Введение

В процессе разработки любого приложения требуется поддерживать актуальность информации о производительности, объемах данных, сбоях и т.п. для своевременного вмешательства разработчика. На текущий момент данная задача решается путем различных методов тестирования, в ходе которых собираются необходимые данные и после анализа которых принимается решение о внесении изменений в приложении, или в требованиях к нему. Примерами может послужить нагрузочное тестирование, результатом которого являются данные о работоспособности приложения при стандартных нагрузках и пик нагрузок, при которых система выходит из строя, а также ручное тестирование, в результате которого оценивается общее восприятие приложения пользователем на основании времени отклика частей приложения, достоверности отображаемых данных и т.п. Схематически текущую схему разработки можно изобразить следующим образом:

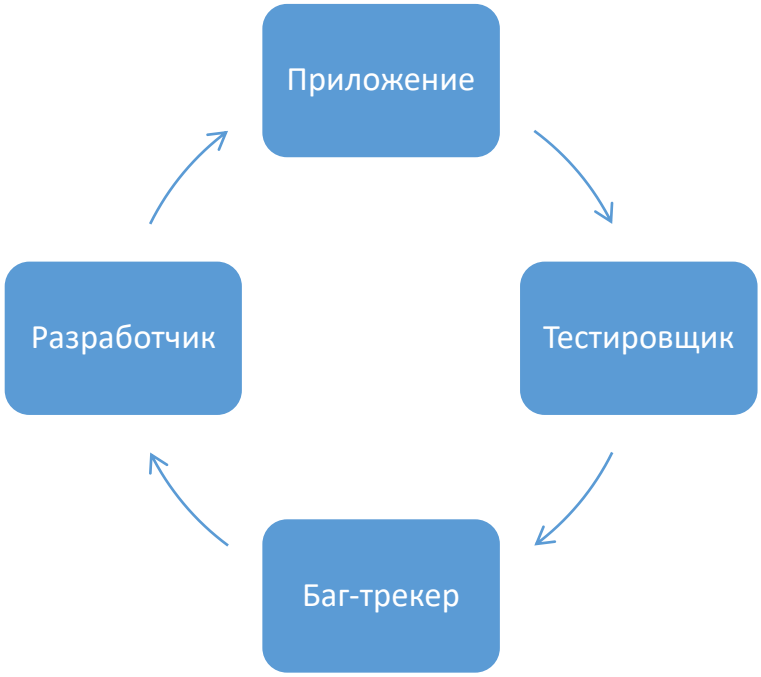
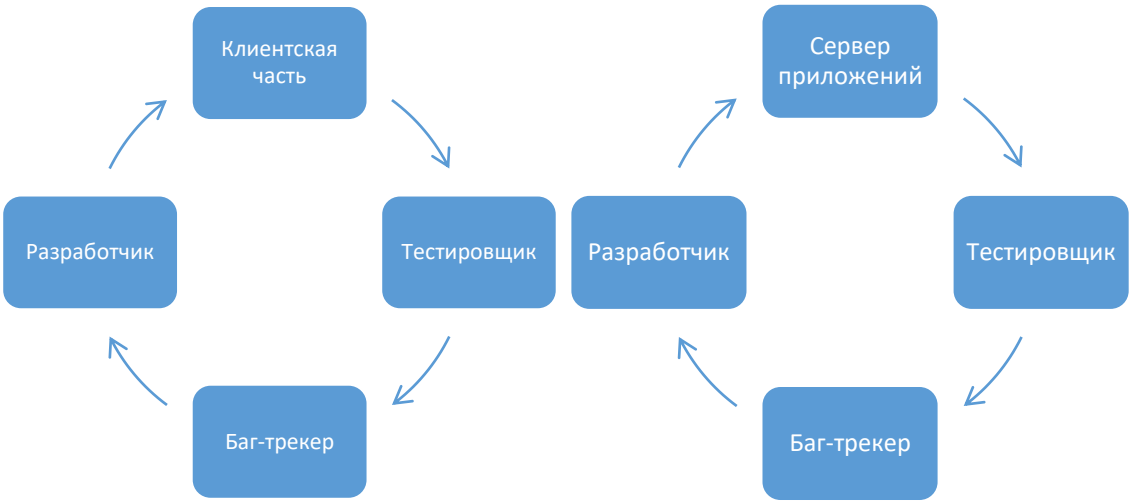


