Selectel

Статические сайты в облачном хранилище

Андрей Емельянов

Содержание

Введение

Большинство современных веб-сайтов являются динамическими. Их страницы формируются на стороне сервера, а затем отображаются в браузере пользователя. Многие такие сайты функционируют на основе простых и удобных CMS (систем управления контентом) в состав которых, как правило, входит WISYWYG-редактор, простой и интуитивно понятный. При всех очевидных удобствах при работе с динамическими сайтами возникает целый ряд неудобств.

Функционирование динамического сайта обеспечивается «связкой» из веб-сервера, приложения для генерации страниц и базы данных. Взаимодействие всех этих компонентов нередко сопряжено с серьезными затратами системных ресурсов. Чтобы снизить нагрузку и уменьшить время генерации страниц, используется кэширование, но его можно использовать не всегда и не везде. Обычно кэшируется контент, имеющий большой объем — например, графика. Также кэширование необходимо для минимизации числа запросов к базе данных.

Отдельные трудности связаны также с настройкой веб-сервера и программного обеспечения. Кроме того, динамические сайты уязвимы для DDOS-атак.

В связи со всеми описанными неудобствами и трудностями в последнее время существенно возрос интерес к статическим сайтам. Статический сайт представляет собой набор файлов (HTML, js, графика, шрифты и т.п.), размещенных на сервере. Для его обслуживания не требуется больших затрат системных ресурсов; с резервным копированием тоже не возникает никаких трудностей.

Страницы для таких сайтов можно создавать и редактировать в любом текстовом редакторе. Для объединения контента с шаблонами можно использовать так называемые генераторы статических сайтов. Генераторов существует очень много (на одном только GitHub их опубликовано несколько десятков).

Ниже мы рассмотрим наиболее популярные и распространённые генераторы.

Выбираем генератор статических сайтов

Генератором статических сайтов называется программный инструмент, превращающий текстовые записи (с разметкой или без) в статичные HTML-страницы. Все инструменты такого рода работают примерно одинаково: берется контент, склеивается с шаблоном, после чего отправляется на хостинг. Количество существующих генераторов статических сайтов исчисляется сотнями, если не тысячами.

Как из всего этого множества выбрать действительно стоящий продукт? Мы решили подготовить свой сравнительный обзор, который, как мы надеемся, поможет кому-то из вас определиться с выбором. При подготовке обзора мы учитывали следующие критерии:

- простоту установки и настройки;
- наличие дополнительных расширений и плагинов;
- уровень поддержки (как развивается продукт, как часто он обновляется, имеется ли подробная документация);
- поддерживаемые способы деплоя

MiddleMan, Jekyll, Octopress

Все три из перечисленных генераторов написаны на Ruby и во многом схожи по набору функций. Именно поэтому мы решили объединить их в одну группу. Процедура установки всех этих продуктов сопряжена с некоторыми трудностями: версии Ruby, включенные в официальные репозитории Linux-систем, скорее всего будет недостаточно — придется обновить ее до самой последней. Также потребуется установить менеджер пакетов RubyGems и менеджер версий Rbenv.

Несомненным преимуществом Middleman является хорошая и подробная документация, написанная простым и понятным языком. Для него написано немало расширений и плагинов, список которых постоянно обновляется.

Поддерживается деплой с помощью FTP, SFTP, rsync, git (на официальном сайте выложены скрипты для автоматизации процедуры деплоя). Возможен также деплой на AWS и BitBalloon (имеются соответствующие плагины)

Jekyll известен в первую очередь тем, что используется в качесте дефолтного движка для статических сайтов на основе GitHub Pages. Очень часто он используется для ведения блогов. Несомненным преимуществом Jekyll является

поддержка разметки Liquid: это дает возможность создавать шаблоны, используя конструкции исключительно языка разметки, а не языка программирования.

Расширений и плагинов для Jekyll существует довольно много. Официальные плагины «заточены» в основном на работу с GitHub Pages. Плагины, опубликованные на GitHub, в основном предназначены для расширения возможностей работы с блогами (добавление облака тэгов, полнотекстовый поиск по блогу и даже специализированный плагин для научных и образовательных блогов). Поддерживается деплой по FTP, а также с помощью rsync и git.

В отличие от двух предыдущих продуктов, Octopress является специализированным генератором: он предназначен исключительно для блогов и по сути представляет собой надстройку над Jekyll с дополнительными плагинами и responsive-шаблоном, обеспечивающими более удобное ведение блогов.

В качестве формата разметки постов по умолчанию используется Markdown, но можно использовать и обычный HTML. Несомненным плюсом Octopress является поддержка переезда с других площадок: например, все записи из блога на Wordpress можно перенести в новый статический блог при помощи специального скрипта (правда, велика вероятность того, что после переноса оформление некоторых текстов может «поломаться», и их потребуется править вручную). «Из коробки» поддерживается и работа с сервисом Disqus, что упрощает перенос комментариев. Блог на основе Octopress можно интегрировать с социальными сетями (Facebook, Twitter, Google Plus и другими). Существуют плагины, реализующие, например, вставку календарей (похожих на те, что иногда встречаются в блогах на Wordpress), списка похожих постов, облака тэгов и так далее.

По умолчанию поддерживается деплой с помощью git (на GitHub Pages или Heroku) или rsync (на любой хостинг, где можно настроить SFTP или можно запустить rsync). Можно настроить и деплой по FTP (о том, как это сделать, можно прочитать, например, здесь).

