Оглавление

[Анализ 2](#_Toc285742980)

[Концепция веб-сервисов 2](#_Toc285742981)

[Введение 2](#_Toc285742982)

[Основные положения модели веб-сервисов 3](#_Toc285742983)

[Серверная часть 6](#_Toc285742984)

[Анализ вариантов использования 6](#_Toc285742985)

[Выбор технологии реализации 8](#_Toc285742986)

[XML RPC в PHP 10](#_Toc285742987)

[CMS Joomla 11](#_Toc285742988)

[Реализация бизнес-логики на базе Joomla API 13](#_Toc285742989)

[Клиентская часть 19](#_Toc285742990)

[Анализ вариантов использования 19](#_Toc285742991)

[Выбор технологии реализации 20](#_Toc285742992)

[Обзор Adobe Lightroom 21](#_Toc285742993)

[Средства автоматизации Lightroom 22](#_Toc285742994)

[Реализация бизнес-логики на базе Lightroom API 23](#_Toc285742995)

[Взаимодействие клиента и сервера 25](#_Toc285742996)

[Общая схема взаимодействия 25](#_Toc285742997)

[Диаграмма последовательности 26](#_Toc285742998)

[Описание методов веб-сервиса 27](#_Toc285742999)

[Формат XML сообщений 28](#_Toc285743000)

[Реализация 30](#_Toc285743001)

[Серверная часть 30](#_Toc285743002)

[Классы модели 30](#_Toc285743003)

[Классы контроллеров 35](#_Toc285743004)

[Классы представлений 37](#_Toc285743005)

[Обеспечение безопасности 39](#_Toc285743006)

[Интерфейс пользователя 41](#_Toc285743007)

[Клиентская часть 43](#_Toc285743008)

[Publish Service 43](#_Toc285743009)

# Анализ

## Концепция веб-сервисов

### Введение

Сеть Интернет стала общепризнанным фактором деловой и общественной жизни. Широкая распространенность и возросшая пропускная способность создают условия, при которых выгодно решать многие задачи при помощи интернет-технологий.

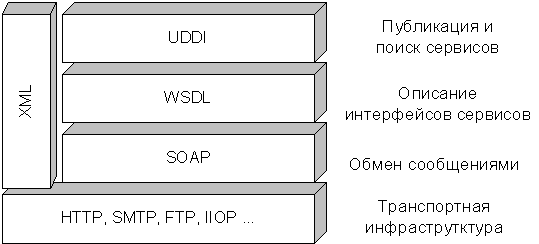
Однако Интернет объединяет в себе много различных платформ, а информация содержится в разнообразных источниках данных. Поэтому актуальна проблема связи таких разнородных данных, а также создания способа, который позволяет получать их в виде удобном для дальнейшей обработки.

Концепция веб-сервисов (Web Services) призвана решить эту задачу объединения, интеграции разнородных систем на основе открытых стандартов. Данная работа посвящена веб-сервисам, в ней кратко рассмотрены основные положения модели веб-сервисов, а также компоненты этой модели и технологии, используемые для их реализации. Практическая часть работы содержит небольшой пример, демонстрирующий разработку веб-сервиса и приложений использующих его. Пример основывается на реализации концепции веб-сервисов в рамках Java-технологий. Для понимания примера достаточно базовых знаний Java.

### Основные положения модели веб-сервисов

Веб-сервисы являются концепцией создания таких приложений, функции которых можно использовать при помощи стандартных протоколов Интернет. В настоящее время эту концепцию применяют и развивают многие ведущие компании в IT-области. Концепция веб-сервисов реализуется при помощи ряда технологий, которые стандартизованы World Wide Web Consortium (W3C).

Взаимосвязь этих технологий можно условно представить следующим образом.



Веб-сервисы являются одним из вариантов реализации компонентной архитектуры.

XML является фундаментом для создания большинства технологий, связанных с веб-сервисами.

Для удаленного взаимодействия с веб-сервисами используется Simple Object Access Protocol (SOAP). SOAP обеспечивает взаимодействие распределенных систем, независимо от объектной модели, операционной системы или языка программирования. Данные передаются в виде особых XML документов особого формата.

Согласно определению W3C, веб-сервисы это приложения, которые доступны по протоколам, которые являются стандартными для Интернет. Нет требования, чтобы веб-сервисы использовали какой-то определенный транспортный протокол. Спецификация SOAP определяет, каким образом связываются сообщения SOAP и транспортный протокол.

Наиболее часто реализуется передача SOAP сообщений по протоколу HTTP. Также широко распространено использование в качестве транспортного протокола SMTP, FTP, TCP.

Согласно определению W3C, "WSDL - формат XML для описания сетевых сервисов как набора конечных операций, работающих при помощи сообщений, содержащих документно-ориентированную или процедурно-ориентированную информацию". Документ WSDL полностью описывает интерфейс веб-сервиса с внешним миром. Он предоставляет информацию об услугах, которые можно получить, воспользовавшись методами сервиса, и способах обращения к этим методам.

Технология Universal Description, Discovery and Integration (UDDI) предполагает ведения реестра веб-сервисов. Подключившись к этому реестру, потребитель сможет найти веб-сервисы, которые наилучшим образом удовлетворяют его потребностям. Технология UDDI дает возможность поиска и публикации нужного сервиса, как человеком, так и программой-клиентом. Поиск и публикация в реестре предоставляется программе-клиенту как набор веб-сервисов реестра UDDI.

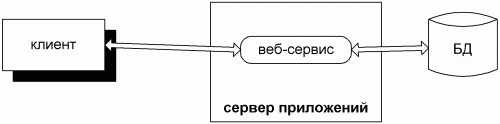
Веб-сервисы позиционируются как программное обеспечение промежуточного слоя. Использовать веб-сервисы могут как клиентские приложения, непосредственно работающие с пользователем, так и другие приложения (в том числе и другие веб-сервисы).

Веб-сервисы размещаются на серверах приложений.

Разработчики концепции веб-сервисов предлагают следующие сценарии применения веб-сервисов:

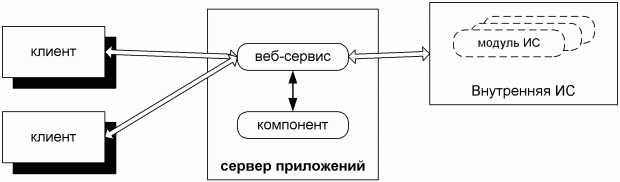
* Веб-сервисы как реализация логики приложения (бизнес-логики).