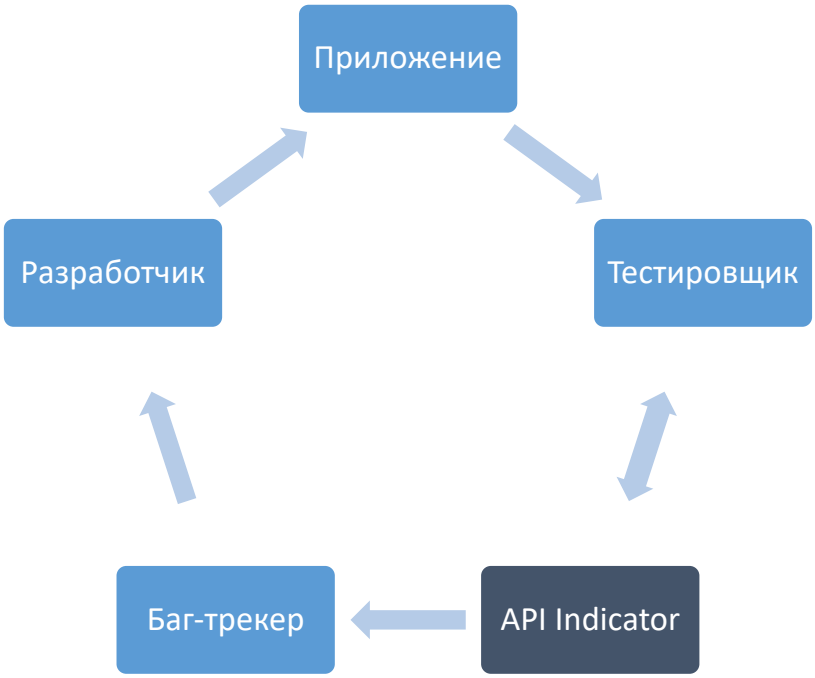
Схема является итерационной, т.е. все шаги разработки повторяются ровно до тех пор, пока приложение не будет удовлетворять требованиям или стандартам. В случае разработки Web-приложения, применив схему разработки получим 2 аналогичные схемы:



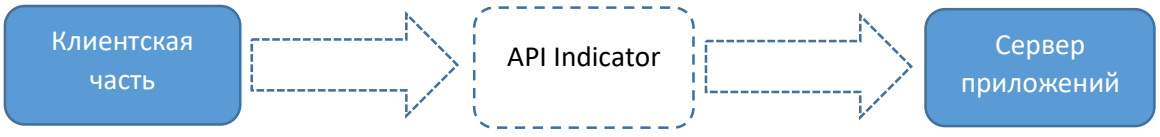
Из схем следует, что разработка Web-приложения является в какой-то степени «параллельной» и синхронизация происходит в ходе сборки приложения и последующего системного тестирования, в ходе которого тестируется все приложение. Из этого следует, что разработчики клиентской части не обязаны знать требования для сервера приложений. А значит, что в ходе тестирования, тестировщик клиентской части может получить время ответа сервера приложений, но утверждать о наличии проблемы на странице он не может. Он может создать соответствующую заявку в баг-трекере, которая будет рассмотрена, и в случае если проблема не на стороне сервера, отклонена. Т.е. на рассмотрение заявки потрачено время, которое можно было бы потратить на рассмотрение других заявок. API Indicator решает эту проблему.

Постановка задачи

API Indicator – приложение, позволяющее автоматизировать поиск проблемных мест в приложении. На основании статистических данных и критериев, заданных руководителем проекта приложение, автоматически определяет тип проблемы, ее важность и ее масштаб (относительно проекта в целом, относительно последней версии и т.п.). Схематически данный процесс выглядит следующим образом:



В обычный цикл разработки встраивается еще один компонент – API Indicator, который по сути является прокси-сервисом между клиентской частью и сервером приложений. Схематичный вид:



Основная задача сервиса – сбор и обработка данных, основанных на запросах, идущих от клиентской части к серверу. Т.е. запросы отправляются на сервер, а на сервис лишь информация об адресе запроса, размере данных, версии приложения и т.д.

По мере накопления такой информации API Indicator руководствуясь критериями оценивания, которые устанавливает руководитель проекта, по

запросу тестера может предоставить динамику производительности за какое-либо время для текущей страницы приложения. Далее тестировщик, основываясь на переданных ему данных о производительности, может создать заявку на баг-трекере, либо же, это сделает сам API Indicator, если руководитель проекта укажет данную опцию.

Обзор аналогов

На сегодняшний день один из самых популярных способов тестирования API – автоматизированное.

