УДК 621.9.048.7

ИСПОЛЬЗОВАНИЕ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ ДЛЯ АВТОМАТИЗАЦИИ СБОРКИ И РАЗВЕРТЫВАНИЯ КРОСС-ПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ

Сумар А.А.
МИДО БНТУ, Минск, Республика Беларусь, andrey.sumar@gmail.com

Поскольку современные интернет-приложения обладают сложной многоуровневой архитектурой, содежат код, разделяемый между клиентом и сервером, при сборке используют различные конфигурации для разных окружений (режим разработки, режим тестирования и готовое приложение), а также имеют многочисленные задачи по сборке клиентской части (конкатенация и минификация скриптов, копирование файлов, автоматическая замена ссылок на JavaScript-библиотеки, компиляция файлов каскадных таблиц стилей, автоматическая замена ссылок на них, аннотация зависимостей в коде и пр.), применение примитивных методов внедрения программного обеспечения (ручное копирование файлов, правка конфигураций и отправка на сервер) сильно замедляет разработку и требует повторения рутинных действий при каждом изменении кода клиента либо сервера, а также при обновлении дизайна клиентского приложения.

Ввиду этого при разработке насыщенных интернет-приложений с использованиям языка JavaScript является целесообразным внедрить систему контроля версий (например, Git) и элементы непрерывной интеграции, которые позволяли бы решать следующие задачи:

- поддержка версионности исходных кодов приложения,
- хостинг исходных кодов приложения в интернете,
- копирование файлов сервера на production-сервер, доступный из интернета,
- копирование файлов клиента на production-сервер,
- копирование разделяемых файлов в соответствующие директории,
- конкатенация JavaScript-файлов клиента в один файл для уменьшения числа запросов,
- минификация этого JavaScript-файла для уменьшения размера и защиты от нежелательного анализа кода третьими лицами,
- замена многочисленных ссылок на различные JavaScript-файлы одной ссылкой на сгенерированный JavaScript-файл,
- подготовка файлов для сборки приложения под платформу Android,
- сборка приложения для Android и копирование результирующего APK-файла в необходимую папку.

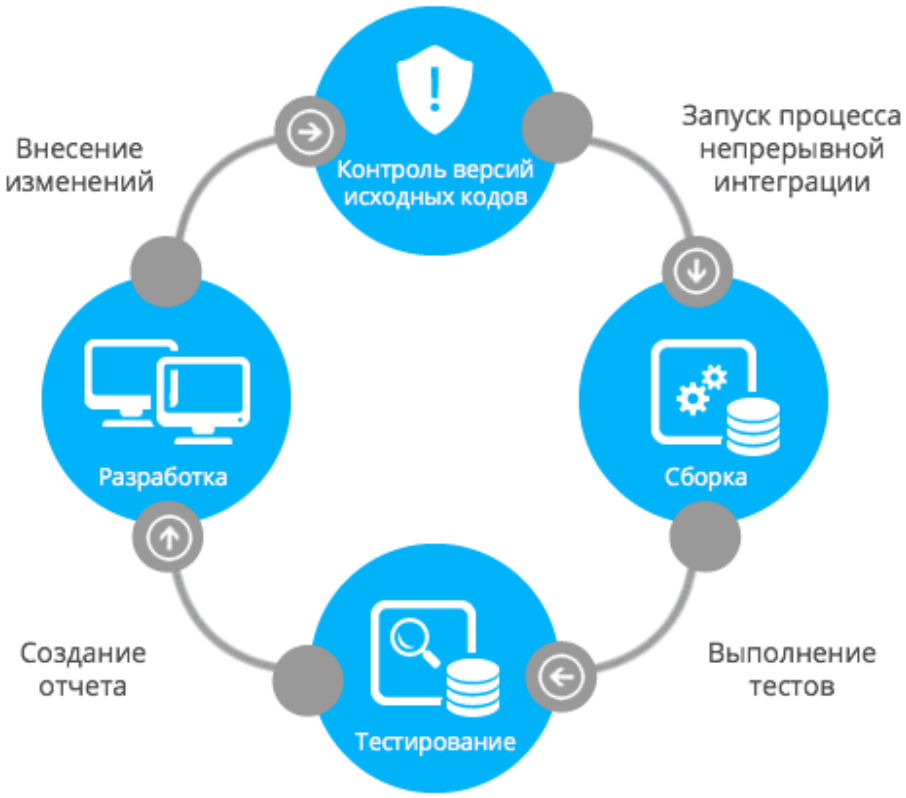

Рисунок 1 – модель непрерывной интеграции

Рассмотрим способ решения поставленых задач на примере клиент-серверного приложения, клиентская часть которого написана на языке JavaScript с использованием одного из наиболее распространённых фреймворков – AngularJS, а серверная часть также написана на языке JavaScript и выполняется в среде Node.js. При этом поскольку язык JavaScript используется в обеих частях приложения, существенная часть кода выполняется как на клиенте, так и на сервере.