То есть, создание нового приложения бизнес-логика, которого реализуется в веб-сервисе



* Веб-сервисы как средство интеграции.

То есть, использование веб-сервиса как способа доступа удаленных клиентов к внутренней ИС компании, или для организации взаимодействия компонента (например, EJB, COM-компонента) с различными удаленными клиентами.



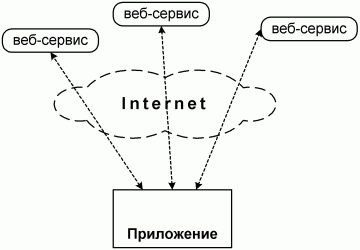
* Создание альтернативного сервиса.

В этом случае, при разработке нового веб-сервиса используется описание интерфейса уже существующего веб-сервиса. Таким образом, сервис имеет много потенциальных клиентов сразу с момента начала работы, а подключение к нему не требует существенных изменений на стороне клиента.

Как было сказано выше, концепция веб-сервисов включает в себя возможность ведения реестра веб-сервисов. Описание интерфейса может быть получено из такого реестра. После создания и внедрения нового веб-сервиса, имеет смысл зарегистрировать его в реестре. Тогда клиенты при поиске сервисов, реализующих исходный интерфейс, получат указание и на новый веб-сервис.

* Использование веб-сервиса как строительного блока при создании приложения.

Приложение может использовать веб-сервисы как удаленные компоненты, которые предоставляют определенную функциональность. Существуют различные сервисы, которые предоставляют качественное решение таких задач как аутентификация, ведение календаря, отправка сообщений, поиск, перевод и т. П

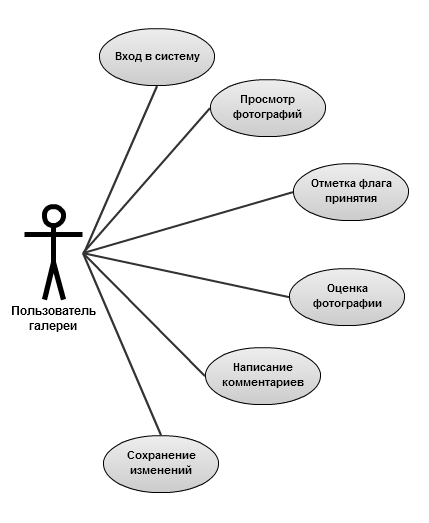


Кроме этого, и возможны другие варианты использования веб-сервисов. Например, существуют исследования по использованию веб-сервисов для построения распределенных вычислительных и информационных систем и одноранговых и со сложной иерархической структурой.

## Серверная часть

### Анализ вариантов использования

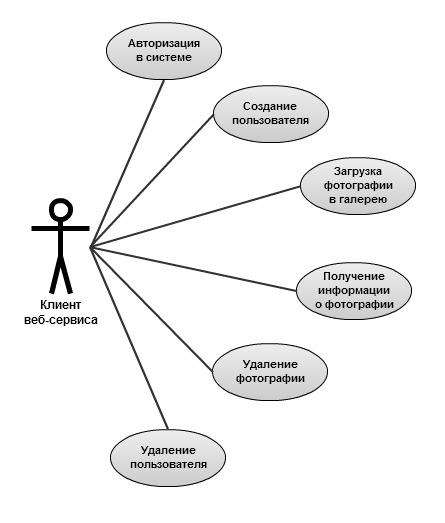
Серверная часть представляет собой объект для действий двух категорий объектов. Во-первых, это пользователи онлайн фотогалереи – именно она является фронтендом серверного программного обеспечения. Во-вторых, это клиентские системы, которые используют эту часть как веб-сервис. Исходя из этого, необходимо проанализировать две группы вариантов использования. На первой UseCase диаграмме изображены варианты использования системы пользователем фотогалереи.



Они включают в себя:

* вход в систему – пользователь проходит авторизацию и аутентификацию для определения списка фотографий, подлежащих просмотру и редактированию;
* просмотр фотографий – пользователь имеет возможность просмотра фотографий, опубликованных для него;
* отметка флага принятия – каждая фотография может быть принята / не принята пользователем, либо отметка может быть не проставлена. Этот признак позволяет фотографу отсеять ненужные снимки
* оценка фотографии – клиент проставляет оценку по 5-балльной шкале каждой фотографии. Это делается для того, чтобы выделить наиболее желаемые для дальнейшей обработки снимки;
* написание комментариев – здесь пользователь может оставить какие-либо пожелания для дальнейшей обработки фотографий;
* сохранение изменений – после того, как клиент проделал все необходимые манипуляции с опубликованными снимками, он сохраняет свои изменения, тем самым уведомляя фотографа об окончании процесса приёмки.

Вторая UseCase диаграмма иллюстрирует варианты использования серверной части системы в качестве веб-сервиса различными клиентами.



Она включает в себя:

* авторизация в системе – для совершения операций, клиентской системе необходимо авторизоваться при помощи пары логин-пароль;
* создание пользователя – в процессе публикации фотографий есть возможность создания нового пользователя фотогалереи с заданным именем и паролем;
* загрузка фотографии в галерею – публикация фотографий осуществляется по одной, при этом необходимо помимо самого графического файла указать пользователя, для которого выполняется публикация;
* получение информации о фотографии – запрос и получение флага принятия, рейтинга и комментариев для конкретной фотографии определенного пользователя;
* удаление фотографии – удаление опубликованной фотографии с сервера;

удаление пользователя – удаление пользователя и всех его фотографий с сервера.

### Выбор технологии реализации

#### PHP или ASP

При определении технологии реализации систем, работающих в веб, традиционно выбор останавливается на одной из двух технологий: APS и PHP.

**Active Server Pages**(ASP) - это технология Microsoft, позволяющая создавать динамические web страницы. ASP поддерживает несколько языков программирования, наиболее распространенным является VBScript (опять же разработка Microsoft). Соответственно, ASP - это технология для Windows систем.

**PHP** - аналог ASP, который может работать как в среде UNIX так и в Windows. PHP - это система разработки скриптов, включающая в себя CGI - интерфейс, интерпретатор языка и набор функций для доступа к базам данных и различным объектам WWW.

По мнению автора данной работы, PHP выигрывает у ASP как в плане технических (функциональных) возможностей, так и по критерию популярности/общедоступности.