Hyde

Этот генератор статических сайтов изначально замышлялся как полный аналог Jekyll, только написанный на Python — отсюда и название, отсылающее к знаменитой повести Р. Л. Стивенсона «Странная история доктора Джекилла и мистера Хайда».

Следует различать старую и новую версию Hyde. Старая версия основана на Django templates; в настоящее время ее разработка приостановлена (последние коммиты в репозитории на GitHub датируются 2009–2010 годом). Новый Hyde (см. также репозиторий на GitHub) в настоящее время находится в активной разработке.

По функциональности новый Hyde не отличается от MiddleMan и Jekyll. Существенный недостаток у него только один, и он уже был указан выше: проект находится в стадии активной разработки. Именно поэтому документация к нему представлена пока что в очень сжатом и лаконичном виде, а плагинов и расширений существуют очень мало (вот их небольшой список на официальном сайте). Поддерживается деплой на GitHub Pages и Amazon S3.

Будем надеяться, что в будущем этот инструмент получит дальнейшее развитие и станет более удобным в работе.

Pelican

Pelican также написан на Python. По сравнению со многими генераторами статических сайтов он обладает исключительно широким набором функций: работа с черновиками, интеграция с социальными сетями, добавление изображений, конвертация HTML-страниц в PDF, поддержка многоязычности и многое другое. Он очень хорошо подходит для ведения блогов (существует плагин для переноса блогов на Wordpress). Посты можно писать в Markdown, а также в форматах reStructuredText и Asciidoc.

Устанавливается Pelican через pip. При установке пользователю будут заданы несколько вопросов: где хранить файлы сайта, как будет называться сайт, куда и каким способом его нужно деплоить. Поддерживается множество способов деплоя: по FTP, по SSH, на Amazon S3, GitHub Pages, Dropbox и RackSpace Cloud Files.

Grow

Grow (см. также официальный репозиторий на GitHub) — очень интересный и перспективный инструмент. Он написан на Python. Чтобы установить Grow, достаточно скачать скрипт с официального сайта — все необходимые пакеты будут установлены в автоматическом режиме.

В основе Grow лежит подход «конфигурация, а не код». Что это значит? Чтобы создать новый проект (в терминологии Grow проекты называются подами — pods), нужно клонировать на локальную машину тему, которая представляет собой репозиторий на GitHub. Тема включает набор конфигурационных файлов, с помощью которых описывается вся архитектура веб-сайта. Никакого программного кода при этом писать не нужно.

Все настройки проекта хранятся в конфигурационном файле podspec.yaml. В нем указываются следующие параметры:

- метаданные проекта (имя и т. п.);

- используемые в проекте инструменты предварительной обработки (например, SASS, Closure Compliler или другие);
- информация по локализации сайта (в частности, список локалей, в которых сайт будет доступен);
- информация о статичных файлах и специальных страницах;
- настройки деплоя.

Как осуществляется в Grow работа с контентом? Весь редактируемый контент (он может быть представлен как в формате Markdown, так и в HTML) хранится в директории /content. Структрура страниц описывается в конфигурационных файлах в формате YAML (см. пример здесь). Во время сборки сайта Grow генерирует страницы на основе прописанных настроек.

Grow может автоматически переводить текстовые фрагменты — для этого используется библиотека Goslate library, работающая с Google Translate. Чтобы перевести сайт, достаточно просто выполнить команду translate.

В качестве площадки для деплоя можно указать любой веб-сервер. Поддерживается деплой на Dropbox, Google Cloud Storage, Amazon S3, Dropbox, Google AppEngine.

Конечно, в рамках беглого обзора рассказать обо всех особенностях работы с Grow вряд ли возможно. Рекомендуем попробовать — инструмент очень перспективный.

Nanoblogger

Этот генератор статических сайтов, ориентированный на создание блогов, примечателен тем, что написан на bash. В качестве основных инструментов для создания статичных HTML-страниц он использует утилиты командной строки cat, grep и sed. При всей своей простоте Nanoblogger по возможностям не уступаем многим генераторам, написанным на Python или Ruby. Из его полезных функций можно выделить поддержку Atom/RSS, возможность создания на сайте календаря, сортировку постов по категориям, создание архива постов и другие.

Nanoblogger включен в официальные репозитории большинства популярных дистрибутивов Linux и устанавливается при помощи стандартного менеджера пакетов.

С Nanoblogger удобно работать из командной строки. Все команды подробно описаны в документации, их синтаксис прост и понятен. Исходный код также написан предельно просто, в случае необходимости его всегда можно

модифицировать и «подогнать» под нужды конкретного проекта (см. например, публикацию, в которой автор делится собственным опытом конфигурирования блога на основе Nanoblogger).

Для nanoblogger существуют плагины и расширения. Официальный набор плагинов (nanoblogger extras) также включен в официальные репозитории и устаналивается стандартным способом.

К сожалению, в 2013 году работа по развитию и усовершенствованию Nanoblogger была приостановлена на неопределенный срок.

DocPad

DocPad написан на CoffeeScript. Для работы с ним на клиентской машине должен быть установлен NodeJS. Многими он используется для блогов, но реальные возможности его применения гораздо шире. Этот продукт не представляет собой генератор статических сайтов в чистом виде: его можно использовать и как генератор, и как движок, и как шаблонизатор. DocPad оснащен достаточно удобным API, который позволяет использовать только те функции, которые нужны в данный момент; остальные всегда можно реализовать самостоятельно. Несомненным преимуществом DocPad является, конечно, очень подробная документация. Кроме того, на официальном сайте опубликованы так называемые «скелеты» — заготовки, на основе которых пользователи могут создавать собственные сайты.

