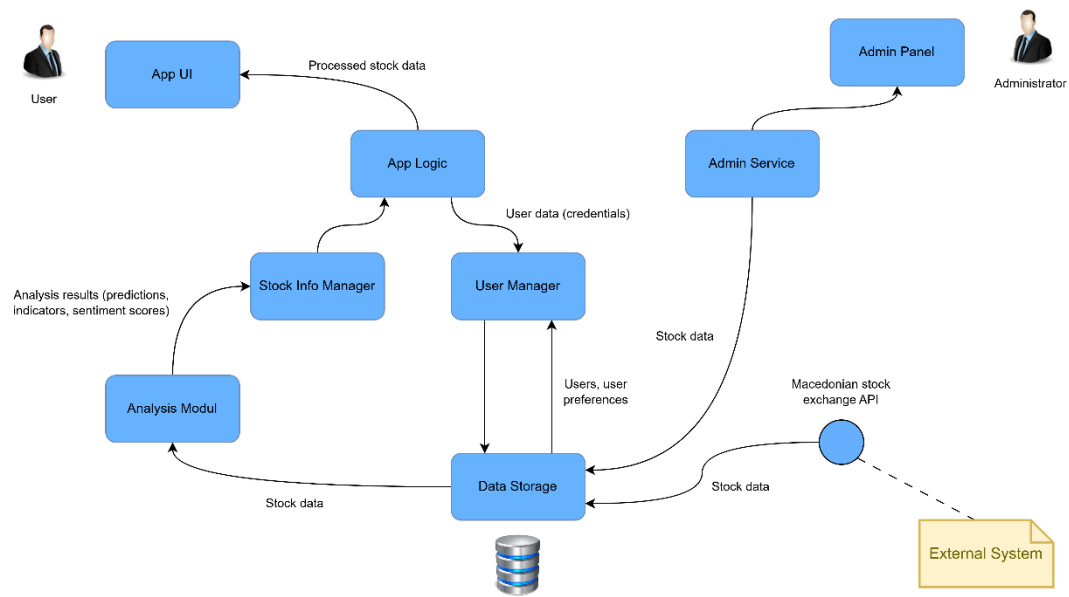
- Корисник
- Администратор

Систем

- Надворешен систем

Абстрактен концепт

- Пораст акции
- Пад акции
- Максимум
- Минимум



3. Извршна архитектура

GUI (Graphical User Interface)

Го претставува корисничкиот интерфејс преку кој корисникот комуницира со апликацијата. Составен од два модула:

- App UI: Главен модул за визуелизација и интеракција.
- Navigation UI Logic: Логика за навигација низ различни делови на апликацијата.

App Logic

Средишен дел од архитектурата што управува со главната бизнис логика на апликацијата.

- Прима барања од GUI преку HTTP.
- Комуницира со базата на податоци за зачувување или преземање информации.
- Вклучува и интеграција со Prediction Engine за пресметки и предвидувања.

