Архитектура серверной части автоматизации работы операторов поддержки Help-desk систем

А. С. Владимирцев, Д. И. Степанов

Проект ГПО АОИ-1601 «Модуль автоматизации работы операторов поддержки для Help-desk систем»

Научный руководитель:
преподаватель кафедры АОИ
Голубева А. А.

Введение

Ни одно современное нагруженное веб-ориентированное приложение не способно существовать без реализации его серверной логики. Help-Desk система - это сервис технической поддержки пользователей, предназначенный для решения проблем пользователей, возникших в ходе использования программного продукта. В ходе анализа архитектурных решений система была декомпозирована на следующие части:

- back-end - это веб-сервер, который реализует бизнес-логику и API для связи всех подсистем приложения;
- service - подсистема, отвечающая за алгоритмическую часть поиска ответа на запросы пользователя;
- front-end - это клиент, с которым непосредственно взаимодействует пользователь системы;
- android client - это мобильный клиент осуществляющий взаимодействие пользователей с системой посредством мобильного приложения.

Подсистема back-end отвечает за обработку сообщений от других подсистем приложения, и непосредственно влияет на скорость обработки заявок, от которой и зависит довольство клиентов. Help-Desk система подразумевает под собой множество одновременно работающих пользователей, в связи с этим нам было необходимо разработать архитектуру, способную выдержать всю возложенную на него нагрузку.

Архитектура back-end’a

Для реализации серверной части системы использована технология Web-Api. Это похоже на паттерн MVC, в котором реализация view разрабатывается сторонним сервисам обращающимся к контроллерам Web-api. Эта концепция идеально

вписывается архитектуру системы, так как клиенты системы разрабатываются как отдельные модули.

В течение разработки были выявлены следующие проблемы.

Первой проблемой стало обновление данных у клиентов. Например, в систему пришла новая заявка, необходимо отобразить ее у всех подключенных операторов. Обычным подходом в данной ситуации является использование AJAX: клиент с некоторыми перерывами, допустим раз в 30 секунд, опрашивает сервер в поисках обновленных данных, в случае появилась некая новая информация, отображаем ее. Явным недостатком такой реализации является повышенная нагрузка на сервер. Использования вышеописанного метода потребует значительные вычислительные мощности, и несколько серверов для распределения нагрузки. Гораздо эффективней не опрашивать сервер, а сообщать клиентам об изменениях. Для реализации этого подхода идеально подходит SignalR - это библиотека для ASP.NET разработчиков, которая упрощает добавление в приложения компонентов, работающих в реальном времени[1]. Функциональность, работающая в реальном времени – это способность сервера отдать свежие данные подключенным клиентам немедленно, вместо того, чтобы ждать пока клиенты запросят данные.

Кроме того, использование SignalR решает еще одну проблему: выбора транспорта для доставки сообщения. SignalR использует несколько способов передачи сообщений, в случае, если можно использовать websocket, то он будет работать через websocket, если такой возможности нет, то он будет спускаться дальше, пока не найдет приемлемый транспорт.

Разработка любого приложения в той или иной степени связана с рисками. И для их уменьшения, а именно улучшения сопровождаемости приложения нами было решено использовать слоистую архитектуру (layer architecture) - это архитектура, при которой система состоит из некоторой упорядоченной совокупности программных подсистем, называемых слоями [2].

Базовые свойства слоёв:

- на каждом слое ничего не известно о свойствах (и даже существовании) последующих (более высоких) слоев;
- каждый слой может взаимодействовать по управлению (обращаться к компонентам) с непосредственно предшествующим (более низким) слоем через заранее определенный интерфейс, ничего не зная о внутреннем строении всех предшествующих слоев;
- каждый слой располагает определенными ресурсами, которые он либо скрывает от других слоев, либо предоставляет непосредственно последующему слою (через указанный интерфейс) некоторые их абстракции.

В системы были выделены следующие слои:

1. Слой данных - содержит в себе модели, а так е классы для взаимодействия с базой данных.
2. Слой сервисов - классы, реализующие основной функционал.

3. Слой контроллеров -представляют из себя API для взаимодействия с другими частями системы.

Введение слоев позволило уменьшить связанность классов между собой. Это позволяет вносить какие либо изменения, в том числе и добавления нового функционала, без ущерба для уже существующего.

Выводы

В ходе работы над серверной частью приложения была реализована архитектура с достаточным уровнем надежности для функционирования системы в условиях большого количества активных пользователей, а также обеспечивающая возможность поддержки системы с минимальными затратами за счет использования слоистой архитектуры.

Источники литературы

1. SignalR в помощь, или как оживить web [Электронный ресурс] - Режим доступа https://habrahabr.ru/company/dnevnik_ru/blog/167307/
2. Тепляков С. Паттерны проектирования на платформе .NET. — СПб.: Питер, 2015 - 286 с.