Для DocPad написано множество разнообразных плагинов. Из наиболее интересных расширений отметим WYSIWYG-редакторы и веб-интерфейсы, облегчающие публикацию постов в статическом блоге. На официальном сайте опубликованы скрипты, автоматизирующие деплой на различные площадки: Heroku, Appfog, Windows Azure, Docker, GitHub Pages и другие. Имеется и специализированный скрипт для деплоя в облачные хранилища — Amazon S3 и GoogleStorage.

Генераторы статических сайтов: итоговая сравнительная таблица

Генератор	Язык	Простота установки и настройки	Поддержка	Расширения	Способы деплоя
MiddleMan	Ruby	Требуется последняя версия Ruby и менеджер пакетов RubyGems	На официальном сайте представлена подробная документация. Проект постоянно обновляется (последние коммиты в официальном репозитории на GitHub были сделаны менее месяца назад).	Имеется большое количество плагинов и расширений: дополнительные функции для блоггинга, интеграция с Disqus и Google Analytics, плагины для автоматизации деплоя и другие. Регулярно появляются новые плагины.	FTP, SFTP, rsync, Git, AWS, BitBalloon. Деплой в наше хранилище возможен по FTP или SFTP
Jekyll	Ruby	Требуется последняя версия Ruby, менеджер пакетов RubyGems и менеджер версий Rbenv	На официальном сайте представлена подробная документация. Находится в активной разработке и постоянно обновляется.	Плагинов и расширений довольно много; в основном они предназначены для ведения блогов.	Git, FTP. SFTP, rsync, Amazon S3, Heroku. Деплой в наше хранилище возможен по FTP или SFTP.
Octopress	Ruby	Требуется последняя версия Ruby, менеджер пакетов RubyGems и менеджер версий Rbenv	Имеется подробная документация. Проект активно разрабатывается, поддерживается и регулярно обновляется.	Существует множество плагинов, расширяющих возможности блоггинга: добавление календарей, облака тэгов, интеграция с социальными сетями и другие.	GitHub Pages, Heroku, FTP, SFTP, rsync. Деплой в наше хранилище возможен с помощью FTP или SFTP.
Hyde	Python	Устанавливается через pip, дополнительных зависимостей не требует	Проект находится в начальной стадии разработки; документация написана в крайне сжатом виде, публикаций по новому варианту Hyde в сети очень мало.	Плагинов очень мало, и их возможности очень ограничены.	GitHub Pages, Amazon S3, SFTP. Деплой в наше хранилище возможен по SFTP

Генератор	Язык	Простота установки и настройки	Поддержка	Расширения	Способы деплоя
Pelican	Python	Устанавливается с помощью pip; может потребовать установки и настройки дополнительных python-модулей	На официальном сайте опубликована подробная документация. Проект активно разрабатывается и поддерживается; последняя версия была опубликована в июле 2014 года.	Имеется много плагинов для расширения возможностей ведения блогов	FTP, SSH, Dropbox, Amazon S3, Rackspace Cloudfiles. Деплой в наше хранилище возможен по FTP
Grow SDK	Python	Устанавливается просто — достаточно скачать с официалного сайта скрипт и запустить его.	На официальном сайте опубликована подробная документация с видеоиллюстрациями.	Плагинов. расширяющих функциональность, нет. На официальном сайте размещены дополнительные темы и шаблоны.	Dropbox, Google Cloud Storage, Amazon S3 Dropbox, Google AppEngine. Деплой в наше хранилище возможен с помощью утилиты supload.
Nanoblogger	Bash	Устанавливается с помощью стандатрного менеджера пакетов.	Имеется краткая, но при этом понятная и четко описывающая все основные операции документация. Разработка проекта в настоящее время приостановлена на неопределенный срок.	Имеется небольшой пакет расширений для блогов, устанавливаемый из официального репозитория	rsync, FTP.Деплой в наше хранилище возможен по FTP;
DocPad	Coffee-Script	Для работы с DocPad на клиентской машине должны быть установлены NodeJS и менеджер пакетов NPM.	На официальном сайте опубликована подробная документация. Проект находится в активной разработке и постоянно обновляется.	Плагинов и расширений очень много; идет активная работа по созданию веб-интерфейса для статических сайтов.	Heroku, Appfog, Windows Azure, Docker, GitHub Pages. Деплой в наше хранилище возможен по FTP или SFTP.

Статические сайты: настройка и оптимизация

Главным критерием отличной работы сайта с точки зрения пользователя является, конечно же, скорость загрузки компонентов. Если сайт по тем или иным причинам загружается слишком долго, это неизбежно приводит к потере посетителей, которым надоедает ждать. Чтобы сделать сайт быстрым и удобным, нужно проделать определенную работу по его оптимизации.

В этом разделе мы дадим ряд рекомендаций, с помощью которых можно увеличить скорость работы статического сайта.

Совет 1. Не забывайте о настройках кэширования

Любая веб-страница включает множество разных элементов: изображения, скрипты, файлы стилей и так далее. Пользователь, посещающий страницу в первый раз, получает все эти элементы, выполнив ряд HTTP-запросов. Чтобы избежать повторной загрузки большого количества файлов, используется кэширование.

