Лабораторная работа по предмету «Технологии разработки и тестирования программного обеспечения» №4

«Исследование архитектурного решения»

Подготовил:

Студент гр. №550504

Стрельцов Г. Ю.

Часть 1. Проектирование архитектуры

1.1 Определить тип приложения.

Тип разрабатываемого приложения: WEB приложение (SPA).

1.2 Выбрать стратегию развертывания.

Development mode: распределенная.

Production mode: нераспределенная.

Проект состоит из 2-ух приложений: web-client и web-server. Оба приложения могут разворачиваться как вместе на одном, так и на отдельных хостингах или облаках. В данной работе было выбрано решение развернуть оба приложение на одном облаке в продакшене. А в процессе разработки web-server будет разворачиваться на облаке, в то время как web-client на локальной машине разработчика.

1.3 Обосновать выбор технологий.

1) HTML5, CSS3, Angular 5, typescript для разработки клиентского web приложения. В современной веб-разработке особую гибкость предоставляют архитектурное решение SPA (Single Page Application). Данный подход позволяет разработчикам клиента и сервера вести разработку независимо друг от друга. Этот довод был ключевым в выборе стека технологий во главе Angular 5 фреймворка.

2) .NET, ASP.NET, Entity framework для разработки серверного web приложения. Поскольку большая часть команды с точки зрения серверных технологий знакома именно с .NET, до было принято использовать именно его на сервере. А те, кто с ним знаком не слишком хорошо, были задействованы в разработке web-client’a.

3) SQL Server 2016 для разработки базы данных. Данная технология отлично поддерживается выбранном стеком технологий на web-server’t, также является довольно удобной в использовании и поддержке. Данные аргументы являлись основными в выборе именно SQL Server 2016 в качестве технлогии для БД.

1.4 Указать показатели качества.

Производительность. Люди не хотят ждать много раз, пока загружается веб-страница. Чтобы не тратить драгоценное время, пользователь должен ждать максимум 1 секунду, чтобы загрузить любую страницу в веб-приложении. Меры во времени, необходимые для загрузки веб-страницы.

Надежность. Веб-приложение должно быть развернуто 24 часа в сутки без проблем для доступа к нему. Важно, потому что пользователи должны знать, что они могут использовать ресурс веб-приложения в любое время, в котором они нуждаются. Измеряет количество ситуаций, когда пользователь не может загрузить веб-страницу (этот счет должен быть ниже, чем 5 ситуаций в день).

Отзывчивый пользовательский интерфейс. Содержимое веб-приложения должно быть доступно для чтения на мобильных и настольных устройствах - в настоящее время все больше и больше пользователей просматривают Интернет, используя только мобильные устройства. Меры по подсчету несостоявшегося контента в мобильных устройствах (этот счет должен быть как можно меньше). Веб-приложение должно поддерживать одновременно не менее 500 пользователей одновременно, поскольку могут возникать проблемы с ответом на веб-контент, когда на веб-сайте много пользователей. Измеряет максимальное количество пользователей, которые могут одновременно использовать веб-приложение.

1.5 Обозначить пути реализации сквозной функциональности.

В данном приложении в связи с выбором нескольких внешних фреймворков: Angular 5, APS.NET Core 2 – фактически вся сквозная функциональность сосредоточена именно в них.

1.6 Изобразить структурную схему приложения в виде функциональных блоков. Выделить слои функциональности. Связи.