* ASP технология основана на модели COM (COM - Component Object Model), то есть работать вы будете с объектами и компонентами. Например, для вывода информации по запросу клиента задействуется объект Response, при обращении к базе данных, задействуется еще один объект, для работы с файлами вызывается третий объект и т.д. Таким образом, подобное нагромождение замедляет работу ASP приложения. При использовании модулей PHP, работа идет в едином пространстве памяти PHP, и это несколько быстрее, чем использование разрозненных объектов, задействованных в различных процессах.
* PHP использует более оптимизированный механизм управления памятью, благодаря чему снижается нагрузка на веб-сервер
* Процесс разработки небольших приложений при помощи PHP занимает гораздо меньше времени
* ASP требует лицензирования веб-сервера, ОС и СУБД, PHP ;

Из этого можно сделать вывод, что для данного проекта целесообразно применять технологию PHP.

#### SOAP или XML RPC

В классической концепции веб-сервисов для передачи сообщений серверу используется SOAP – Simple Object Access Protocol — простой протокол доступа к объектам. Однако он используется не только для вызова удаленных процедур, но и для передачи произвольных сообщений в формате XML. XML-RPC (сокр. от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Extensible Markup Language Remote Procedure Call — XML-вызов удалённых процедур) — стандарт/[протокол](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [вызова удалённых процедур](http://ru.wikipedia.org/wiki/Remote_Procedure_Call), основанный на [XML](http://ru.wikipedia.org/wiki/XML), является прародителем [SOAP](http://ru.wikipedia.org/wiki/SOAP), отличается исключительной простотой применения. XML-RPC, как и любой другой интерфейс [RPC](http://ru.wikipedia.org/wiki/Remote_Procedure_Call), определяет набор стандартных типов данных и команд, которые программист может использовать для доступа к функциональности другой программы, находящейся на другом [компьютере](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) в [сети](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Следовательно, для вызова удаленных процедур на сервере без необходимости обмена произвольными сообщениями, XML RPC подходит лучше, благодаря простоте описания и реализации.

### XML RPC в PHP

Существует множество проектов Open Source, которые могут облегчить жизнь программиста, избавив от необходимости писать свой код для генерирования XML-RPC-запросов и ответов.

* *XMLRPC-EPI*, <http://xmlrpc-epi.sourceforge.net/>

XMLRPC-EPI изначально было разработан для внутреннего пользования и оказалось настолько успешным, что в настоящее время снабжает сам PHP экспериментальными XML-RPC функциями. По сути XMLRPC-EPI -это базовый класс, написанный в C++ (остальные написаны в PHP), а поэтому без пользователя root установить на сервер его не удастся: потребуется перекомпилировать PHP.

* *eZ xmlrpc*, <http://developer.ez.no/article/static/53/>

Разработка Бэрда Фастарда. Это XML-RPC класс, используемый в приложении ezPublish для получения запросов с локального клиента. Дизайн класса превосходен интуитивностью управления, кроме того, прекрасно описан в техническом руководстве, и позволяет проводить интроспекцию.

Всё прекрасно, но в одном класс нехорош: для его работы требуется, чтобы в конфигурации PHP был доступен и установлен **[xml parser](http://doc.trolltech.com/2.3/xml-dom.html%20qt-dom" \t "_blank)** (--with-qtdom). Для большинства инсталляций PHP это не типично, чаще встречается **[expat XML parser](http://sourceforge.net/projects/expat/" \t "_blank)** (--with-xml).

* *Fase 4 XML-RPC*, <http://www.fase4.com/xmlrpc/>
* *phpRPC*, <http://sourceforge.net/projects/phprpc/>

Класс с большим потенциалом. Разработчики не остановились на простой интерпретации и управлении запросами/ответами и предоставили пользователю более широкую функциональность, как, например, подключение к "абстрактной" базе данных. Последняя доступная версия - альфа 0.9,

* *phpxmlrpc*, <http://phpxmlrpc.sourceforge.net/>

Эта разработка корпорации Useful, самого создателя стандарта XML-RPC, очевидно содержит много полезного. По существу, приложение полностью поддерживает стандарт XML-RPC и, кроме того, предоставляет возможность отладки, которая (как мы увидим позже) может стать самой большой вашей проблемой при разработке web-сервиса.

Таким образом, существует множество реализаций технологии XML RPC в PHP. В используемой в данном проекте системе применяется разработка phpxmlrpc, о ней речь пойдет ниже.

### CMS Joomla

Большинство веб-проектов не создаются полностью с нуля, а используют уже существующие системы в качестве каркаса. Одним из видов таких систем является Система управления содержимым (контентом)([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) Content management system, CMS). Главной целью такой системы является возможность собирать в единое целое и объединить на основе ролей и задач все разнотипные источники знаний и информации, доступные как внутри организации, так и за ее пределами, а также возможность обеспечения взаимодействия сотрудников, рабочих групп и проектов с созданными ими базами знаний, информацией и данными так, чтобы их легко можно было найти, извлечь и повторно использовать привычным для пользователя образом. Однако, помимо функций по управлению содержимым, многие из таких систем предоставляют также развитое API для облегчения процесса разработки. Одной из таких CMS является Joomla – [система управления содержимым](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D1%8B%D0%BC), написанная на языках [PHP](http://ru.wikipedia.org/wiki/PHP) и [JavaScript](http://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript), использующая в качестве хранилища базу данных [MySQL](http://ru.wikipedia.org/wiki/MySQL" \o "MySQL). Является [свободным программным обеспечением](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), распространяемым под лицензией [GNU GPL](http://ru.wikipedia.org/wiki/GNU_General_Public_License). Её возможности:

* Функциональность можно расширять с помощью дополнительных расширений ([компонентов](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82), модулей и [плагинов](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)).
* Модуль безопасности для многоуровневой аутентификации пользователей и администраторов (используется собственный алгоритм аутентификации и "ведения" сессий).
* Система шаблонов позволяет легко изменять внешний вид сайта или создать свой уникальный.
* Настраиваемые схемы расположения модулей, включая левый, правый, центральный и любое другое произвольное положения блока (при желании содержимое модуля можно включить в содержимое материала. например выражение {loadmodule mod\_fpslideshow} (вместе с фигурными скобками) введенное в произвольное место в статье выведет содержимое модуля, которому задана позиция вывода как "mod\_fpslideshow").
* К преимуществам системы можно отнести то, что все компоненты, модули, плагины и шаблоны можно написать самому, разместить их в структурированном каталоге расширений или отредактировать существующее расширение по своему усмотрению.
* Для каждой динамической страницы можно создать своё описание и ключевые слова в целях повышения рейтинга в поисковых системах;
* Начало и окончание публикации любых материалов можно запрограммировать по календарю;
* Возможность ограничить доступ к определённым разделам сайта только для зарегистрированных пользователей а с выходом Joomla 1.6 доступ как к разделу так и к определённому материалу с точностью до конкретной связи материал<->пользователь;
* Настраиваемые схемы расположения элементов по областям шаблона
* Различные модули (последние новости, счётчик посещений, подробная статистика посещений, гостевая книга, форум и другие.);
* Возможность создания не одной, а нескольких форм обратной связи для каждого контакта;
* Модуль приёма от удалённых авторов новостей, статей и ссылок;
* Иерархия объектов;
* Менеджер рассылки новостей. Поддержка более чем 360 служб рассылки новостей по всему миру;
* Встроенный визуальный редактор [TinyMCE](http://ru.wikipedia.org/wiki/TinyMCE" \o "TinyMCE);
* [ЧПУ](http://ru.wikipedia.org/wiki/%D0%A7%D0%9F%D0%A3_(%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)) — «человекопонятный URL».