В основе модели кэширования, используемой в протоколе HTTP, лежат так называемые валидаторы — специальные заголовки, используемые клиентом для того, чтобы убедиться, что кэшируемый документ все еще актуален. Благодаря валидаторам клиент может проверять состояние документа, не передавая на сервер кэшированную копию целиком. В свою очередь сервер передает в ответе документ только в том случае, если полученный им валидатор свидетельствует о наличии в кэше клиента устаревшей копии.

Валидаторы подразделяются на сильные и слабые. Сильные валидаторы появились в HTTP/1.1. Они называются так потому, что они изменяются всякий раз, когда изменяется файл. К ним относятся так называемые ETags (entity tags). ETag представляет собой идентификатор содержимого документа; он изменяется, если в документе изменится хотя бы один бит. В качестве идентификатора может использоваться, например, MD5-сумма содержимого документа. Когда клиент запрашивает с сервера документ, значение ETag передается в ответе, например:

```
HTTP/1.1 200 OK
Server: Selectel_Storage/1.0
Accept-Ranges: bytes
Last-Modified: Mon, 18 Aug 2014 12:25:38 GMT
X-Timestamp: 1408364738.80296
```

```
Content-Type: image/jpeg
Content-Length: 458073
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Last-Modified, ETag, X-Timestamp
ETag: «ebef3343a7b152ea7302eef75bea46c3»
Date: Wed, 20 Aug 2014 11:52:48 GMT
```

При повторном запросе этого же документа сохраненное значение валидатора передается уже в заголовке If-None-Match:

```
GET / HTTP/1.1
Host: example.org
If-None-Match: «ebef3343a7b152ea7302eef75bea46c3»
```

Если документ не был изменен, то в ответе сервер вернет только заголовки и код 304 Not Modified. Иначе сервер вернет код 200 и передаст новую версию документа, а также новое значение ETag для нее.

В нашем хранилище и ETag гененерируется сразу же после загрузки файла; он соответствует MD5-хэшу содержимого. Если контент изменяется, то изменяется и ETag.

Слабыми называются валидаторы, которые не обязательно изменяются при каждом изменении файла.

Примером слабого валидатора может служить заголовок Last-Modified. Значением этого заголовка является дата последнего изменения файла. В нашем хранилище оно устанавливается автоматически. Если указать при запросе в заголовке If-Modified-Since дату, более раннюю, чем та, которая в данный момент содержится в заголовке Last-Modified, то ответом также будет являться 304 Not Modified.

Сильные валидаторы могут использоваться в любом контексте. Слабые валидаторы используются в контексте, который не зависит от точного содержания файла.

Например, валидаторы обоих типов могут быть использованы в GET-запросах с условием (If Modified Since или If None Match). Однако при скачивании файлов по частям могут использоваться только сильные валидаторы — иначе клиент получит файл в неконсистентном виде.

Совет 2. Обратите внимание на заголовок Cache-Control

Чтобы устанавливать срок хранения в кэше браузера копии файла, оригинал которого находится в хранилище, используется заголовок Cache-Control

с директивой max-age. Благодаря этому заголовку можно достаточно сильно увеличить скорость загрузки сайта — если файл закэширован, то браузер будет мгновенно отображать контент из кэша, не делая ни одного запроса к сайту.

Время хранения файла в кэше указывается в секундах:

```
Cache-Control: max-age=7200
```

В приведенном примере оно составляет 7200 секунд (2 часа). Обычно таким способом кэшируются CSS, JS и графические файлы. Их желательно кэшировать навсегда, а при изменении содержимого изменять ссылки на них в HTML. В RFC 2616 рекомендуется указывать время кэширования, не превышающее 1 год:

```
Cache-Control: max-age=31536000
```

Если требуется, чтобы определенный файл не кэшировался, а всегда отдавался «свежим», для заголовка Cache-Control устанавливается следующее значение:

```
Cache-Control: no-cache
```

Оно указывает, что элемент вообще не должен кэшироваться и что клиент должен запрашивать его при каждом обращении к хранилищу (время загрузки файла в этом случае увеличится, так как придется скачать тело файла).

Еще один способ, с помощью которого можно всегда получать файл в актуальной версии, заключается в добавлении к имени файла контрольной суммы содержимого. Если содержимое файла изменится хотя бы на один бит, то и контрольная сумма тоже изменится. Если никаких изменений не было, то браузер используется файл из кэша. При изменении файла изменится ссылка на него, и будет загружена новая версия. Получить контрольную сумму можно как с помощью стандартных утилит md5sum или sha1sum, так и с помощью специальных утилит.

Можно также добавлять к ссылках на файлы какой-нибудь произвольный набор символов, например, метку времени (http://example.com/script.js?timestamp_here) и обновлять ссылки при каждом деплое сайта. При использовании такого способа, однако, нет никакой гарантии того, что браузер не будет делать лишних запросов, т.к. даже на файлы, содержимое которых не менялось, будет вести уже другая ссылка (ключём кэширования выступает вся ссылка целиком вместе с query-параметрами) и браузер вынужден будет заново их скачивать.

Для HTML-страниц предпочтительнее устанавливать для заголовка Cache-

Control значение no-cache. Если нужно что-то срочно изменить на странице, а у клиента эта страница уже закэширована (современные браузеры делают это по умолчанию), то клиент может вообще не увидеть внесенных изменений.

