РАЗРАБОТКА АРХИТЕКТУРЫ ВЕБ-СЕРВИСА

Юшкевич Е.С.
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Ионин В.С. – канд.техн.наук, доцент

Цель работы – обеспечение возможности разработки программного обеспечения для построения сложной структуры архитектуры Веб- сервиса. Проектирование подобных систем должно осуществляться с учетом потребностей пользователя. Системы ИТ-инфраструктуры и бизнес-целей. Современные инструменты и платформы упрощают задачу по созданию таких приложений, однако требуют учета требований потребностей пользователя и обеспечения управления ими в среде производственной эксплуатации. Основное назначение архитектуры- описание использования или взаимодействия основных элементов и компонентов приложения. Выбор структур данных и алгоритмов их обработки или деталей реализации отдельных компонентов является вопросами проектирования. В работе рассмотрены вопросы использования популярных шаблонов проектирования. В их основе лежит разделение данных приложения, интерфейса пользователя и управляющей логики на отдельные модули.

Как и любая другая сложная структура, программное обеспечение (ПО) должно строиться на прочном фундаменте. Неправильное определение ключевых сценариев, неправильное проектирование общих вопросов или неспособность выявить долгосрочные последствия основных решений могут поставить под угрозу все приложение.

Современные инструменты и платформы упрощают задачу по созданию приложений, но не устраняют необходимости в тщательном их проектировании на основании конкретных сценариев и требований. Неправильно выработанная архитектура обусловливает нестабильность ПО, невозможность поддерживать существующие или будущие бизнес-требования, сложности при развертывании или управлении в среде производственной эксплуатации.

Проектирование систем должно осуществляться с учетом потребностей пользователя, системы (ИТ-инфраструктуры) и бизнес-целей. Для каждой из этих составляющих определяются ключевые сценарии и выделяются важные параметры качества (например, надежность или масштабируемость), а также основные области удовлетворенности и неудовлетворенности. По возможности необходимо выработать и учесть показатели успешности в каждой из этих областей.

Основное назначение архитектуры – описание использования или взаимодействия основных элементов и компонентов приложения. Выбор структур данных и алгоритмов их обработки или деталей реализации отдельных компонентов являются вопросами проектирования. Часто вопросы архитектуры и проектирования пересекаются. Вместо того, чтобы вводить жесткие правила, разграничивающие архитектуру и проектирование, имеет смысл комбинировать эти две области. В некоторых случаях принимаемые решения, очевидно, являются архитектурными по своей природе, в других – больше касаются проектирования и реализации архитектуры.

Рассмотрим один из популярных шаблонов проектирования MVC (Model-View-Controller – «модель-представление-контроллер). В основе архитектурного паттерна MVC лежит идея разделения данных приложения, интерфейса пользователя и управляющей логики на отдельные модули: модель, представление и контроллер, что видно из рисунка 1 [1]. Модификация каждого модуля при этом может осуществляться независимо. Модель предназначена для извлечения данных из базы и манипуляции данными, функция представления – отображение этих данных в формате, доступном пользователю (в веб приложении – HTML-файл, пересылаемый сервером браузеру в ответ на запрос), а контроллер используется для управления описанными выше процессами. Такой подход отвечает принципу «единой ответственности», то есть ответственность за различные функции приложения чётко разделена между его модулями. Концепция «модель-представление-контроллер» не привязана не только к конкретному языку программирования, но и к конкретной парадигме (например, к объектно-ориентированному программированию).

Модификация каждого модуля при этом может осуществляться независимо. Модель предназначена для извлечения данных из базы и манипуляции данными, функция представления – отображение этих данных в формате, доступном пользователю (в веб приложении – HTML-файл, пересылаемый сервером браузеру в ответ на запрос), а контроллер используется для управления описанными выше процессами. Такой подход отвечает принципу «единой ответственности», то есть ответственность за различные функции приложения чётко разделена между его модулями. Концепция «модель-представление-контроллер» не привязана не только к конкретному языку программирования, но и к конкретной парадигме (например, к объектно-ориентированному программированию).

Говоря про шаблон MVC, стоит упомянуть один из самых популярных архитектурных стилей на сегодняшний момент – REST [2]. Они отлично сочетаются друг с другом. Архитектурный стиль взаимодействия компонентов распределённого приложения в сети REST (сокр. от англ. Representational State Transfer – «передача состояния представления») представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы,

основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

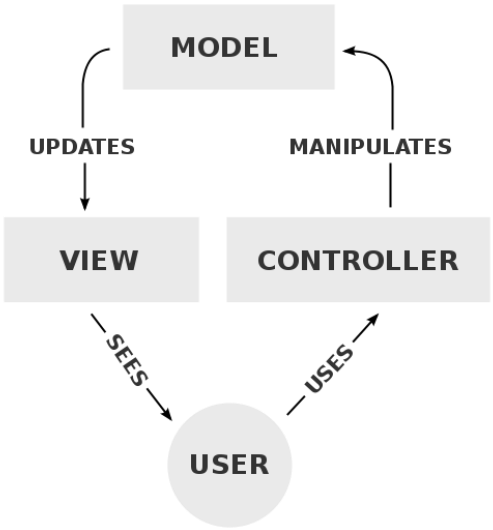

Рисунок 1 – Шаблон проектирования MVC

Полученный от клиента запрос (который отправляет браузер при взаимодействии пользователя и интерфейсом, например, HTTP GET или POST) анализируется контроллером и при условии его корректности в зависимости от запрашиваемого действия (просмотр или изменение данных) обращается к соответствующему методу модели. Основная функция контроллера – определение, вызов и координирование действий, необходимых для решения задачи, заданной пользователем и определяемой запросом, поступившим на сервер. Контроллер определяет нужную модель и её функции, а также формирует ответ в виде соответствующего представления.

Модель определяет правила, по которым приложение взаимодействует с базой данных. Любые функции приложения сводятся к обработке данных каким-либо образом, и именно в модели определяются ограничения по обрабатываемым данным и описывается логика манипуляций с данными, в результате которых решаются задачи пользователя. Если контроллер отвечает за организацию приложения, то в модели описаны алгоритмы по работе с данными.

Формат представления данных, являющихся результатом работы модели, описывается в представлении. Обычно представление состоит из шаблонов, которые динамически наполняются данными. Перед отправкой ответа клиенту из шаблонов и данных формируется HTML-страница.

Описанные выше подходы и архитектурные решения в настоящее время являются одними из самых популярных: они понятны, просты и надежны, а также без труда позволяют строить высоконагруженные приложения.

Список использованных источников:

1. Ruby on Rails Guides [Электронный ресурс]. – 2016. – Режим доступа: http://guides.rubyonrails.org. – Дата доступа: 06.05.2018.

2. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. — Springer Science & Business Media, 2011. — 528 p.