Именно возможность расширять стандартную функциональность при помощи подключаемых программных модулей, а также развитый API послужили причинами для выбора этой CMS в качестве основы для построения веб-сервиса.

### Реализация бизнес-логики на базе Joomla API

#### Программные модули

#### Поддержка MVC

#### Работа с БД

#### Отображение интерфейса пользователя

#### JavaScript и AJAX

#### Joomla и XML RPC

CMS Joomla использует проект phpxmlrpc в качестве базы при работе с XML RPC. При этом запросы отправляются по адресу http://site.ru/service/publish. Обработчик сообщений разбирает XML сообщения и

## Клиентская часть

### C:\Documents and Settings\Артем\Рабочий стол\Реализация\Пользователь lightroom.pngАнализ вариантов использования

### Выбор технологии реализации

### Обзор Adobe Lightroom

### Средства автоматизации Lightroom

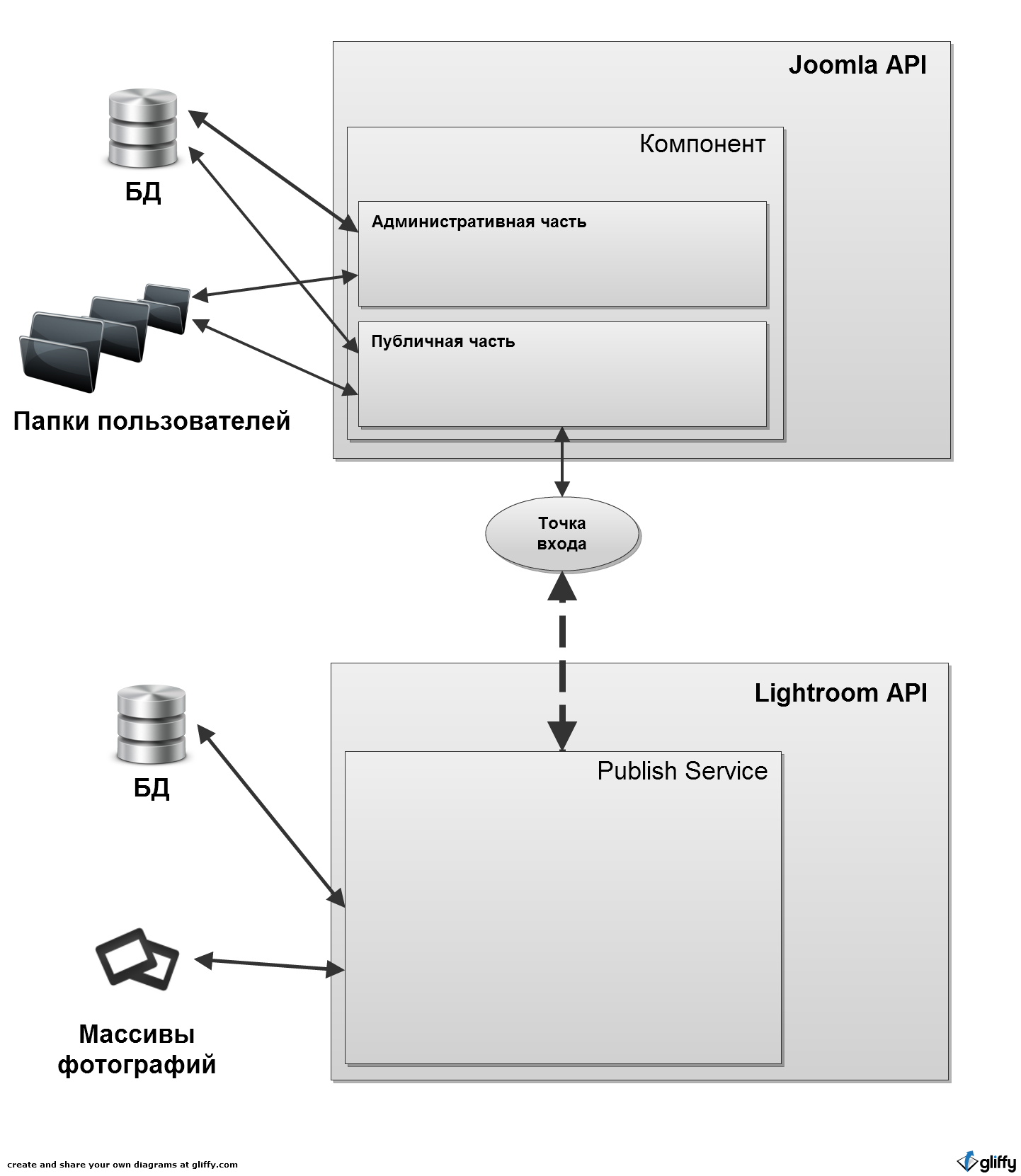
### Реализация бизнес-логики на базе Lightroom API