Это особенно важно при использовании CDN, т. к. Большинство CDN по умолчанию кэшируют файлы без соответствующих заголовков на 24 часа. Можно, конечно, очистить кэш, но придется ещё ждать как минимум 15 минут после отправки соответствующего запроса. Установка же значения no-cache поможет избежать возможных проблем — страница всегда будет загружаться в актуальном виде. Браузеры все равно будут использовать заголовки If-None-Match (или If-Modified-Since), и неизмененная страница не будет загружаться лишний раз.

В некоторых случаях время кэширования HTML-страниц лучше указывать исходя из частоты изменений. Например, если страница с новостями на сайте обновляется каждый час, то для max-age можно установить значение секунд 3600 (1 час).

Вместо Cache-Control можно использовать заголовок Expires. В его значении указывается дата в виде RFC 1123 date format, по наступлении которой файл перестает быть актуальным (например: Tue, 31 Jan 2012 15:02:53 GMT). До наступления этой даты файл браузер не будет делать запросов к сайту, а файл будет забирать из кэша. После этой даты файл будет загружен снова.

Совет 3. Используйте сжатие gzip

С помощью сжатия можно существенно ускорить загрузку сайта. Начиная с HTTP/1.1 клиенты сообщают о поддерживаемых методах сжатия в заголовке Accept-Encoding:

```
Accept-Encoding: gzip, deflate
```

В ответе сервера информация об используемом методе сжатия передается в заголовке Content-Encoding:

```
Content-Encoding: gzip
```

Одним из самых популярных и наиболее часто используемых методов является, конечно же, gzip. За счет использования gzip можно существенно сократить время загрузки. Gzip особеннно эффективно работает с текстовыми файлами — HTML, CSS, JS. Благодаря сжатию размер текстовых файлов (и, соответственно, объем передаваемого трафика) уменьшается в среднем в 5–10 раз. Это позволяет значительно увеличить скорость загрузки страницы, что особенно актуально для мобильных клиентов с медленным соединением.

Для графических файлов использовать gzip смысла нет: сжатие не позволяет значительно снизить их размер, а часто даже увеличивает его.

Совет 4. Минифицируйте JS и CSS

Под минификацией понимается удаление лишних/необязательных символов из файла с целью уменьшения его размера и сокращения времени загрузки. Благодаря этому размер файлов уменьшается в среднем в 1,5÷3 раза. Сегодня широкое распространение получает практика минификации не только JS и CSS, но и других типов файлов (HTML, графических файлов и т. д.).

Для минификации используются специальные инструменты, в частности:

- UglifyJS 2;
- Google Closure Compiler;
- YUI Compressor;
- JSMin;
- Clean CSS;
- CSS Minifier;
- html-minifier;
- imagemin.

С их помощью можно убрать незначимые пробелы и переносы строк (в CSS и JS они опциональны), а иногда выполнять и более сложные операции. Например, в JS функцию вида:

```
function summ (first_param, second_param) {
return (first_param + second_param);
}
```

можно превратить в function s (a, b) {return (a+b)} и далее везде в коде использовать s вместо summ, при этом полностью сохранив логику её работы. Посмотреть, как работает процедура минификации JavaScript, можно на странице http://lisperator.net/uglifyjs/ в разделе Open Demo.

Совет 5. Используйте конкатенацию

Современные браузеры делают в среднем 6 параллельных запросов на домен. Если сайт содержит много файлов небольшого размера, время его загрузки может затянуться — особенно это заметно при медленном или нестабильном соединении. Здесь может помочь конкатенация — объединение нескольких файлов одного типа (например, JS или CSS) в один. Она позволяет уменьшить количество запросов и тем самым увеличить скорость загрузки страниц.

Конкатенация может также использоваться для ускорения загрузки изображений. Она может осуществляться двумя способами: с помощью внедрения данных в URL и с помощью спрайтов.

Внедрение данных осуществляется с помощью особого вида URL — data: URI. URI (Universal Resource Identifator) может использоваться как в атрибуте src тэга img, так и в URL фонового изображения в CSS.

Для конвертации изображений в data: URI существуют онлайн-инструменты.

Спрайтом называется коллекция изображений, объединенная в одну картинку. Для формирования сайтов используются различные программные инструменты. С помощью CSS можно обращаться к необходимому участку большого изображения и помещать его в нужное место на сайте.

Спрайты помогают увеличить скорость загрузки, но следует отметить, что работа с ними часто бывает сопряжена с затруднениями. Чтобы внести даже небольшое изменение в спрайт, потребуется вносить сопутствующие изменения и в CSS.

В современных инструментах для сборки проектов на JS (Brunch, Grunt, Gulp и других). Процедуры минификации и конкатенации можно автоматизировать. Чтобы при помощи одной команы выполнять все необходимые операции с файлами (включая итоговой деплой на сервер), достаточно создать небольшой конфигурационный файл, описывающий порядок и свойства сборки.

Single Page Applications

SPA как подход

Аббревиатура SPA означает «single page application» («одностраничное приложение»). В узком смысле она употребляется для обозначения одностраничного сайта, непосредственно выполняемого на стороне клиента в браузере. В более широком смысле SPA (иногда также употребляется аббревиатура SPI — «single page interface») означает целый подход в веб-разработке, который в наши дни получает все более широкое распространение. В чем заключается смысл этого подхода и почему он становится всё более популярным?

Как уже было отмечено выше, на сегодняшний день самым распространенным типом сайтов являются, конечно же, динамические сайты, в которых генерация страниц осуществляется на стороне сервера. При всех очевидных удобствах для разработчиков (каждый запрос создаёт страницу с чистого листа), такие сайты порой представляют собой настоящую головную боль для клиентов. Вот далеко не полный список неудобств, с которыми им приходится сталкиваться:

- Даже несмотря на то, что некоторые фреймворки позволяют вернуть в ответ форму с заполненными данными, введёнными клиентом в прошлый раз, необходимость перезагрузки страницы для валидации делает работу с сайтом неудобной.