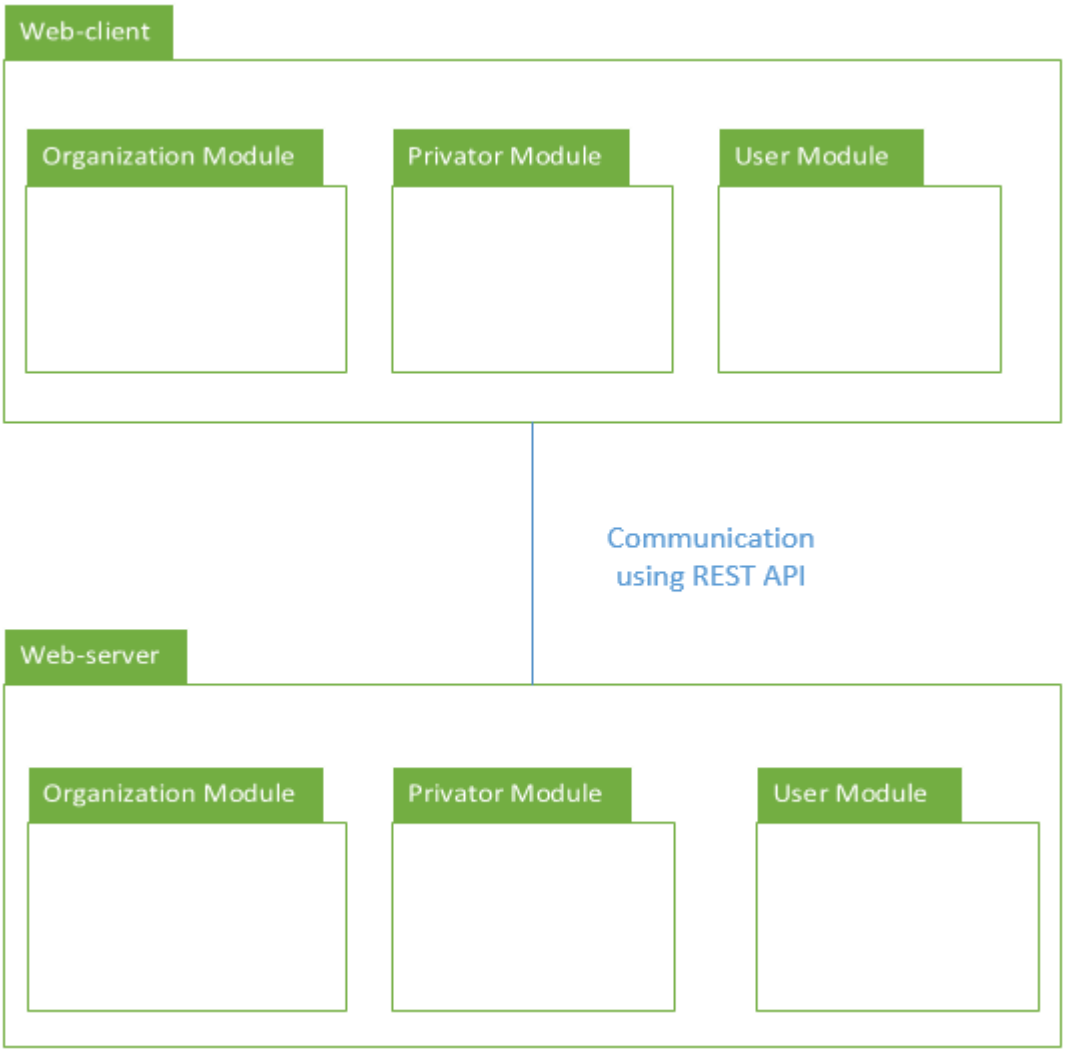

Рис 6.1 Структурная схема приложения

Часть 2. Анализ архитектуры

2.1 Проанализировать архитектуру разрабатываемого приложения

Разрабатываемое приложение построено в соответствии с архитектурным решением SPA, которое отличается повышенной гибкостью и простотой. Это позволило распределить участников команды на разработку независимых приложений: web-client, web-server.

