Дизајн и архитектура на софтвер
Домашна работа 4

Софтверски шаблони - Макции

Во рамките на нашиот проект решивме да го имплементираме Model View Controller (MVC) патернот. Овој документ го објаснува нашиот избор, структурата на MVC и како таа се усогласува со барањата на нашата апликација.

Преглед на MVC дизајн шемата

MVC дизајн-шемата е широко користена архитектонска шема која ја дели апликацијата на три меѓусебно поврзани компоненти:

1. **Модел (Model)**: Моделот ја претставува основната логика и податоците на апликацијата. Тој го одредува форматот на податоците. Во нашиот случај, моделот е одговорен за историските цени на акциите, издавачите и интеракциите со базата на податоци.

2. **Поглед (View)**: Погледот е одговорен за претставување на податоците на корисникот. Тој функционира како слој за кориснички интерфејс (UI). Во нашата апликација, ова ќе вклучува фронтенд изграден со React, кој ќе прикажува цени на акции, графици и предвидување, преглед на листата на зачувани издавачи за даден корисник и слично.

3. **Контролер (Controller)**: Контролерот делува како посредник помеѓу моделот и погледот. Ги обработува корисничките влезови, прави интеракција со моделот и го ажурира погледот. Во нашиот проект, контролерот се справува со барањата за најава, регистрација, сервирање на податоци за издавачи и зачуваните акции од интерес на корисниците.

Зошто го избравме MVC патернот?

- Енкапсулација на одговорностите: Со делење на апликацијата на три различни слоеви, MVC обезбедува секоја компонента да има јасна одговорност. Ова одделување го прави кодот полесен за разбирање, тестирање и одржување.
- Скалабилност: Како што расте нашата апликација, MVC овозможува додавање на нови функционалности без нарушување на постоечките. Ова го увидовме при надоградбата за анализите и предвидувањата.
- Усогласување на технологии: MVC шемата е добро усогласена со избраните технологии – SpringBoot природно го поддржува MVC со својата вградена рамка за контролери, сервисни слоеви (модели) и restful крајни точки, и React кој е одличен за градење динамични погледи кои лесно се ажурираат врз основа на промените на состојбата.

Забелешка: Имаме мало отстапување од MVC шаблонот, конкретно техничката анализа која се процесира на frontend. Ова се должи на тоа што откривме готови функции на за извршување на овие пресметки кои беа погодни за нашите барања.