МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

СОГЛАСОВАНО
Генеральный директор
ЗАО «АйТи»

_____ Бакиев О.Р.
"29" декабря 2011 г.

УТВЕРЖДАЮ
Ректор НИУ ИТМО

_____ Васильев В.Н.
"29" декабря 2011 г.

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ХРАНЕНИЯ ДАННЫХ CLAVIRE/STORAGE

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 37-ЛУ

Представители
Организации-разработчика

Руководитель разработки,
профессор НИУ ИТМО

_____ Бухановский А.В.
"29" декабря 2011 г.

Ответственный исполнитель,
с.н.с. НИУ ИТМО

_____ Луценко А.Е.
"29" декабря 2011 г.

Нормоконтролер
ведущий инженер НИУ ИТМО

_____ Позднякова Л.Г.
"29" декабря 2011 г.

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

2011

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 37-ЛУ

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ХРАНЕНИЯ ДАННЫХ CLAVIRE/STORAGE

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 37

ЛИСТОВ 19

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

Документ содержит описание программного компонента хранения данных CLAVIRE/Storage RU.СНАБ.80066-06 01 37, обеспечивающего работу с распределенными хранилищам данных в процессе функционирования МИТП CLAVIRE. Программный компонент хранения данных разработан в ходе выполнения проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» (Договор № 21057 от 15 июля 2010 г., шифр 2010-218-01-209) в рамках реализации постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства».

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Компонент хранения данных CLAVIRE/Storage RU.СНАБ.80066-06 01 37 предназначен для организации работы с распределенными хранилищами данных, полученных в процессе функционирования компонентов многопрофильной инструментально-технологической платформы (МИТП) CLAVIRE в распределенной среде. Данный компонент разработан на языке Java и представляет собой два отдельных модуля: модуль хранения и модуль агента.

Компонент способен функционировать на любых аппаратных платформах, имеющих Java Virtual Machine (JVM) и способных исполнять байт-код Java. Для его работы необходимо наличие JVM версии не ниже 1.6. На хосте с модулем хранения должна быть установлена СУБД MongoDB версии не ниже 2.0.0.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программный компонент хранения данных предназначен для:

- предоставления возможности распределенного хранения данных, образующих в результате функционирования других компонентов МИТП CLAVIRE;
- реализации возможности резервирования и отката к предыдущим версиям файлов, сохраненных в компоненте, при помощи функции контроля версий.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Принципы функционирования

Функционирование программного компонента хранения данных определено характером компонента. В нем выделены модуль агента и модуль хранения. Обмен управляющей информацией между модулями и клиентами компонента производится посредством формирования сообщений в формате JSON.

Работа клиентов компонента хранения данных, которыми могут являться другие компоненты МИТП CLAVIRE, а также обособленные приложения, осуществляется посредством обмена управляющими сообщениями JSON с хранилищем и запросов по работе с файлами непосредственно с агентами по протоколу HTTP. Все взаимодействия

как внутри компонента, так и с другими компонентами, производятся с использованием REST-интерфейса.

Файлы распределены между агентами хранения, для работы с ними клиент компонента хранения данных по протоколу HTTP, используя методы PUT, GET, DELETE и информацию об идентификаторе (id) файла или его имени, может совершать необходимые операции. В запросе обязательно указываются следующие параметры:

1) действие, которое необходимо совершить над файлом (action);
2) имя файла либо его id (file_name, file_id);
3) размер файла для операции put (file_size).

Полученная от модуля хранения информация в сообщении JSON при этом будет иметь URL для подключения к агенту для выполнения необходимой операции. Возможно получение метаданных о файле из хранилища путем формирования соответствующего JSON-запроса.

Добавление нового агента к хранилищу осуществляется путем создания JSON-сообщения о регистрации, которое содержит сетевой адресе агента, порт и квоту на размещение файлов компонента.

Мониторинг работоспособности агентов хранения производится путем постоянного их опроса по протоколу HTTP. В ответ на запрос по URL /agent_status агент хранения выдает JSON-сообщение, содержащее параметры агента, а также текущее значение доступного места. Проверка выполняется отдельным потоком модуля хранения с периодом, который задается в конфигурационном файле. Недоступный агент хранения удаляется из базы доступных агентов.