2.2 Изобразить обобщённое представление архитектуры (как в предыдущей части)

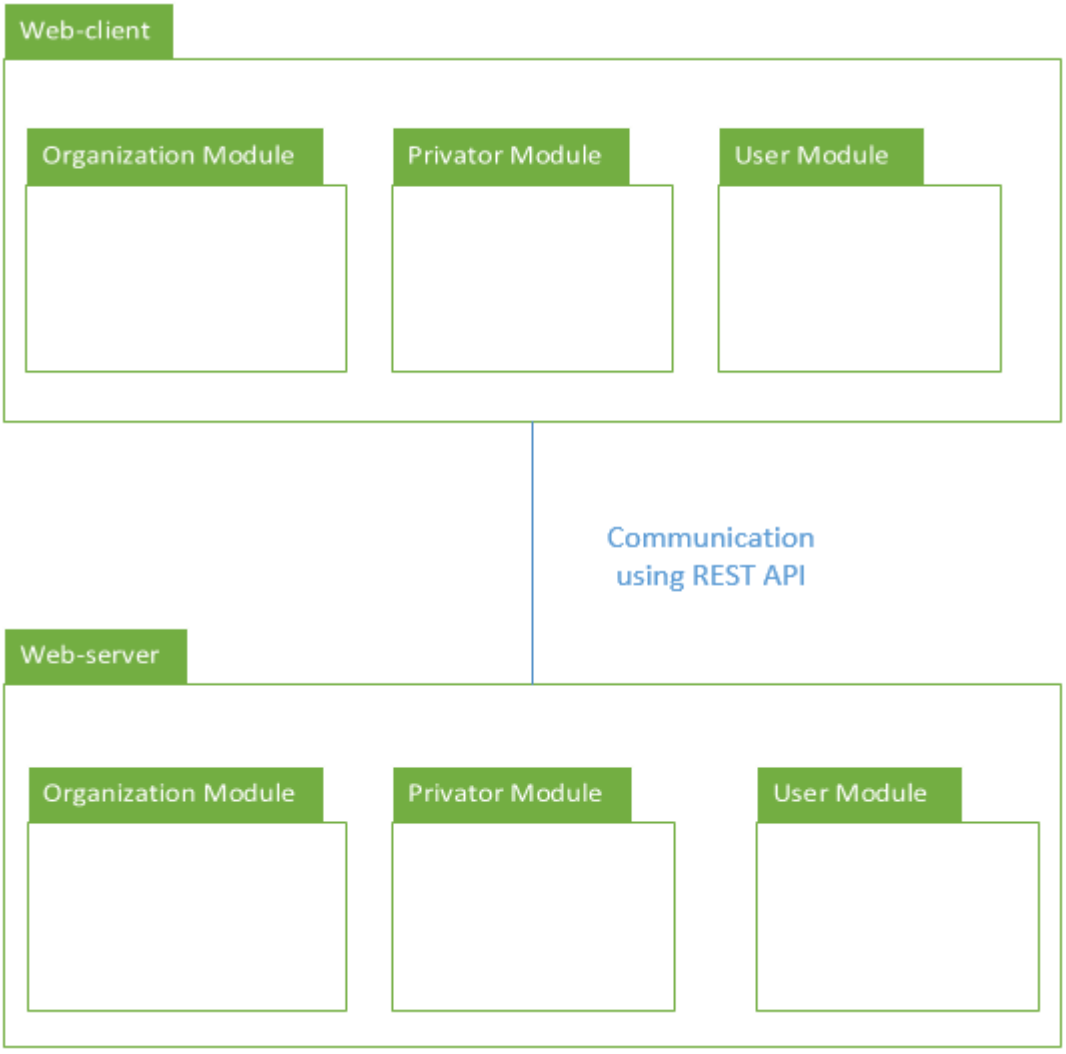

Рис 1.6.1 Структурная схема приложения

2.3 Используя автоматизированные средства обратной инженерии (например IBM Ration Rose) сгенерировать диаграммы классов (всей системы или какой-либо особенно интересной её части)