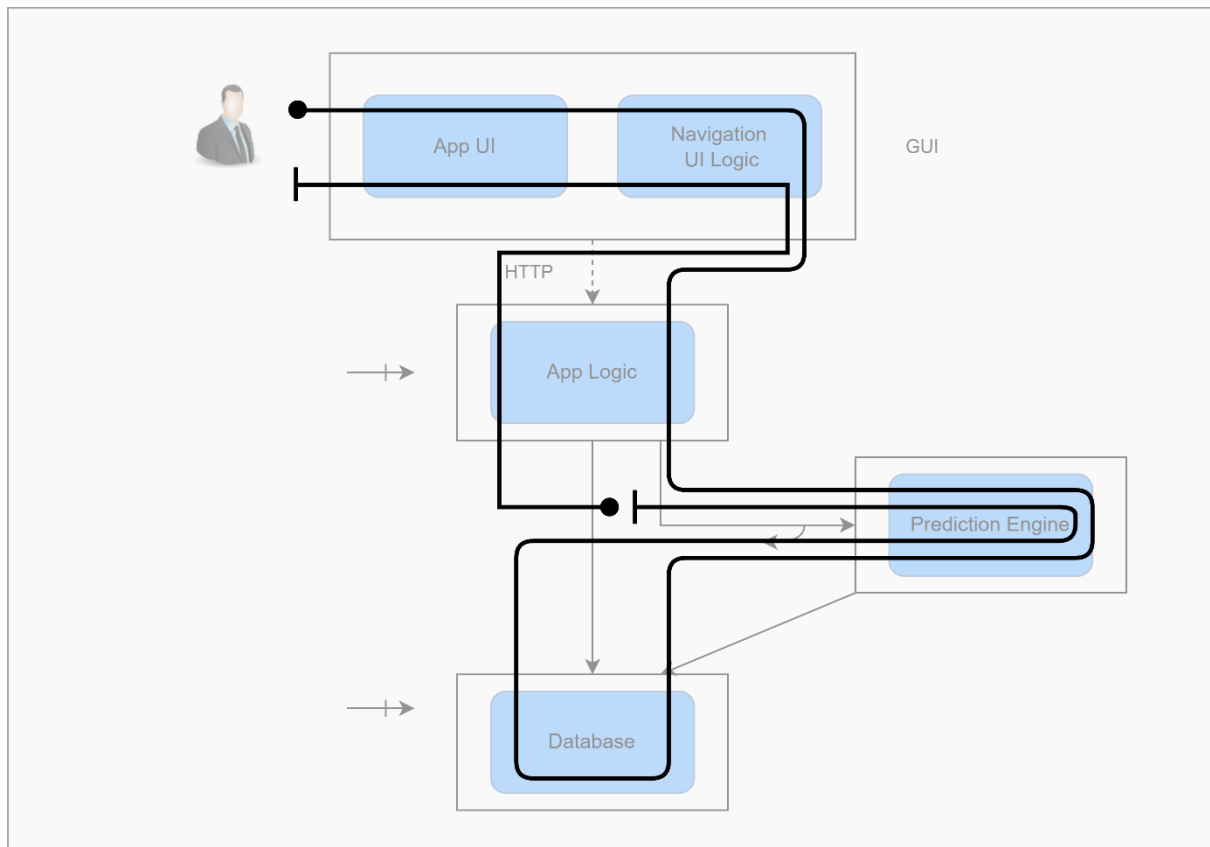
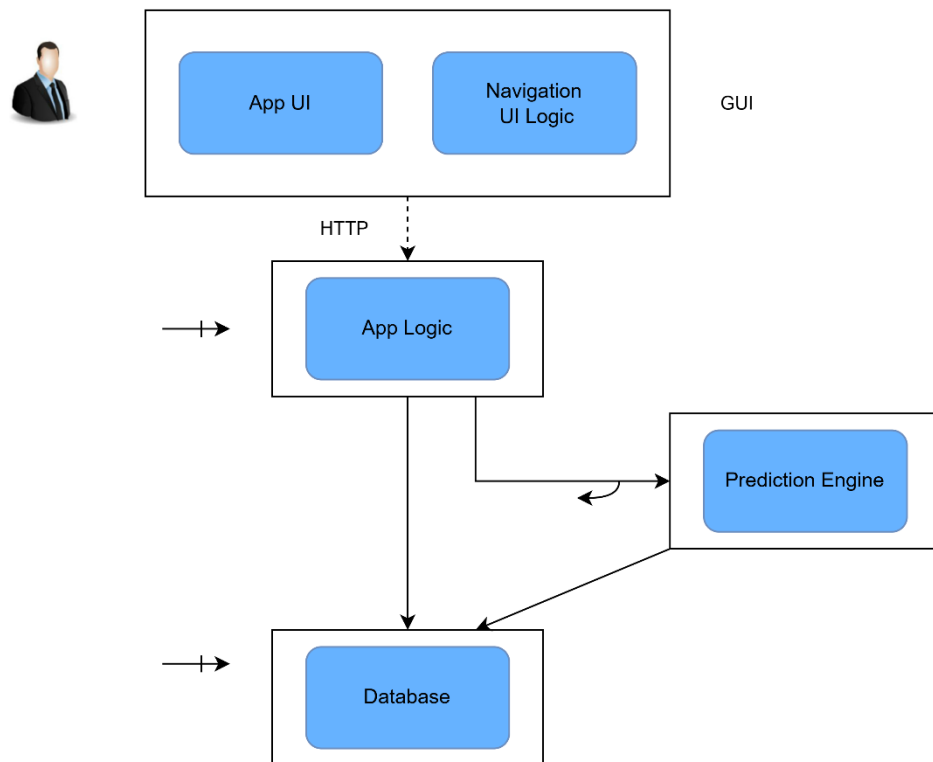
Prediction Engine

Независен модул што ја обработува логиката за пресметување и предвидувања на идните цени на акциите и сентименталните фактори за да се одреди дали акциите на одредена компанија ќе пораснат или паднат (врз основа на аналитички алгоритми, обработка на природни јазици и машинско учење).