3.2. Программная архитектура

Концептуальная схема взаимодействия модулей хранилища представлена на рис. 3.1. В качестве составных частей выступают агент хранения данных, модуль хранения, база данных и клиент, в качестве которого могут выступают другие компоненты МИТП CLAVIRE, а также сторонние программные продукты.

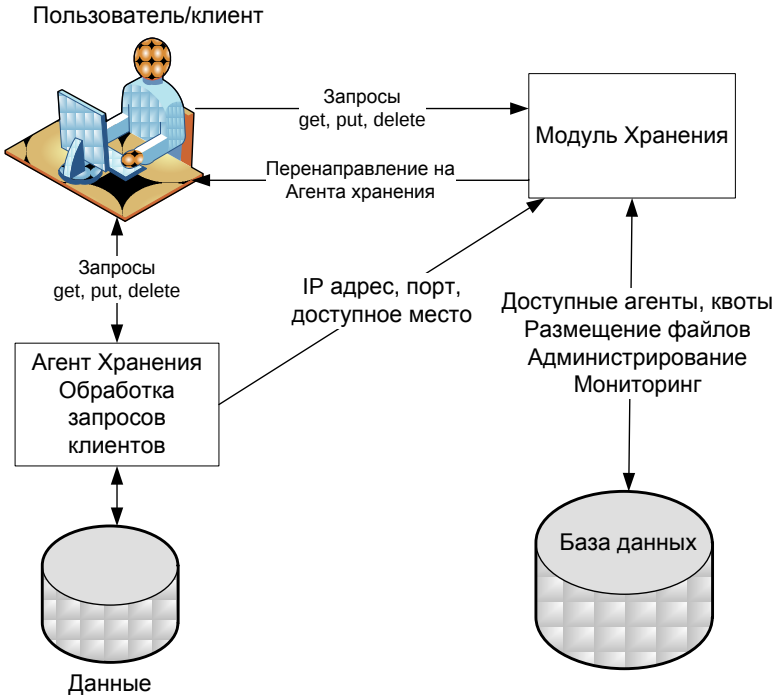

Рисунок 3.1 – Высокоуровневая структура программного компонента хранения данных

Агент хранения – программное обеспечение, выполняющееся на узле, предоставляющем свои ресурсы (дисковое пространство), в использование в рамках платформы облачных вычислений. Функция по хранению файлов является основной для агентов хранения, поэтому программный код агента позволяет управлять хранением файлов при помощи операций создания, получения и удаления файлов. Данные методы в виде интерфейса REST предоставляются клиентам как способ работы с файлами. Для использования ресурсов узла агент хранения взаимодействует с модулем хранения и представляет всю необходимую для работы хранилища информацию. Доступность агента проверяется модулем хранения путем выполнения периодических запросов к агенту о размере свободного дискового пространства. Файлы хранятся на агентах в виде плоской структуры, для отображения общей структуры каталогов используется информация, хранящаяся в базе данных модуля хранения. Такая схема введена вследствие того, что хранить на каждом агенте структуру каталогов нерационально. Поддержка этой структуры будет являться дополнительной нагрузкой как на сам агент, так и на модуль хранения.

Таким образом, агент хранения выполняет следующие функции:

1) взаимодействие с модулем хранения;
2) взаимодействие с клиентом;

3) хранение файлов;
4) создание, получение и удаление файлов.

Модуль хранения обеспечивает диспетчеризацию потоков данных внутри распределенного хранилища файлов. Управляя работой остальных компонентов, модуль хранения не является узким местом всего хранилища благодаря работе в режиме предоставления только управляющей информации, без необходимости обработки потока данных, который проходит между отправителями и получателями файлов (клиенты и агенты). Для обеспечения распределенного хранения файлов модуль хранения взаимодействует с агентами хранения и поддерживает в актуальном состоянии базу данных местоположений файлов, а также другую служебную информацию (версии файлов, дату и время их создания, размер и т.д.). Логическая структура каталогов целиком хранится в базе данных и работа с ней осуществляется исключительно посредством модуля хранения. Первоначальной запрос от клиентов на операции с файлами обрабатывается модулем хранения, после получения необходимой информации по запросу он либо сразу обрабатывается (как в случае получения метаданных), либо клиенту возвращается информация о дальнейших действиях по завершению запроса (адрес агента хранения с параметрами запроса к нему). Для обеспечения надежности работы всего компонента модуль хранения отслеживает доступность агентов хранения. Также в блоке хранения присутствует возможность получения служебной информации относительно работы всего хранилища, а также, при необходимости, – внесения изменений. Модуль хранения ответствен за распределение нагрузки между агентами при операциях сохранения файлов. Распределение осуществляется по схеме Round-robin. Такое поведение позволяет также распределять нагрузку и при получении файлов, хотя и менее равномерно.

