УДК 004.021

ИСПОЛЬЗОВАНИЕ АРХИТЕКТУРНОГО ПАТТЕРНА MVC ПРИ ПРОЕКТИРОВАНИИ СИСТЕМЫ ТУРИСТИЧЕСКОГО ЦЕНТРА

А.И. ПОПОВ
(Представлено: канд. техн. наук, доц. И.Б. БУРАЧЕНОК)

В данной статье рассмотрены принципы построения взаимодействия компонентов в приложении на языке программирования Java. Рассмотрим основные идеи паттерна MVC и задачи, которые нужно реализовать для следования паттерну. Представлены преимущества при разработке системы по модели MVC.

Как правило, современные приложения строятся по шаблону MVC (Model-View-Controller или Модель-Вид-Контроллер). MVC – это не паттерн проектирования. MVC – это именно набор архитектурных идей и принципов для построения сложных систем с пользовательским интерфейсом. [1] Но для удобства его обычно называют шаблоном или паттерном. Идея данного шаблона проста – разделение обязанностей. Таким образом, любая разрабатываемая система должна быть разделена на три составные части, каждая из которых отвечает за свои задачи как показано на рисунке 1.

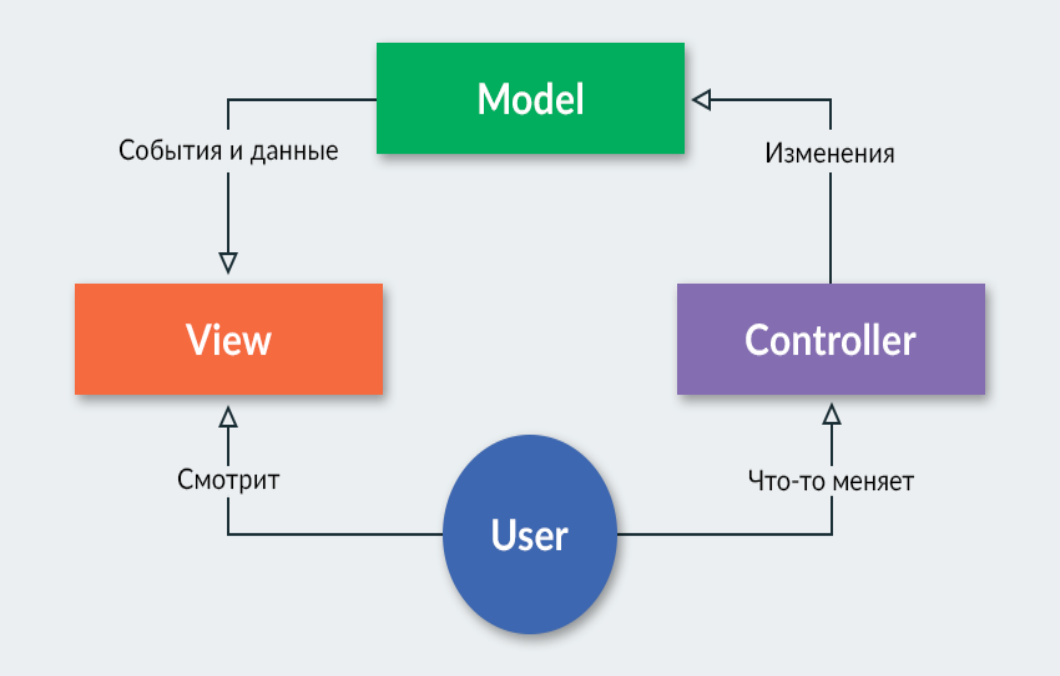

Рисунок 1. – Структура MVC приложения

Задача контроллера – обработка действий пользователя (нажатие по кнопкам, обработка запросов к серверу и т. д.); модель – предоставляет контроллеру представление данных, запрашиваемых пользователем, содержит всю бизнес-логику приложения; вид – обеспечивает представление данных, полученных из модели. Данный модуль отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом. [2]

Для того чтобы добиться реализации MVC нужно проделать несколько обязательных шагов.

Первый шаг – это разделение бизнес-логики приложения от пользовательского интерфейса. Таким образом приложение уже разбивается на два блока. Первый отвечает за реализацию бизнес-логики и будет являться ядром системы. В этом модуле и содержится реализация модели предметной области приложения. Второй модуль отвечает за интерфейс, который отображается пользователю и логику взаимодействия пользователя с приложением. Основная идея данного шага заключается в том, чтобы ядро системы, это то, что мы выделили как Model, могло быть разработано и протестировано независимо. Архитектура приложения после такого разделения показана на рисунке 2.