- Генерация страниц на стороне сервера сопряжена с серьёзными нагрузками.

Часто описанные проблемы решаются так: страницы всё ещё генерируются на стороне сервера, но на стороне клиента выполняются небольшие JavaScript-сценарии, с помощью которых можно, например, выполнить валидацию формы до того, как она будет отправлена на сервер. Решение на первый взгляд логичное, но и у него есть недостатки:

- Если раньше функции фронтенда и бэкенда были чётко разделены (бэкенд отвечает за генерацию view и логику, а фронтенд за отображение), то теперь логика дублируется на фронтенде, что вряд ли можно назвать хорошей архитектурной практикой.

- Код, отвечающий за генерацию view, приходится постоянную дублировать и из-за этого возникают проблемы: «копипаста», расхождение разметки, поломанные селекторы, сложности в сопровождении кода и т. д.

Конечно, все эти проблемы решаемы. Но подход SPA во многих случаях оказывается гораздо более эффективным.

С точки зрения этого подхода сайт понимается не как набор страниц, а как набор состояний одной и той же HTML-страницы. При смене состояния происходит асинхронная подгрузка нового контента без перезагрузки самой страницы.

SPA представляет собой не сайт в классическом понимании, а приложение, которое исполняется на стороне клиента, в браузере. Поэтому с точки зрения пользователя проблем со скоростью работы почти не бывает даже при медленном или нестабильном соединении с Интернетом (например, при просмотре сайта с мобильного устройства). Высокая скорость работы обеспечивается ещё и благодаря тому, что с сервера на клиента приходит теперь не разметка, а данные (в основном в формате JSON), и размер их невелик.

В последнее время появилось немало интересных технологических решений и сервисов, значительно упрощающих создание и использование SPA. Рассмотрим некоторые из них более подробно.

BAAS (Backend as a Service)

Технологии, позволяющие отрисовывать интерфейс на стороне клиента и взаимодействовать с сервером только путём отправки запросов без перезагрузки страницы, существуют уже давно. Например, API XMLHttpRequest появился ещё в 2000 году. Но использование этих технологий было сопряжено с трудностями: нужно было переписывать бэкенд, реализующий функции авторизации запросов и доступа к данным.

Сегодня всё стало намного проще благодаря появлению многочисленных BaaS-сервисов. Аббревиатура BaaS означает Backend as a Service. BaaS-сервисы предоставляют разработчикам веб-приложений готовую серверную инфраструктуру (расположенную, как правило, в облаке).

Благодаря им можно сэкономить время (а зачастую ещё и деньги) на написание серверного кода и сосредоточиться на совершенствовании самого приложения и развития его функциональности. С помощью BaaS можно подключать к приложению или сайту любой бэкенд с любым набором функций — достаточно просто добавить на страницу соответствующую библиотеку.

В качестве примера можно привести сервис MongoLab, позволяющий подключать к веб-приложениям и сайтам облачную базу данных MongoDB.

Ещё один интересный пример — FireBase. Этот сервис представляет собой облачную базу данных и API для realtime-приложений. С его помощью можно, в частности, организовать обмен данными между клиентом и сервером в режиме реального времени: достаточно просто подключить на страницу JavaScript-библиотеку и настроить события на изменения данных. На основе FireBase легко реализовать, например, чат или ленту пользовательской активности.

В числе интересных и заслуживающих внимания BaaS-сервисов стоит также выделить:

- Backendless — платформа, предоставляющая готовую облачную серверную инфраструктуру для всех типов приложений. С её помощью на сайт можно добавить функциональность управления пользователями, хранения пользовательских данных, обмен сообщениями в режиме реального времени, рассылка уведомлений, геолокацию, управление файлами и много другое.

- Syncano — сервис, предоставляющий API для real-time приложений, близкий по функциональности к Firebase.

- QuickBlox — небезынтересный продукт российских разработчиков.

Наше облачное хранилище тоже может быть использовано в качестве внешнего бэкенда. С его помощью можно, например, интегрировать в веб-приложения и сайты функциональность по управлению файлами.

При использовании хранилища в качестве внешнего бэкенда следует обратить особое внимание на два ключевых момента: аутентификация и работа с контентом. Ключ аутентификации (токен) можно получить с помощью запроса к auth.selcdn.ru.

Затем его нужно будет указывать в качестве значения заголовка X-Auth-Token во всех запросах работы с данными. Ответы на эти запросы будут включать заголовок Access-Control-Allow-Origin со значением «*», дающий возможность обращаться к хранилищу с любого внешнего домена.

В качестве примера веб-приложения, использующего API хранилища, можно привести фотогалерею (см. также репозиторий на GitHub).

HTML5 History API

Каждая страница динамического сайта имеет собственный URL. SPA же устроены так, что в них можно изменять состояние страницы, не изменяя при этом URL. Но в таком случае возникает проблема: пользователь не сможет сохранить ссылку и затем вернуться к посещённому ранее разделу. Решить её можно несколькими способами.