В процессе своего функционирования модуль выполняет следующие функции:

1) регистрирует агенты хранения;
2) отслеживает доступность агентов и размер свободного места на них;
3) распределяет нагрузку между агентами по алгоритму Round-robin;
4) следит за целостностью базы метаданных;
5) принимает и обрабатывает запросы от клиентов;
6) обеспечивает хранение версий файлов для возможности отката исправлений для результатов работы вычислительных сервисов обработки данных;
7) предоставляет интерфейс для администрирования и мониторинга системы.

База данных предназначена для хранения метаданных модуля хранения. Реализация базы данных выполнена с учетом требований быстродействия и возможности простого горизонтального масштабирования. Хранение структуры в единой БД помогает снизить накладные расходы при изменении структуры в распределенной системе хранения файлов и ускоряет работу по сравнению со стандартной схемой хранения структуры на каждом агенте. БД позволяет увеличить отказоустойчивость системы за счет использования нескольких модулей хранения. Также за счет размещения экземпляров БД на разных хостах можно увеличить быстродействие для территориально удаленных объектов и повысить отказоустойчивость системы в целом. В качестве платформы используется MongoDB.

Пользователь функционирует с блоком хранения при первоначальном запросе и при необходимости – с агентом хранения. Взаимодействие между пользователем и блоками модуля осуществляется по REST-интерфейсу.

Порядок взаимодействия модулей друг с другом

1) Пользователь (клиент) посылает запрос модулю хранения при помощи сформированного сообщения в формате JSON.
2) При обработке запроса модулем хранения выполняется обращение к базе данных с целью получения информации о выполняемой в рамках запроса операции.
3) Пользователь получает результат выполнения действий на стороне модуля хранения в виде сформированного запроса к заданному агенту хранения.
4) Обработка запроса на работу с файлом агентом хранения и определение статуса выполнения операции.

3.3. Основные классы

Ниже приводятся сокращенные описания структуры и методов основных классов компонента хранения файлов.

3.3.1. Класс DSagent

Основной класс агента хранения. Реализует обработку запросов от клиентов на получение, доставку и удаление файлов. Также выполняет регистрацию агента в хранилище и предоставляет интерфейс для проверки работоспособности.

Свойства

- configProperty (тип: DSConfig) – конфигурационные параметры агента.

Внутренние классы

- Класс DSagentHttpHandler – обработчик запросов.
- Класс DSagentRequestListenerThread – диспетчер запросов. При поступлении запроса создает поток для его обработки.
- Класс DSagentWorkerThread – создание потока для обработки запроса

Открытые методы

- getDocRootSize – подсчет размера директории
    a) входной параметр fDirectory (тип: File) – директория для подсчета ее размера;
    b) возвращает значение размера директории в байтах (тип: long).
- registerAgent – регистрация агента в хранилище
    a) возвращаемое значение отсутствует (тип: void).
- returnAgentStatus – возвращает при запросе /agent_status информацию о конфигурации и текущий размер квоты в формате JSON
    a) входной параметр response (тип: HttpResponse) – идентификатор для ответа на http-запрос к агенту;
    b) возвращаемое значение отсутствует (тип: void).

3.3.2. Класс DSagentHttpHandler

Класс для обработки запросов по протоколу http к агенту. В соответствии с http-методом выполняет соответствующее действие.

Свойства

- docRoot (тип: String) – путь до директории для хранения файлов.

Открытые методы

- handle – обработка http-запросов PUT, GET и DELETE к агенту.

3.3.3. Класс DSagentRequestListenerThread

Класс для обработки сетевого подключения. Для обработки клиентского подключения создается DSagentWorkerThread

Свойства

- httpService (тип: HttpService) – обработчик http.
- params (тип: Httpparams) – параметры http-подключения.
- serversocket (тип: ServerSocket) – сетевой сокет.