Работи тесно поврзано со App Logic, но може да презема и податоци директно од базата.

Database

- Централна точка за складирање на податоци.
- Пристапувана од App Logic за читање или пишување информации.
- Исто така, го снабдува Prediction Engine со потребните податоци за пресметки.



4. Имплементациска архитектура

Клиент (Web Browser)

Циклусот започнува кога корисникот испраќа барање преку веб-прелистувач. Ова може да биде иницирано со клик на линк, пополнување на форма или било кое друго HTTP барање. Прелистувачот ја визуализира содржината добиена од серверот користејќи HTML, CSS и JavaScript.

Веб сервер (Apache)

Веб серверот е првата точка на контакт меѓу клиентот и апликацијата. Тој ги прима HTTP барањата од клиентите и е одговорен за следново:

Рутирање на барања – ги препраќа динамичките барања (на пр. API или веб-страници генерирани од Django) до апликацискиот сервер преку WSGI.

Опслужување статички датотеки – ги обработува статичките ресурси како што се слики, CSS и JavaScript директно, без вклучување на апликацискиот сервер.

Апликациски сервер (WSGI - Web Server Gateway Interface)

Апликацискиот сервер ја претставува врската помеѓу веб серверот и апликацијата направена во Django. Во овој контекст, WSGI е стандарден протокол за поврзување на веб серверите со Python-базирани веб апликации.

Веб апликација (Django)

Откако барањето стигнува до Django преку WSGI, започнува неговата обработка.

Рутирање на URL (URLconf)

Django ги користи шаблоните за URL дефинирани во URL конфигурацијата (URLconf) на проектот за да одреди која функција или класа ќе го обработи барањето. URL конфигурацијата ги поврзува URL адресите со view-та, со што се дефинира логиката која треба да се изврши за секој URL шаблон.

Обработка на View функција/класа

Откако ќе се идентификува соодветното view, се извршува функцијата или view-то базирано на класа. View-то ја содржи апликативната логика и е одговорено за обработка на барањето и враќање соодветен одговор.

Обработка на барањето (Middleware)

Пред да стигне до view, барањето поминува низ middleware компоненти. Middleware е начин за глобална обработка на барања пред да стигнат до view-то или

откако view-то го обработило барањето. Примери за middleware вклучуваат middleware за автентикација, безбедност и сесии.

Објект барање

View-то добива HttpRequest објект кој содржи информации за барањето, како што се хедери, метод и испратени податоци.

Интеракција со базата на податоци

Ако view-то треба да обработи податоци од база, тој ја користи ORM (Object-Relational Mapping) алатката на Django за извршување на барања, како и за добивање или менување на податоци.

Рендерирање на шаблони

Ако view-то треба да генерира HTML содржина, го користи системот за шаблони на Django за да рендерира динамична содржина во HTML шаблоните. Шаблоните можат да вклучуваат логика, променливи и тагови за генерирање динамичен излез.

Објект одговор

View-то враќа HttpResponse објект, кој содржи генерирана содржина, статусен код и хедери. Одговорот може да биде целосна HTML страница, JSON податоци или било кој друг формат, во зависност од барањата на апликацијата.

Middleware (пост-обработка)

Откако view-то го обработило барањето и генерирало одговор, одговорот поминува низ middleware компоненти за дополнителна обработка. Ова може да вклучува компресија на одговорот, додавање безбедносни хедери или други задачи за пост-обработка.

Испраќање на одговорот до корисникот

Крајниот HttpResponse се испраќа назад до прелистувачот на корисникот, со што се завршува циклусот на барање-одговор.

Овој циклус се повторува за секоја интеракција на корисникот со веб-апликацијата. Архитектурата на Django, вклучувајќи го рутирањето на URL адреси, view-та, middleware и системот за шаблони, обезбедува структуриран и ефикасен начин за обработка на барања и генерирање одговори.