#### Программные модули

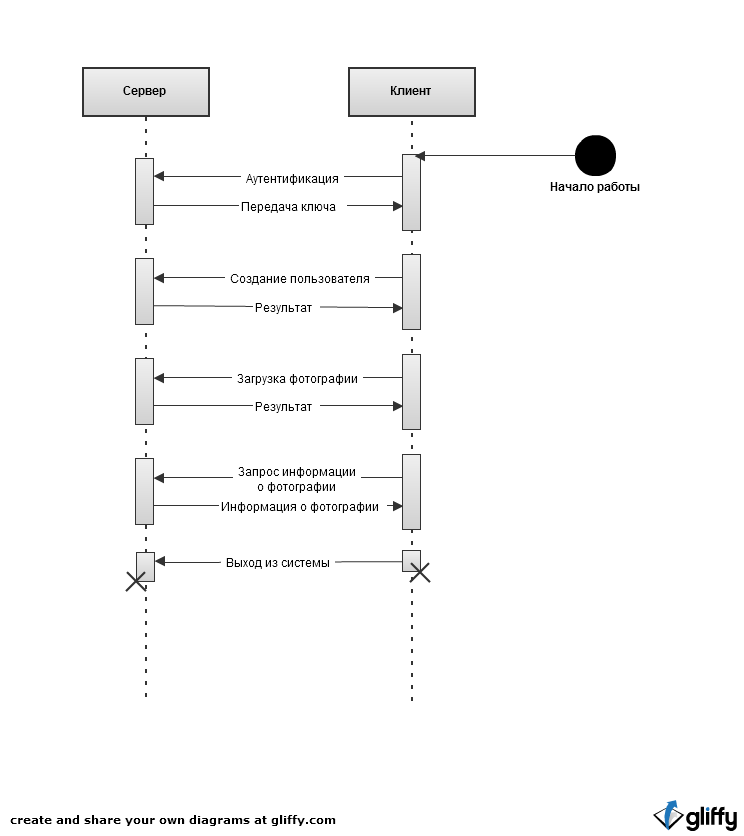
#### Работа с массивами фотографий

## Взаимодействие клиента и сервера

### Общая схема взаимодействия



### Диаграмма последовательности



### Описание методов веб-сервиса

В соответствии с диаграммой вариантов использования, веб-сервис публикации фотографий имеет следующие методы:

* login(login, password)

Метод выполняет вход в систему с использованием имени пользователя и пароля. Метод возвращает секретный токен, используемый для дальнейших операций с сервером и сообщение об ошибке (если таковая произошла);

* createUser(username, password, token)

Метод выполняет создание пользователя с указанным именем и паролем. Также на сервере создается папка с именем, равным имени пользователя, для хранения фотографий. Метод возвращает статус операции создания и сообщение об ошибке (если таковая произошла);

* uploadPhoto(username, photo\_name, photo\_file, token)

Метод выполняет публикацию фотографии для указанного пользователя; указывается название фотографии и передаётся файл с изображением. Метод возвращает идентификатор фотографии на сервере, а также статус операции и сообщение об ошибке (если таковая произошла);

* getPhotoInfo(username, photo\_name, token)

getPhotoInfo(photo\_id, token)

Методы выполняют запрос на получение информации о фотографии, которая определяется по серверному идентификатору, либо по комбинации имени пользователя и названия фотографии. Метод возвращает информацию о запрошенном изображении, а также статус операции и сообщение об ошибке (если таковая произошла);

* deletePhoto(username, photo\_name, token)

deletePhoto(photo\_id, token)

Методы удаляют фотографию на сервере, которая определяется по серверному идентификатору, либо по комбинации имени пользователя и названия фотографии. Метод возвращает статус операции удаления и сообщение об ошибке (если таковая произошла);

* deleteUser(username, token)

Метод удаляет указанного пользователя на сервере, а также все его фотографии. Метод возвращает статус операции и сообщение об ошибке (если таковая произошла);

* logout(token)

Метод выполняет выход из системы. Дальнейшая работа с указанным токеном становится невозможной; для повторных запросов необходимо заново авторизоваться на веб-сервисе.

### Формат XML сообщений

XML сообщения, используемые для удаленного вызова процедур сервиса, имеют следующий формат:

<?xml version="1.0"?>

<methodCall>

<methodName>getPhotoInfo</methodName>

<params>

<param>

<value>

<username>ivanov</username>

</value>

</param>

<param>

<value>

<photo\_name>Portrait001</photo\_name>

</value>

</param>

<param>

<value>

<token>djk32i38dsjdk</token>

</value>

</param>

</params>

</methodCall>

XML сообщения, используемые для ответа сервера на вызов процедуры, имеют следующий формат:

<?xml version="1.0"?>

<methodResponse>

<params>

<param>

<value>

<accepted\_flag>true</accepted\_flag>

</value>

</param>

<param>

<value>

<rating>5</rating>

</value>

</param>

<param>

<value>

<comment>bla bla bla</comment>

</value>

</param>

</params>

<errors>

<error>

<code>0</code>

<message>OK</message>

</error>

</errors>

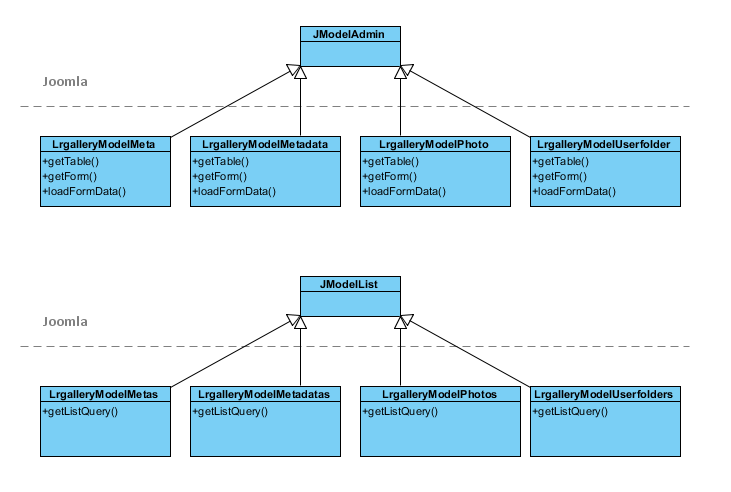
</methodResponse>

# Реализация

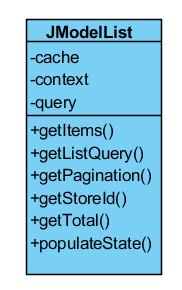
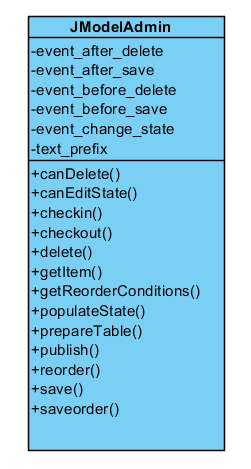
## Серверная часть

### Классы модели

Административная часть:



Все классы моделей административной части наследуются от базовых классов JModelAdmin и JModelList:



Большинство операций выполняются при помощи методов базовых классов и суперклассов. Есть всего несколько методов, которые требуют определения в наследуемых классах.

* LrgalleryModelMeta

Класс модели для работы с полями метаданных.