Во-первых, можно использовать хэш-фрагмент (часть ссылки, которая идёт после символа «#»). В хэш-фрагмент помещается «виртуальный» адрес страницы, по которому можно восстановить прежнее состояние. Если пользователь перезагрузит страницу, то код на JavaScript, прочитав значение хэша, загрузит нужные данные и отобразит соответствующий ему раздел. Этот вариант используется на многих сайтах, но следует отметить, что подобные URL (например, http://example.com/base/#!/section1/page2) выглядят не очень естественно. Кроме того, они состоят из двух сущностей: «реальный» адрес, по которому страница запрашивается у веб-сервера и «виртуальный», который обозначает логический раздел.

Во-вторых, можно воспользоваться HTML5 History API. History API позволяет полностью изменять URL страницы в пределах текущего домена без её перезагрузки. В браузерах, поддерживающих этот API, нет никаких отличий между URL, использованных при загрузке страницы и URL, заданным из JavaScript.

Вызовы функций History API также связаны с нативными браузерными кнопками «Вперёд» и «Назад» и историей: при нажатии кнопки «Назад» браузер вызовет событие popstate, которое можно обработать в JavaScript-коде и отобразить предыдущий раздел. Посмотреть, как это работает, можно, например, здесь.

Чтобы сайт использовал описанную схему URL, нужно соответствующим образом настроить веб-сервер. Необходимо сделать так, чтобы вместо несуществующих страниц отдавалась главная страница SPA — обычно это index.html. В этом случае браузер клиента загрузит JavaScript, который уже отрисует нужный файл сайта исходя из текущего адреса страницы.

В настройках nginx это прописывается так:

```
location / {rewrite.* /index.html;}
```

Если SPA размещается на сервисах типа GitHub Pages или нашего хранилища, то настройки нужно устанавливать через панель управления.

Поисковая оптимизация

Индексация SPA поисковыми системами представляет собой отдельную проблему. Если сайт полностью генерируется на стороне клиента, то индексироваться он будет плохо, хотя в последнее время и наблюдаются улучшения в этом плане: боты постепенно начинают выполнять JavaScript при индексировании.

Одним из способов решения этой проблемы заключается в следующем: генерируются снапшоты страниц специально для ботов. Генерировать страницы следует по-разному в зависимости от того, используется ли HTML5 History API или нет.

В случае использования хэш-фрагментов Google и Yandex предлагают один и тот же подход: для разделения «реального» и «виртуального» адресов следует использовать «#!» вместо «#»; поисковые боты, увидев ссылку с таким разделителем, будут использовать специальный query-параметр «_escaped_fragment_» в URL и помещать в него часть адреса, следующую после после хэша.

Веб-сервер должен специальным образом обрабатывать такие запросы и отдавать полноценные HTML-страницы, чтобы боты могли их проиндексировать. (Потребность в этом возникает вследствии того, что часть URL после «#» по стандарту не является часть HTTP-запроса.) Рекомендуется также использовать специальный метатег, так как некоторые страницы могут не содержать ссылок с «#!».

Если же для адресации используются полноценные URL, то сделать сайт индексируемым ещё проще: достаточно генерировать соответствующую заданному адресу страницу на сервере. Бот, посетив сайт, получит страницу с содержимым и успешно её проиндексирует.

Если сайт открыт в браузере, то страница сначала будет отображена в соответствии с HTML-разметкой, а затем уже в дело вступит JavaScript; при дальнейших переходах по ссылкам страница перезагружаться не будет. Такое поведение можно реализовать, перехватывая события нажатия на ссылки и отменяя их нативное поведение. В таком случае, бот перейдет по ссылке (атрибут «href»), а мы сможем контролировать все переходы, обрабатывая событие так, как нам нужно.

Отметим также, что такой подход позволяет оптимизировать скорость загрузки сайта: современные браузеры намного быстрее отрисовывают приходящий с сервера HTML, а загрузка JavaScript-сценариев, их парсинг и отрисовка страницы через DOM занимает обычно больше времени.

Как сгенерировать HTML-страницу, имея в наличии только JavaScript-код, работающий напрямую с DOM? Это можно сделать несколькими способами.

Во-первых, можно установить PhantomJS (это так называемый «headless»

браузер на основе WebKit, управляемый через API) и настроить его на генерацию снапшотов страниц (см. практический пример здесь).

Во-вторых, можно воспользоваться инструментом prerender или сервисом Prerenderer.io, которые основаны также на PhantomJS.

В-третьих, в некоторых современных фреймворках (например, DerbyJS и React) эта функция уже реализована (см. также пример здесь: react-server-example).

Размещение SPA

В современном Интернете есть немало площадок, на которых можно размещать SPA. Рассмотрим наиболее популярные из них.

Dropbox

Самый простой вариант — размещение SPA в Dropbox. Для этого нужно создать в публичном каталоге Dropbox каталог для SPA, поместить в него все необходимые файлы, а затем выделить файл index.html и вызвать контекстное меню, нажав на правую кнопку мыши. В меню нужно выбрать пункт «Dropbox — Поделиться ссылкой». После этого будет создана ссылка и скопирована скопирована в буфер обмена, по которой приложение будет доступно через браузер.

В последнее время предпринимаются попытки расширить возможности по размещению сайтов в Dropbox. Однако главной проблемы они не решают: в Dropbox имеются довольно жёсткие (вплоть до полной блокировки) ограничения трафика по публичным ссылкам. Поэтому описываемый вариант можно рекомендовать разве что для демонстрации SPA друзьям и коллегам.

