*Design Patterns in the project*

Имплементиравме **шаблон за дизајн MVC (Model-View-Controller)** во проектот заради неговите предности во организирање на кодот и подобрување на одржливоста.Во класата CompanyController, користиме Spring MVC анотација @Controller, што ја дефинира оваа класа како контрола за обработка на HTTP барања. **Model**: ја содржи логиката на податоците и управува со операциите со податоците, како што се складирање и обработка. **Controller**: ја посредува комуникацијата помеѓу моделот и приказот, обработувајќи ги барањата од корисникот и ажурирајќи го моделот и приказот соодветно.

Оваа архитектура го прави кодот помодуларен, што овозможува полесно додавање нови функционалности и тестирање на индивидуални компоненти. Дополнително, ги одвојува одговорностите, што ја зголемува одржливоста на проектот и ја намалува комплексноста на кодот.

Во проектот, исто така, е имплементиран **шаблонот Singleton** за управување со единствена глобална инстанца на одредена класа.**Зошто го користевме Singleton**: го осигурува постоењето на само една инстанца на класа во текот на извршувањето на програмата, овозможува пристап до истата инстанца на класа од различни делови на апликацијата, без повторно создавање, се избегнува непотребна потрошувачка на меморија со повторно користење на постоечката инстанца.

Класите CompanyRepository и DashboardRepository се обележани со @Repository, а класите CompanyServiceImpl и DashboardServiceImpl се обележани со @Service, така што овие анотации ги користиме за дефинирање на единечни (singleton) бинови. Класата CSVProcessor користи различни стратегии за парсирање на задолжителни и опционални полиња. На пример: parseValue() ја користиме за задолжителни полиња, а parseOptionalValue() за опционални полиња кои можат да бидат null.

**Template Method-**  Во методите getCompanyData, getStatsData, и getNews, користиме шаблон на обработка на податоци, каде извршуваме исти операции за обработка на податоци, но менуваме некои детали (на пример, различни пресметки на индикатори, филтрирање на податоци според години, итн.).

**Dependency Injection (DI)-**  Овој шаблон го имплементиравме преку Spring Framework користејќи анотации како @Service, @Repository, и @Controller.