Методы:

* public function getTable($type = 'meta', $prefix = 'lrgalleryTable',

$config = array())

Возвращает экземпляр класса JTable для хранения данных о полях метаданных

* public function getForm($data = array(), $loadData = true)

Возвращает экземпляр класса JForm для хранения шаблона отображения полей данных о полях метаданных

* protected function loadFormData()

Выполняет заполнение полей формы данными

* LrgalleryModelMetadata

Класс модели для работы со значениями полей метаданных.

Методы:

* public function getTable($type = 'meta', $prefix = 'lrgalleryTable',

$config = array())

Возвращает экземпляр класса JTable для хранения данных о значениях полей метаданных

* public function getForm($data = array(), $loadData = true)

Возвращает экземпляр класса JForm для хранения шаблона отображения полей данных о значениях полей метаданных

* protected function loadFormData()

Выполняет заполнение полей формы данными

* LrgalleryModelPhoto

Класс модели для работы с фотографиями.

Методы:

* public function getTable($type = 'meta', $prefix = 'lrgalleryTable',

$config = array())

Возвращает экземпляр класса JTable для хранения данных о фотографиях

* public function getForm($data = array(), $loadData = true)

Возвращает экземпляр класса JForm для хранения шаблона отображения полей данных о фотографиях

* protected function loadFormData()

Выполняет заполнение полей формы данными

* LrgalleryModelUserfolder

Класс модели для работы с папками пользователей.

Методы:

* public function getTable($type = 'meta', $prefix = 'lrgalleryTable',

$config = array())

Возвращает экземпляр класса JTable для хранения данных о папках пользователей

* public function getForm($data = array(), $loadData = true)

Возвращает экземпляр класса JForm для хранения шаблона отображения полей данных о папках пользователей

* protected function loadFormData()

Выполняет заполнение полей формы данными

* LrgalleryModelMetas

Класс модели для работы со списком полей метаданных.

Методы:

* protected function getListQuery()

Возвращает экземпляр класса JQuery для выполнения запроса для получения списка полей метаданных

* LrgalleryModelMetadatas

Класс модели для работы со списком значений полей метаданных.

Методы:

* protected function getListQuery()

Возвращает экземпляр класса JQuery для выполнения запроса для получения списка значений полей метаданных

* LrgalleryModelPhotos

Класс модели для работы со списком фотографий.

Методы:

* protected function getListQuery()

Возвращает экземпляр класса JQuery для выполнения запроса для получения списка фотографий

* LrgalleryModelUserfoldes

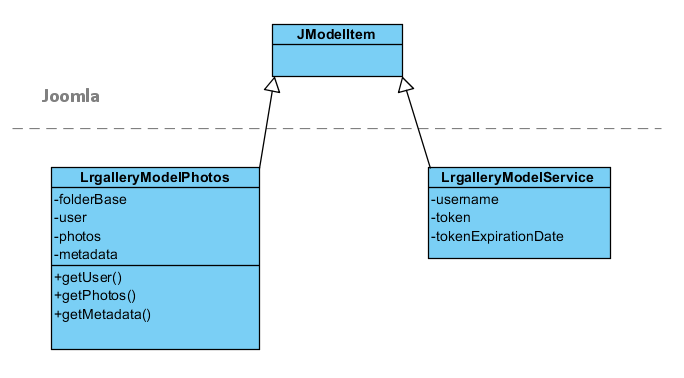
Класс модели для работы со списком папок пользователей.

Методы:

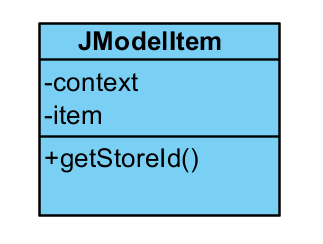
* protected function getListQuery()

Возвращает экземпляр класса JQuery для выполнения запроса для получения списка папок пользователей

Публичная часть:



Все классы моделей публичной части наследуются от базового класса JModelItem:



Некоторые операции выполняются при помощи методов базовых классов и суперклассов. Однако большинство методов требуют определения в наследуемых классах.

* LrgalleryModelPhotos

Класс модели для основной работы с интерфейсом пользователя, отображения фотографий и данных о них.

Поля:

* const folderBase = 'media/user\_folders'

Базовый путь для папок с пользовательскими фотографиями

* protected $user

Текущий пользователь

* protected $photos

Фотографии текущего пользователя

* protected $metadata

Значения полей метаданных для фотографий текущего пользователя

Методы:

* public function getUser()

Возвращает текущего пользователя

* public function getPhotos()

Возвращает фотографии текущего пользователя

* public function getMetadata()

Возвращает значения полей метаданных для фотографий текущего пользователя

* LrgalleryModelService

Класс модели для работы с обработкой запросов клиентов веб-сервиса.

Поля:

* protected $username

Хранит текущего пользователя

* protected $token

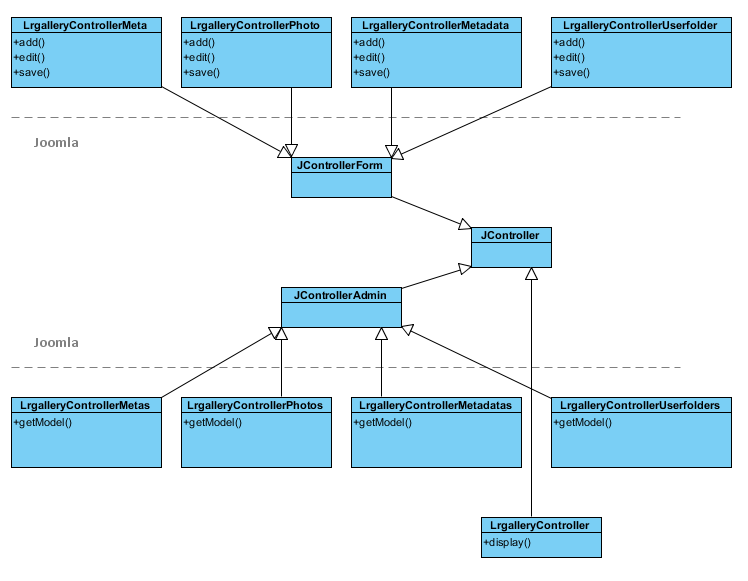
Хранит токен текущей аутентификации

* protected $tokenExpirationDate

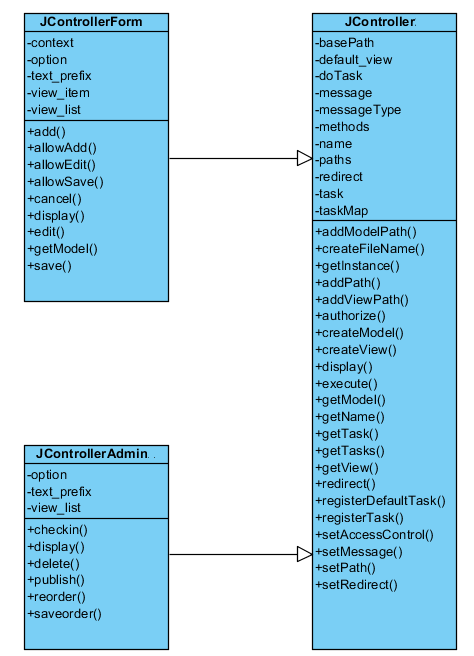
Хранит дату окончания срока действия токена

### Классы контроллеров

Административная часть:



Все пользовательские классы контроллеров наследуются от базовых классов JController, JControllerForm и JControllerAdmin:



Большинство операций выполняются при помощи методов базовых классов и суперклассов. Есть всего несколько методов, которые требуют переопределения в наследуемых классах.