3.3.4. Класс DSagentWorkerThread

Класс для обработки http запроса в отдельном потоке.

Свойства

- conn (тип: HttpServerConnection) – структура связанная с обработчиком http.
- httpService (тип: HttpService) – обработчик http.

Классы модуля хранения представлены ниже.

3.3.5. Класс DSCore

Основной класс модуля хранения. Реализует поддержку и управление соединениями с клиентами и агентами в многопоточном режиме, хранит пул соединений с базой данных.

Свойства

- Balancer (тип: Agent) – экземпляр класса DSBalancer.
- Cfg (тип: DSConfig) – экземпляр класса DSConfig.
- StoragePort (тип int) – порт для подключения.
- HeartBeatPeriod (тип: long) – временной период синхронизации с агентами.
- Agents (тип ArrayList) – список агентов на момент запуска ядра.

Внутренние классы

- Класс HttpHandler – управление http-подключениями и передачей данных. Является интерфейсом всего модуля.
- Класс EventLogger – обработчик ошибок для http-подключений.
- Класс Agent – хранение данных об агенте.

Закрытые методы

- putFile– сохранение файла и его метаданных в базе
    a) входной параметр FileName (тип: String) – имя файла;
    b) входной параметр FileSize (тип: long) – размер файла в байтах;
    c) входной параметр Version (тип: int) – номер новой версии файла;
    d) входной параметр Comment (тип: String) – комментарий к версии файла;
    e) входной параметр Agent (тип: String) – IP адрес агента;
    f) входной параметр Port (тип: int) – порт, на котором работает агент;
    g) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    h) возвращает результат операции в виде строки (тип: String).

- getFileVersion – получение номера следующей версии файла
    a) входной параметр FileName (тип: String) – имя файла;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает следующую версию файла (тип: int).
- getFile – получение адреса и номера порта агента, который хранит запрашиваемый файл по его идентификатору
    a) входной параметр FileID (тип: String) – идентификатор файла;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает результат операции в виде строки (тип: String).
- getFile – получение адреса и номера порта агента, который хранит запрашиваемый файл по его имени и номеру версии
    a) входной параметр FileName (тип: String) – имя файла;
    b) входной параметр Version (тип: int) – номер новой версии файла;
    c) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    d) возвращает результат операции в виде строки (тип: String).
- getMetaFileName – получение метаданных о файле по его имени
    a) входной параметр FileName (тип: String) – имя файла;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает результат операции в виде строки (тип: String).
- getMetaFileID – получение метаданных о файле по его идентификатору
    a) входной параметр FileID (тип: String) – идентификатор файла;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает результат операции в виде строки (тип: String).
- deleteFile – удаление файла по его идентификатору
    a) входной параметр FileID (тип: String) – идентификатор файла;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает результат операции в виде строки (тип: String).
- deleteFile – удаление файла по его имени и номеру версии
    a) входной параметр FileName (тип: String) – идентификатор файла;
    b) входной параметр Version (тип: int) – номер версии файла;
    c) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    d) возвращает результат операции в виде строки (тип: String).
- agentReg – регистрация агента
    a) входной параметр agent (тип: String) – IP-адрес агента;

b) входной параметр port (тип: int) – порт, на котором работает агент;

c) входной параметр quota (тип: long) – размер доступного мета в килобайтах;

d) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;

e) возвращает результат операции в виде строки (тип: String).

- agentUnreg – отмена регистрации агента.
    a) входной параметр agent (тип: int) – IP-адрес агента;
    b) входной параметр object (тип: String) – объект, для которого формируется сообщение;
    c) возвращает результат операции в виде строки (тип: String).
- returnMsg – возвращает текст ответа в зависимости от переданного кода.
    a) входной параметр code (тип: int) – код сообщения;
    b) входной параметр Db (тип: DB) – подключение к СУБД MongoDB;
    c) возвращает результат операции в виде строки (тип: String).
- makeJSON – формирует объект JSON.
    a) входной параметр msg (тип: String) – сообщение;
    b) входной параметр status (тип: String) – статус сообщения;
    c) возвращает результат операции в виде строки (тип: String).
- parseJSON – формирует объект JSON из строки
    a) входной параметр obj (тип: String) – строка для обработки;
    b) возвращает результат в виде объекта JSON.

3.3.6. Класс DSConfig

Читает конфигурационный файл.

