MVC (Model–View–Controller)

Вовед

Во овој проект беше применет архитектонскиот шаблон MVC (Model–View–Controller) поради неговата способност да обезбеди јасна поделба на одговорностите помеѓу податоците, корисничкиот интерфејс и логиката за обработка на корисничките акции. Со ваквиот пристап се подобрува одржувањето и развојот на апликацијата.

MVC е еден од најкористените шаблони, особено кај веб, десктоп и мобилни апликации со графички кориснички интерфејс. Основната идеја е различните аспекти на апликацијата да бидат јасно одделени.

Основна идеја на MVC

MVC архитектурата го дели системот на три компоненти: Model, View и Controller. Секоја компонента има јасно дефинирана улога, а комуникацијата помеѓу нив е контролирана со цел да се намали зависноста и да се зголеми флексибилноста на системот.

Model

Model го претставува јадрото на апликацијата и ги содржи податоците, состојбата и бизнис логиката. Тој е независен од корисничкиот интерфејс и не знае како податоците се прикажуваат. Неговата задача е да ги обработува податоците и да ја одржува конзистентноста на системот.

View

View е задолжен за прикажување на корисничкиот интерфејс и ги визуелизира податоците што ги обезбедува Model. Тој не содржи бизнис логика и не ја менува состојбата на податоците, туку ги проследува корисничките акции кон Controller и се освежува при промена во Model.

Controller

Controller е посредник помеѓу View и Model. Тој ги обработува корисничките акции и одлучува кои промени треба да се направат во Model. Controller управува со текот на апликацијата, но не се занимава со прикажување или складирање на податоци.

Комуникација помеѓу компонентите

Процесот започнува со корисничка акција во View, која се проследува до Controller. Controller го ажурира Model, по што Model ги известува View-компонентите за промената. View потоа ги прикажува ажурираните податоци. Важно правило е дека Model никогаш директно не го менува View.

Предности и недостатоци

MVC овозможува јасна организација на кодот, полесно тестирање и едноставно одржување, како и користење на повеќе View-компоненти над ист Model. Сепак, воведува поголема структурна комплексност, што може да биде непрактично кај мали апликации.

Заклучок

MVC архитектонскиот шаблон претставува стабилна основа за развој на одржливи и флексибилни софтверски апликации. Со јасна поделба на Model, View и Controller, системот станува поорганизиран и поотпорен на идни промени.