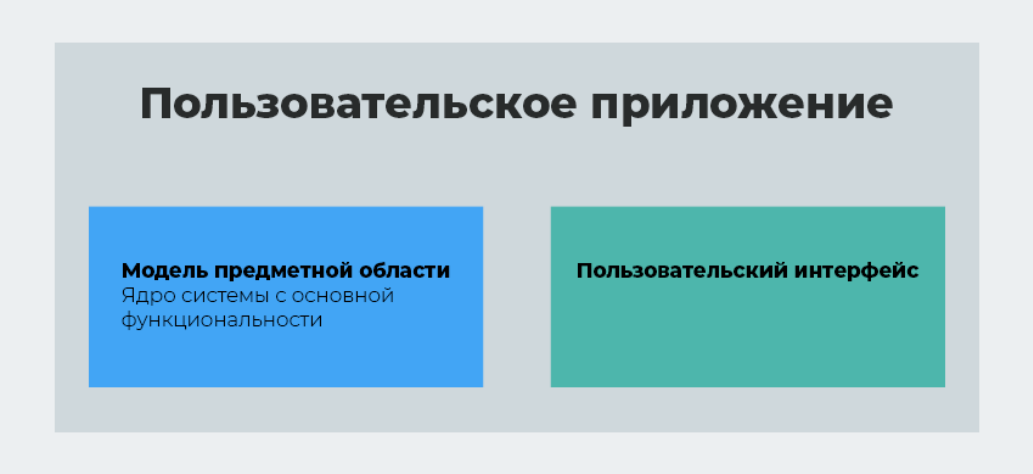

Рисунок 2. – Разделение приложения на модель и представление

Вторым шагом будет разделение приложения на ещё более низком уровне иерархии, то есть мы разделим представление, на два уровня, а именно на вид и контроллер. Вид – это то, что видит пользователь, а контроллер – это механизм, посредством которого пользователь может взаимодействовать с системой. Важно понимать, что элементы управления, например, кнопки на странице – это, по сути, часть контроллера. Но они также видны пользователю, как и любая часть вида. Поэтому при этом разделении подразумевается функциональное разделение интерфейса на части как показано на рисунке 3. Так как у интерфейса можно выделить две основные функции:

– выводить и удобно отображать пользователю информацию о системе

– вводить данные и команды пользователя в систему (передавать их системе) то по этим функциям мы и проводим разделение.

Первую функцию будет выполнять вид, а вторую контроллер.

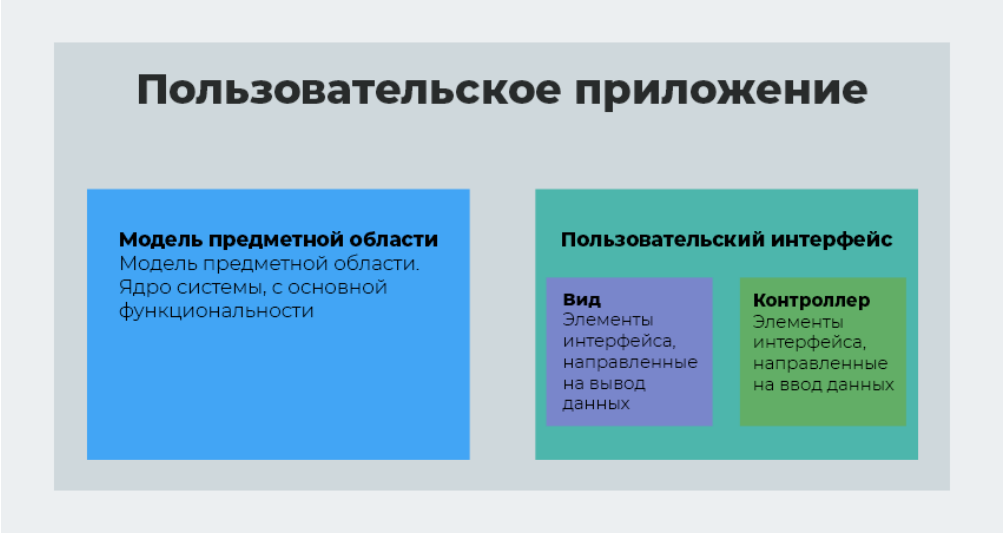

Рисунок 3. – Разделение приложения на модель, вид и представление

Когда пользователь вводит информацию через контроллер, он тем самым вносит изменения в модель. Точнее, пользователь вносит изменения в данные модели. Когда пользователь получает информацию через элементы интерфейса (через Вид), пользователь получает информацию о данных модели.