* LrgalleryController

Класс основного контроллера для координации показа представлений и создания моделей.

Методы:

* public function display($cachable = false)

Управляет начальным отображением view и координирует действия остальных контроллеров

* LrgalleryControllerMeta

Класс контроллера для работы с полями метаданных.

Методы:

* public function add()

Метод для добавления нового поля метаданных

* public function edit($key = null, $urlVar = null)

Метод для редактирования поля метаданных

* protected function save($key = null, $urlVar = null)
* Метод для сохранения поля метаданных
* LrgalleryControllerMetadata

Класс контроллера для работы со значениями полей метаданных.

Методы:

* public function add()

Метод для добавления нового значения поля метаданных

* public function edit($key = null, $urlVar = null)

Метод для редактирования значения поля метаданных

* protected function save($key = null, $urlVar = null)
* Метод для сохранения значения поля метаданных
* LrgalleryControllerPhoto

Класс контроллера для работы с фотографиями.

Методы:

* public function add()

Метод для добавления новой фотографии

* public function edit($key = null, $urlVar = null)

Метод для редактирования фотографии

* protected function save($key = null, $urlVar = null)
* Метод для сохранения фотографии
* LrgalleryControllerUserfolder

Класс контроллера для работы с папками пользователей.

Методы:

* public function add()

Метод для добавления новой папки пользователя

* public function edit($key = null, $urlVar = null)

Метод для редактирования папки пользователя

* protected function save($key = null, $urlVar = null)
* Метод для сохранения значения папки пользователя
* LrgalleryControllerMetas

Класс контроллера для работы со списком полей метаданных.

Методы:

* public function getModel()

Возвращает экземпляр класса JModel, используемый при работе с данными

* LrgalleryControllerMetadatas

Класс контроллера для работы со списком значений полей метаданных.

Методы:

* public function getModel()

Возвращает экземпляр класса JModel, используемый при работе с данными

* LrgalleryControllerPhotos

Класс контроллера для работы со списком фотографий.

Методы:

* public function getModel()

Возвращает экземпляр класса JModel, используемый при работе с данными

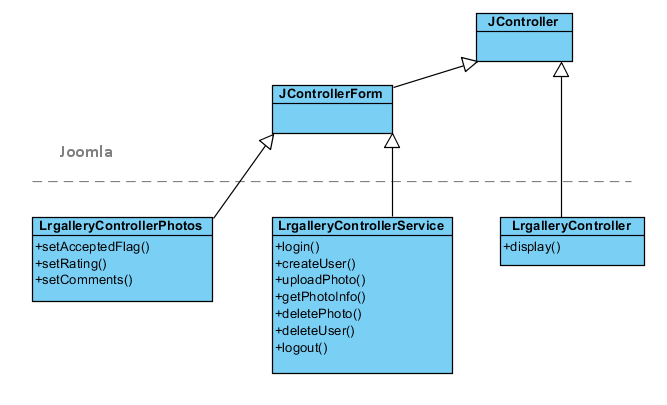
* LrgalleryControllerUserfolders

Класс контроллера для работы со списком папок пользователей.

Методы:

* public function getModel()

Возвращает экземпляр класса JModel, используемый при работе с данными

Публичная часть:

Некоторые операции выполняются при помощи методов базовых классов и суперклассов. Однако большинство методов требуют определения в наследуемых классах.

* LrgalleryController

Класс основного контроллера для координации показа представлений и создания моделей.

Методы:

* public function display($cachable = false)

Управляет начальным отображением view и координирует действия остальных контроллеров

* LrgalleryControllerPhotos

Класс контроллера для основной работы с интерфейсом пользователя, отображения фотографий и данных о них.

Методы:

* public function setAcceptedFlag($flag)

Устанавливает флаг принятия фотографии

* public function setRating($rating)

Устанавливает рейтинг фотографии

* public function setComments($comments)

Устанавливает комментарии к фотографии

* LrgalleryControllerService

Класс контроллера для обработки удаленных запросов клиентов веб-сервиса.

Методы:

* public function login($username, $password)

Метод выполняет вход в систему с использованием имени пользователя и пароля. Метод возвращает секретный токен, используемый для дальнейших операций с сервером и сообщение об ошибке (если таковая произошла);

* public function createUser($username, $password, $token)

Метод выполняет создание пользователя с указанным именем и паролем. Также на сервере создается папка с именем, равным имени пользователя, для хранения фотографий. Метод возвращает статус операции создания и сообщение об ошибке (если таковая произошла)

* public function uploadPhoto($username, $photo\_name, $photo\_file,

$token)

Метод выполняет публикацию фотографии для указанного пользователя; указывается название фотографии и передаётся файл с изображением. Метод возвращает идентификатор фотографии на сервере, а также статус операции и сообщение об ошибке (если таковая произошла);

* public function getPhotoInfo($username, $photo\_name, $token)

public function getPhotoInfo($photo\_id, $token)

Методы обрабатывают запрос на получение информации о фотографии, которая определяется по серверному идентификатору, либо по комбинации имени пользователя и названия фотографии. Метод возвращает информацию о запрошенном изображении, а также статус операции и сообщение об ошибке (если таковая произошла)

* public function deletePhoto($username, $photo\_name, $token)

public function deletePhoto($photo\_id, $token)

Методы удаляют фотографию на сервере, которая определяется по серверному идентификатору, либо по комбинации имени пользователя и названия фотографии. Метод возвращает статус операции удаления и сообщение об ошибке (если таковая произошла)

* public function deleteUser($username, $token)