Для конкретизации требований, предъявляемых системе, рассмотрим существующие решения:

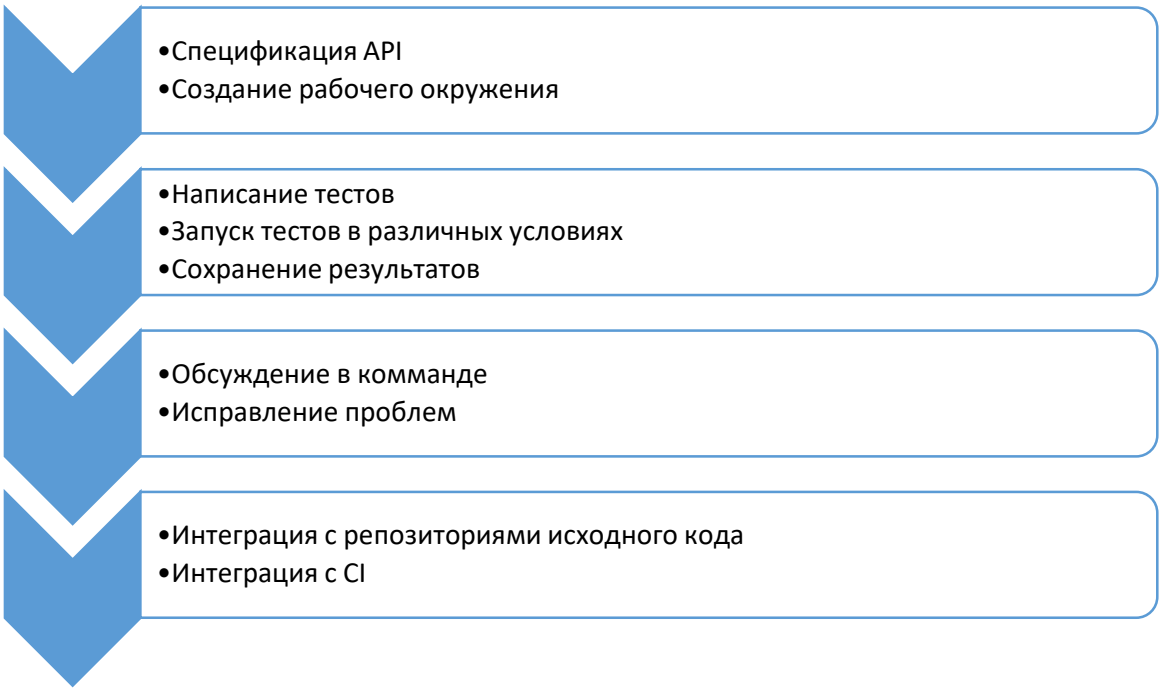
Postman

Страница проекта

Postman предоставляет огромный спектр возможностей по контролю API, наиболее интересные из них:

- Рабочие пространства
- Документирование
- Возможность работы в команде

Организация работы:



Плюсы:

- Возможность создания базы тестов
- Версионирование API/тестов
- Огромные возможности настройки валидации тестов
- Возможность делиться тестами/результатами с членами команды
- Возможность мониторинга статуса API

Минусы:

- Множество функций доступны платно (~12$/месяц/человек)
- Требуется время на написание/поддержку тестов
- Не собирается статистика

Анализ требований

На основании характеристик аналогов к реализуемой системе предъявляются следующие требования:

1. Простое администрирование системы
2. Минимум действий со стороны тестера
3. Открытый исходный код

Рассмотрим каждое требование подробнее

1. Простое администрирование системы

Администрирование системы заключается в управлении проектами, продуктами (модуль проекта), тестерами и версиями. Для начала работы системы необходимо зарегистрировать проект и продукты, после чего добавить одного или несколько тестировщиков. Система готова к эксплуатации. Система не требует предварительного описания или импорта схемы API, не требует написания тестов, создания условий для их выполнения.

2. Минимум действий со стороны пользователя

Основной пользователь системы – тестировщик. Работа в системе не должна особым образом отличаться от работы с тестируемым продуктом. Тестировщик получает доступ к функциям системы через специальный интерфейс, расположенный, например, в правом нижнем углу странице в виде маленького индикатора. В случае возникновения проблем при обработке запросов к API система оповестит тестировщика, изменив цвет индикатора, или наложив на него дополнительное изображение, характеризующее проблему.

3. Открытый исходный код

Система должна иметь открытый исходный код. Это может привлечь разработчиков/компании к дальнейшему развитию системы. Более активно будет происходить интегрирование новых технологий для взаимодействия как с API, так и с клиентской частью (изначально планируется что система поддерживает REST со стороны API, и Angular со стороны клиентской части), реализация мониторинга различных параметров API (изначально – отслеживание статуса ответа API, сбор статистики производительности), интеграция с другими системами, участвующими в разработке.