Следование данным правилам позволяет добиться следующих преимуществ: основная цель следования принципам MVC – отделить реализацию бизнес-логики приложения (модели) от ее визуализации (вида). Такое разделение повысит возможность повторного использования кода. Польза применения MVC наиболее наглядна в случаях, когда пользователю нужно предоставлять одни и те же данные в разных формах, например, в виде таблицы, графика или диаграммы, используя различные виды. При этом, не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на

кнопке, ввод данных). Также, если следовать принципам MVC, можно упростить написание программ, повысить читаемость кода, сделать легче расширение и поддержку системы в будущем. [3]

Рассмотрев принципы реализации MVC, построим примерную модель разбиения проектируемой системы для соблюдения требований паттерна. Модель структуры системы представлена на рисунке 4.

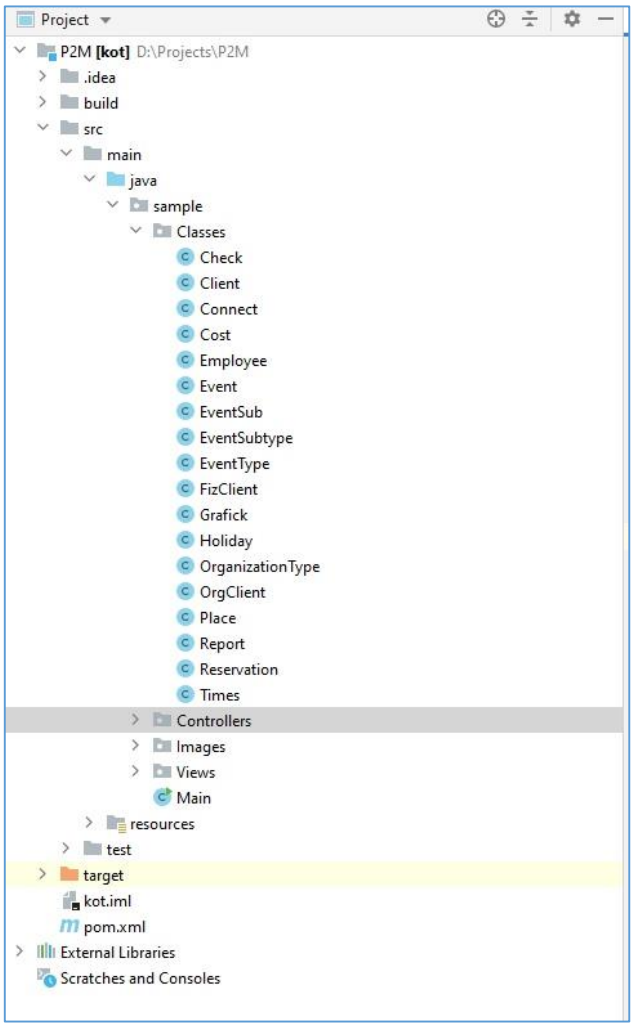

Рисунок 4. – Схема модели проекта

Проектирование системы туристического центра с использованием архитектурного паттерна MVC позволит упростить дальнейшее масштабирование системы. Кроме того, разделение приложения на составные части позволит разделить ответственность между различными компонентами системы и упростит дальнейшую разработку и реализацию новых функциональных решений.

ЛИТЕРАТУРА

1. Model-View-Controller. [Электронный ресурс] / Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller. – Дата доступа: 19.09.2020.
2. Что такое паттерн проектирования MVC в Java. [Электронный ресурс] / Pro-java. – Режим доступа: https://pro-java.ru/patterny-proektirovaniya-java/chto-takoe-pattern-proektirovaniya-mvc-v-java/. – Дата доступа: 18.09.2020.
3. MVC [Электронный ресурс] / Javarush. – Режим доступа: https://javarush.ru/quests/lectures/questcollections.level06.lecture01/. – Дата доступа: 18.09.2020.