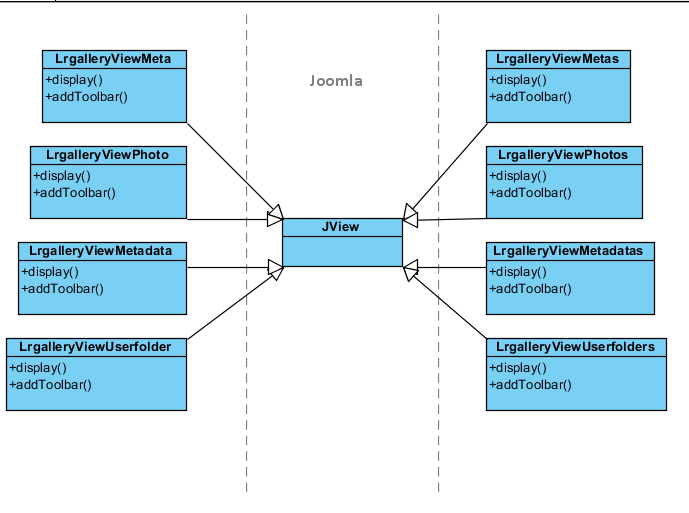
Метод удаляет указанного пользователя на сервере, а также все его фотографии. Метод возвращает статус операции и сообщение об ошибке (если таковая произошла)

* public function logout($token)

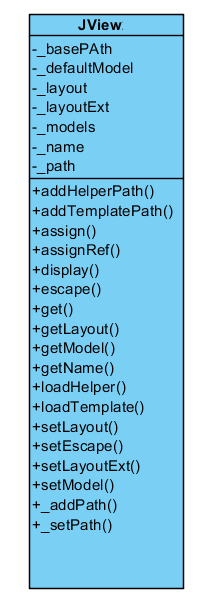
Метод выполняет выход из системы. Дальнейшая работа с указанным токеном становится невозможной; для повторных запросов необходимо заново авторизоваться на веб-сервисе

### Классы представлений

Административная часть:



Все классы представлений административной части наследуются от базового класса JView:



Большинство операций выполняются при помощи методов базовых классов и суперклассов. Есть всего несколько методов, которые требуют определения в наследуемых классах.

* LrgalleryViewMeta

Класс представления для работы с полями метаданных.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewMetadata

Класс представления для работы со значениями полей метаданных.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewPhoto

Класс представления для работы с фотографиями.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewUserfolder

Класс представления для работы с папками пользователей.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewMetas

Класс представления для работы со списками полей метаданных.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewMetadatas

Класс представления для работы со списками значений полей метаданных.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewPhotos

Класс представления для работы со списками фотографий.

Методы:

* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

* LrgalleryViewUserfolders

Класс представления для работы со списками папок пользователей.

Методы:

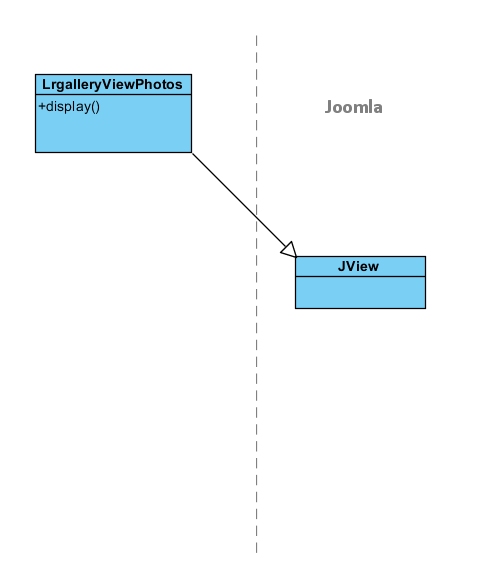
* public function function display($tpl = null)

Метод для отображения view

* public function addToolbar()

Метод для добавления панели инструментов в административном интерфейсе

Публичная часть:



Все классы представлений публичной части наследуются от базового класса JView, описание которого приведено выше.

* LrgalleryViewPhotos

Класс представления для основной работы с интерфейсом пользователя, отображения фотографий и данных о них.

Методы:

* public function function display($tpl = null)

Метод для отображения шаблона view

### Обеспечение безопасности

#### Защита коммуникаций между сервером и клиентом на основе токенов

При совершении операции входа в систему клиентом, в ответном сообщении сервер сообщает токен, соответствующий текущей сессии. Эта строка является необходимой для аутентификации всех последующих действий клиента и должна передаваться в качестве параметры с каждым сообщением XML RPC. Выданный токен хранится в базе данных сервера в течение всего периода общения с клиентом и используется для его авторизации.Время действия токена ограничено; каждый удаленный вызов процедуры с использованием токена продлевает его действие. Таким образом, если клиент в течение определенного времени не производит никаких действий по отношению к веб-сервису, срок действия токена истекает, и следующая операция требует повторной аутентификации.

#### Парольная защита папок пользователей на сервере

Для защиты папок пользователей на сервере используется механизм .htaccess.

1. Подготовка файла с паролями

Для подготовки используется утилита htpasswd. Для генерации паролей в командной строке даются следующие команды:

* + Для первого пароля (логин - yyy, пароль - zzz):   
    htpasswd.exe -bc .htpasswd yyy zzz  
    При этом будет создан файл .htpasswd.
  + Для добавления паролей в файл:  
    htpasswd.exe -b .htpasswd xxx zzz  
    При этом в файл .htpasswd будут дописаны пароли.

После этого файл можно открыть в текстовом редакторе и убедиться, что там есть строчки, соответствующие пользователям, которые были добавлены. Пример файла:

yyy:$apr1$XE1.....$U22NqaiWxdRURmPcwEAmS.  
xxx:$apr1$2K1.....$x.wbKD7PCD.3EgbcsXA8d/

1. Защита папки

Необходимо создать файл .htaccess примерно следующего содержания:

AuthUserFile /home/username/www/users/ivanov/ .htpasswd    
AuthName Admin  
AuthType Basic  
Require valid-user

Здесь первая строка - путь до файла .htpasswd. Часть /ivanov/ соответствует папке, которую нужно закрывать. Путь перед ней - это абсолютный путь к сайту.

Вторая строчка - имя блока авторизации, тут нужно ввести имя пользователя.После этого файлы .htpasswd и .htaccess необходимо поместить в папку ivanov.

1. Проверка авторизации

Здесь важно не путать отказ в просмотре каталога из-за отсутствия индексного файла и отказ авторизации - сообщения выдаются похожие, но всё же немного разные.

### Интерфейс пользователя

#### Публичная часть

#### Административная часть

## Клиентская часть

### Publish Service