Свойства

- configFile (тип: Properties) – экземпляр класса.

Открытые методы

- getProperty – чтение значения для параметра
    a) входной параметр key (тип: String) – параметр, для которого надо получить значение;
    b) возвращает значение параметра в виде строки (тип: String).

3.3.7. Класс DSBalancer

Балансировщик нагрузки для агентов. Следит за равномерностью распределения нагрузки при закачке файлов. Также хранит номер порта и текущее свободное дисковое пространство (квоту) агентов и не допускает превышения этой квоты. Поддерживает многопоточный доступ.

Свойства

- objectList (тип: ArrayList<Agent>) – список агентов.
- position (тип: AtomicInteger) – номер текущего агента, сохраняющего файл.
- isInit (тип: boolean) – для начальной инициализации списка при запуске.
- listSize (тип: int) – размер списка.

Открытые методы

- init – инициализация списка при запуске
    a) входной параметр objects (тип: List<Agent>) – список объектов типа Agent;
    b) возвращаемое значение отсутствует (void).
- getNext – получение следующего агента для загрузки файла
    a) входной параметр отсутствует;
    b) возвращает объект типа Agent из списка.
- getNextPosition – получение номера следующего агента для загрузки файла
    a) входной параметр отсутствует;
    b) возвращает номер агента из списка (тип: int).
- addAgent – добавление нового агента в список
    a) входной параметр newAgent (тип Agent) – агент, который необходимо добавить в список;
    b) возвращаемое значение отсутствует (void).
- removeAgent – удаление агента из списка
    a) входной параметр myAgent (тип Agent) – агент, который необходимо удалить из списка;
    b) возвращает статус операции – успех или провал (тип: boolean).
- removeAgent – удаление агента из списка по номеру
    a) входной параметр i (тип int) – номер агента в списке;
    b) возвращает статус операции, успех или провал (тип: boolean).
- findAgent – поиск агента в списке доступных агентов

a) входной параметр myAgent (тип Agent) – агент, который необходимо найти в списке;
b) возвращает агента из списка (тип: Agent).

- getAgent – получение объекта типа Agent по его индексу в списке
  a) входной параметр i (тип int) – номер агента в списке;
  b) возвращает агента из списка (тип: Agent).
- numAgent – количество агентов в списке
  a) входной параметр отсутствует;
  b) возвращает количество агентов в списке (тип: int).

3.3.8. Класс DSDBConnect

Устанавливает соединение с базой данных и поддерживает подключение.

Свойства

- mongo (тип: Mongo) – объект, который хранит подключение.

Методы

- getConnection – создает подключение и поддерживает соединение
  a) входной параметр отсутствует;
  b) возвращает подключение к СУБД (тип: DB).

3.3.9. Класс DSLiveChecker

Проверяет доступность зарегистрированных агентов. В случае обнаружения недоступного агента удаляет его из списка балансировщика и базы данных.

Свойства

- mongo (тип: Mongo) – объект который хранит подключение.
- AgentAddress (тип: String) – IP-адрес агента.
- HeartBeatPeriod (тип: long) – период проверки доступности агентов, мс.
- balancer (тип: DSBalancer<Agent>) – список агентов балансировщика.
- db (тип: DB) – подключение к СУБД MongoDB.

Методы

- removeAgent – удаление агента
  a) входной параметр AgentAddress (тип: String) – IP-адрес агента;
  b) входной параметр reason (тип: ReasonEnum) – причина отмены;
  c) возвращает курсор на объект в базе данных (тип: BasicDBObject).

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Программный компонент хранения данных предназначен для функционирования на аппаратных системах с характеристиками:

- архитектура процессора – все поддерживаемые JRE 1.6, список доступен по ссылке <http://www.oracle.com/technetwork/java/javase/system-configurations-135212.html> ;
- минимальный объем оперативной памяти – 1 ГБ;
- минимальный объем свободного пространства на жестком диске – 1 ГБ;
- минимальная тактовая частота процессора – 1 ГГц.

Агент хранения требует наличия JRE не ниже 1.6, модуль хранения – дополнительно требует установленного MongoDB не ниже 2.0.0.

5. ВЫЗОВ И ЗАГРУЗКА

На первом этапе необходимо запустить базу данных MongoDB.