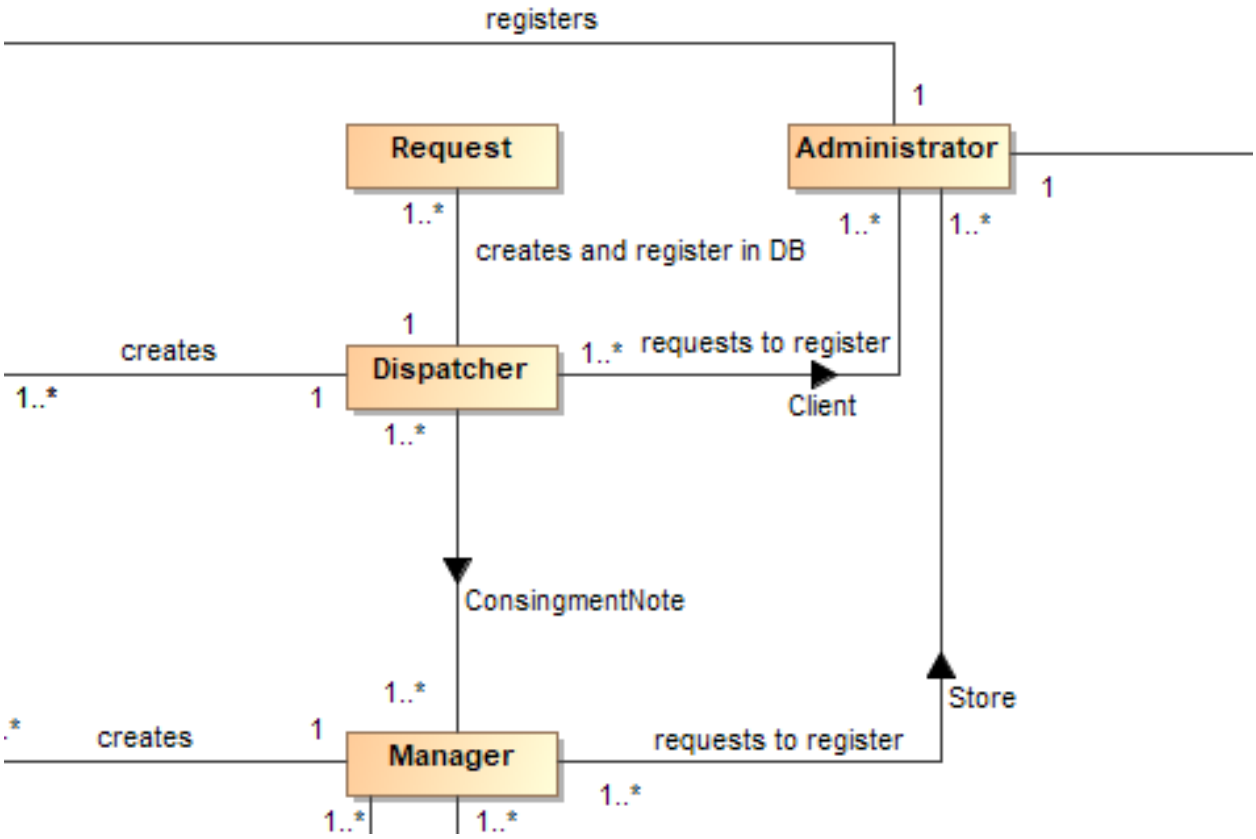

Рис 2.3.1 Взаимодействие пользователей модуля Organization.

Часть 3. Сравнение и рефакторинг

1. **Сравнить архитектуры «As is» и «To be»**

   Отличие в том, что в архитектуре «As is» система разворачивается на одном компьютере, а в «To be» система должна быть развернута на сервере, с которого пользовательский интерфейс должен быть виден в браузере пользователя

2. **Причины:**

   Значительно упрощает разработку

3. **Пути улучшений**

   Развернуть два отдельных приложения на сервере. При этом клиентское приложение занимается рендерингом контента под разные виды браузеров и виды устройств, в то время как серверное приложение занимается данными и их обработкой