GitHub Pages

GitHub Pages представляет собой бесплатную площадку для размещения статических сайтов и SPA. Чтобы разместить SPA на GitHub Pages, нужно создать для него репозиторий на GitHub и поместить статические файлы в ветку gh-pages для Project Pages или в ветку master для User/Organization Pages. Все вносимые изменения нужно будет коммитить в этот репозиторий. Более подробная справка доступна в официальной документации.

Bit Balloon

Bit Balloon — сервис новый, но уже достаточно популярный.

Каждому зарегистрированному пользователю бесплатно предоставляется 10 Мб дискового пространства. Чтобы разместить на нём SPA, достаточно просто загрузить все необходимые файлы. Сервис автоматически минифицирует JS, HTML и CSS, а также подключит CDN.

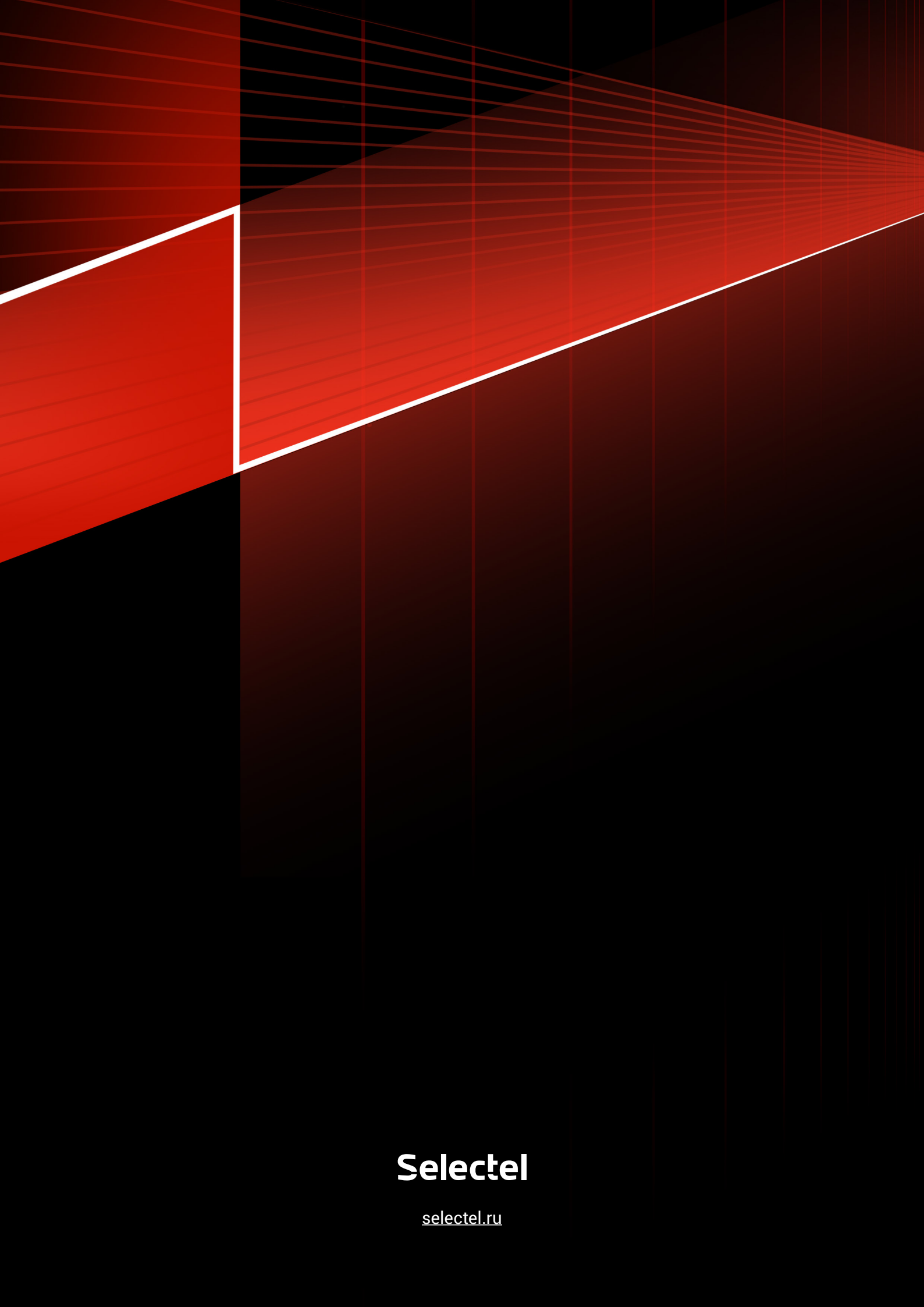
Премиум-аккаунт стоит 5$ в месяц. Для владельцев платных аккаунтов доступны различные дополнительные возможности — в частности, прикрепление собственных доментов и подключение разнообразных внешних сервисов.

Selectel Storage

Облачное хранилище Selectel представляет собой удобную площадку для хостинга SPA. Процедура размещения SPA включает следующие шаги:

- Создание публичного контейнера и прикрепление домена
- Настройка специальных страниц: процедура настройки совсем недавно была нами усовершенствована. Это позволяет использовать полноценные URL (см. раздел о History API)

К несомненным преимуществам хранилища «Селектел» перед другими площадками (в том числе и перед описанными выше) относятся низкая стоимость и гибкая система оплаты (никаких фиксированных тарифов; оплата взимается только за объём хранимых данных и исходящий трафик — посчитать можно здесь), а также наличие CDN и гибкой системы настроек параметров контейнеров и отдаваемых файлов.


Selectel
selectel.ru