Перечисленные выше требования, предъявляемые системе автоматической сборки и непрерывной интеграции, можно представить как задачи по преобразованию файлов исходных кодов приложения. Одним из инструментов описания, настройки и запуска таких задач является инструмент сборки Grunt.

В общую задачу сборки приложения («build») входят следующие подзадачи:

- очистка выходной директории,
- копирование файлов в выходную директорию,
- перечисление подлежащих конкатенации и минификации файлов,
- конкатенация файлов,
- минификация файлов,
- замена старых ссылок на файлы на ссылку на сгенерированный на предыдущих этапах файл в разметке.

```
grunt.registerTask('build', [
    'clean:dist',
```

```
    'copy:dist',
    'useminPrepare',
    'concat:generated',
    'uglify:generated',
    'usemin'
  ]);
```

Листинг 1 – структура задачи «build»

При необходимости сгенерировать на основе веб-приложения нативные приложения для мобильных платформ (например Android) с помощью таких инструментов как, например, Apache Cordova, может быть выделена ещё одна задача сборки («build-native»), которая дополняет предыдущую следующими подзадачами:

- копирование необходимых файлов для создания приложения для Android,
- запуск Apache Cordova для сборки приложения,
- копирование собранного приложения в необходимую папку,
- запуск приложения на эмуляторе либо реальном устройстве (опционально).

```
  grunt.registerTask('build-native', [
    'clean:dist',
    'copy:dist',
    'useminPrepare', //todo: lessminify
    'concat:generated',
    'uglify:generated',
    'usemin',
    'copy:native',
    'shell:buildNative',
    'copy:apk'
  ]);
```

Листинг 2 – структура задачи «build-native»

Важно отметить, что задача «build-native» запускается локально на компьютере разработчика, так как ему необходимо получить готовый к запуску APK-файл на своей машине, а задача «build» запускается как на машине разработчика для проверки production-конфигурации, так и на сервере, на котором приложение развёртывается.

Одним из самых распространённых средств хостинга такого рода приложений в интернете является платформа Heroku, которая предоставляет бесплатные учётные записи для приложений с открытым исходным кодом, которые не требуют больших вычислительных мощностей. При появлении нового коммита в репозитории системы контроля версий Git, Heroku обновляет файлы в своей файловой системе до новой версии, после чего выполняются шаги из раздела “postinstall” файла package.json проекта. К этим шагам относятся:

- bower install – установка всех необходимых для клиентского приложения библиотек из интернета,
- grunt build – сборка приложения задачей «build», процесс которой был рассмотрен выше.

Если сборка текущей версии кода была неуспешной, происходит откат до последней работоспособной версии, в противном случае обновлённая версия приложения становится доступной в интернете по адресу, заданному в панели администратора системы Heroku.

Таким образом, описанная методика соответствует важнейшим принципам непрерывной интеграции, таким как хранение исходного кода и всего, что необходимо для сборки и тестирования проекта, в репозитории системы управления версиями; автоматизация операций копирования, сборки и тестирования всего проекта и их простой вызов из внешней программы; постоянное наличие текущей стабильной версии вместе со сборками приложения для разработки, демонстрации и тестирования.

Список использованных источников

1. Непрерывная интеграция: улучшение качества программного обеспечения и снижение риска / М. Поль, Дюваль, Стивен М. Матиас III, Эндрю Гловер; – издательство «Вильямс», 2008 – 240 с.

2. Руководство Microsoft по проектированияю архитектуры приложений / С. Сомасегар, С. Гатри, Д. Хилл, 2010 – 529 с.

3. Automate with Grunt: The Build Tool for JavaScript / Brian P. Hogan, 2014 – 84 с.

4. Getting Started with Grunt: The JavaScript Task Runner / J. Pillora; - издательство «Packt», 2014 – 132 с.