На втором этапе необходимо запустить модуль хранения. Запуск производится из командной строки приложения java.exe с параметром в виде предварительно скомпилированного jar файла dStorage.jar с указанием конфигурационного файла. Для ОС семейства Windows пример запуска: **java -jar dStorage.jar c:\dist\core.cfg.**

На третьем этапе необходимо создать конфигурационный файл для запуска агента. В этом файле необходимо указать сетевой адрес и порт хранилища, а также сетевой адрес и порт агента, путь к каталогу для хранения файлов, после этого запустить в командной строке приложение java.exe с параметром jar файла DSagent.jar с указанием конфигурационного файла. Для ОС семейства Windows пример запуска: **java -jar DSagent.jar c:\dist\agent.cfg.**

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для компонента являются конфигурационные файлы, файлы, предназначенные для хранения в нем, а также объекты JSON, получаемые от клиентов и агентов хранения компонента в процессе работы.

```
AgentPort = 8888
AgentAddress = 192.168.4.102
AgentdocRoot = c:\\dist\\
AgentQuota = 500000000
StorageCorePort=8080
StorageCoreAddress=192.168.4.214
```

Рисунок 6.1 – Пример конфигурационного файла агента хранения

```
StoragePort = 8080
DBAddress = 127.0.0.1
DBPort = 27017
HeartBeatPeriod = 120000
```

Рисунок 6.2 – Пример конфигурационного файла модуля хранения

Сообщения JSON от клиентов различаются в соответствии с типом операции над файлом.

Примеры для различных типов операции сообщений JSON приведены в табл. 6.1

Таблица 6.1

Примеры сообщений JSON в соответствии с операцией

Операция на файлом	Описание операции	Пример сообщения JSON
PUT	Загрузка файла в хранилище	{'action': 'post', 'file_size': 3462785445, 'file_name': 'data.bin'}
GET	Получение файла из хранилища (последней версии)	{'action': 'get', 'file_id': 'A4R5OPE38123'}
DELETE	Удаление файла из хранилище	{"action":"DELETE","file_id":"4ef6ac0c9c0405901e5a2c25"}
GET_METADATA	Получение метаданных о файле	{'action': 'get_metadata', 'file_name': '18.bin'}

7. ВЫХОДНЫЕ ДАННЫЕ

Для модуля хранения выходными данными являются ответы в виде JSON сообщений. Сформированный ответ зависит от статуса операции (успех/неудача) и типом входного запроса. Примеры ответов указаны в табл. 7.1.

Таблица 7.1

Примеры ответов в виде JSON сообщений от модуля хранения

Операция на файлом	Статус операции	Пример сообщения JSON
PUT	Успех	{'file_id': '4f1408f49c043f6136ff61f1', 'agent_uri': 'http://10.0.0.10:8989'}
GET	Неудача	{'Status': 'Error', 'MSG': 'File 4f0ac6ce9c04c68a8ef63af8 not found'}
DELETE	Успех	{'agent_uri': 'http://10.0.0.10:8989'}}
GET_METADATA	Успех	{'id': '4f104e559c04b2fe9163a0c9', 'FileSize':128, 'Status': 'OK', 'Time': '19:31:33', 'Date': '13.01.2012', 'Comment': '', 'AgentAddress': '10.0.0.10', 'Version':1}

Для агентов основными выходными данными являются файлы, а также статус в виде сообщения в формате JSON, в котором указаны сетевой адрес, порт и текущее доступное место.

Пример: {'quota': '10000', 'agent_port': '8888', 'action': 'status', 'agent_ipaddress': '10.67.1.15'}.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

API	Application programming interface (интерфейс программирования приложений)
DFT	Метод функционала плотности
WCF	Windows Communication Foundation (программный фреймворк)
WF	Workflow (поток заданий)
XML	eXtensible Markup Language (расширяемый язык разметки)
БД	База данных
БЗ	База знаний
ИС	Информационная система
МИТП	Многопрофильная инструментально-технологическая платформа
ОС	Операционная система
ПО	Программное обеспечение
СУБД	Система управления базами данных
ЭВМ	Электронно-вычислительная машина

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	**Номера листов (страниц)**				**Всего листов (страниц) в докум.**	**№ документа**	**Входящий № сопроводительного докум. и дата**	**Подп.**	**Дата**
	измененных	**замененных**	**новых**	**аннулированных**					