МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**“Санкт-Петербургский государственный электротехнический университет
“ЛЭТИ” им. В.И.Ульянова (Ленина)”
(СПбГЭТУ)**

Факультет КТИ

Специальность АСОИУ

Кафедра АСОИУ

К защите допустить:
Заведующий кафедрой **Советов Б. Я.**

Пояснительная записка

К ДИПЛОМНОМУ ПРОЕКТУ

Тема: «Обеспечение тестирования взаимодействия Flex приложений с сервером через AMF»

Студент	_____	*/ Тедикова О.В. /*
Руководитель	_____	*/ Выговский Л. С. /*
Консультант по экономическому обоснованию	_____	*/ Швецова О.А. /*
Консультанты	_____	*/ Берковская К.И. /*

Санкт-Петербург
2012 г.

Реферат

В этой работе ...

Содержание

Глоссарий

Распределённый — Слово, которое нельзя употреблять. Но надо протестировать длинные строки в глоссарии.

Обозначения и сокращения

АИС — Автоматизированная информационная система.

Введение

Данный дипломный проект ориентирован на практическое применение навыков автоматизации тестирования: обзор и анализ существующих методов и средств нагрузочного тестирования, разработку программного обеспечения, в значительной степени снижающего сложность осуществления функционального и нагрузочного тестирования взаимодейтсвия клиента и сервера по AMF-протоколу, разработку методики функционального и нагрузочного тестирования с использованием предложенного ПО.

В настоящее время активно развивается рынок Web-приложений. Все больше различных услуг предоставляется через Интернет, классические информационные системы в различных компаниях также становятся Web-приложениями, предоставляя для своих сотрудников и партнеров доступ к ресурсам компании не только из локальной, но и из глобальной сети. Концепции и технологии, используемые при разработке web-приложений, постоянно развиваются и совершенствуются. Оптимизируется использование ресурсов и времени, улучшаются возможности по отображению предоставляемой информации, динамичность и интерактивность web-приложений.

На сегодняшний день одним из наиболее перспективных подходов к обеспечению всего вышеперечисленного является концепция Rich Internet Application (в дальнейшем RIA) - это приложения, доступные через сеть Интернет, обладающие особенностями и функциональностью традиционных настольных приложений.

Приложения RIA, как правило:

а) передают веб-клиенту необходимую часть пользовательского интерфейса, оставляя большую часть данных (ресурсы программы, данные и пр.) на сервере;

б) запускаются в браузере и не требуют дополнительной установки программного обеспечения;

в) запускаются локально в среде безопасности, называемой «песочница» (sandbox).

На Рис. 0.1 демонстрируется место Rich Internet Applications среди других технологий программных систем

Рисунок 0.1 — RIA среди других технологий

Термин «RIA» впервые был упомянут компанией Macromedia в официальном сообщении в марте 2002 года. Подобные концепции существовали и несколькими годами ранее, например:

а) DHTML — HTML страницы с динамическим содержимым;

б) Java-applet — это программы написанные на языке Java, как правило, предназначенные для загрузки посредством браузера;

Для начала рассмотрим архитектурные особенности построения web-приложений базирующихся на RIA концепции. Для этого сравним принцип работы традиционного web-приложения и RIA-приложения (Рис. 0.2).

Работа традиционных web-приложений сконцентрирована вокруг клиент серверной архитектуры с тонким клиентом. Такой клиент переносит все задачи по обработке информации на сервер, а сам используется в основном для отображения статического контента. Основной недостаток этого подхода в том, что все взаимодействие с приложением должно обрабатываться сервером, что требует постоянной отправки данных на сервер, ожидания ответа сервера, и загрузки страницы обратно в браузер. В отличие от традиционного web-приложения в RIA значительная часть функционала исполняется на стороне клиента, поэтому появляется возможность отправлять и получать данные с сервера только по мере

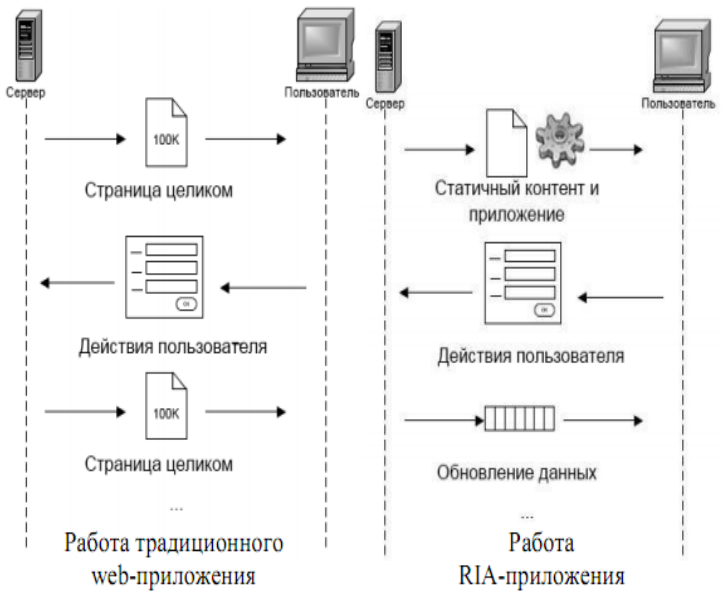

Рисунок 0.2 — Принципы работы традиционного web-приложения и RIA приложения

необходимости. Если подключение нестабильно, клиентская часть RIA-приложения обладает возможностями кэширования данных и работы без подключения к сети в режиме offline.

На основе вышесказанного, можно выделить ряд преимуществ RIA как перед настольными приложения, так и перед стандартными веб-приложениями:

а) для работы с RIA не требуется установка приложения, как правило, все что необходимо, это установка плагина для браузера;

б) пользователи могут использовать приложение на любом компьютере, имеющем соединение с Интернет, причем обычно не важно, какая операционная система на нем установлена;

в) интерфейс RIA обладает большей интерактивностью, он не ограничен лишь использованием языка HTML, применяемого в стандартных веб-приложениях. Наиболее сложные приложения RIA предлагают внешний вид и функциональность, близкие к настольным приложениям;

г) при работе веб-приложения компьютер пользователя гораздо меньше подвержен риску вирусному заражению, чем при запуске исполняемых бинарных файлов;

д) использование вычислительных ресурсов клиента и сервера лучше сбалансировано;

е) взаимодействие клиента с сервером осуществляется без ожидания каких-либо действий пользователя.

Для реализации RIA многими IT-компаниями предлагаются различные технологии. Наиболее известными из них являются AJAX, Flash, Flex (в дальнейшем Flex) и AIR фирмы Adobe; ActiveX, WPF и Silverlight корпорации Microsoft; Java FX и Java Applets компании Sun Microsystems.

В настоящее время одной из наиболее распространенных технологий разработки RIA является Adobe Flex, которая представляет собой среду с открытым кодом для создания и обслуживания web-приложений, совместимую со всеми распространенными браузерами, платформами персональных компьютеров и версиями операционных систем и имеющую максимальную поддержку со стороны разработчиков.

Flex представляет собой набор классов, расширяющих возможности Flash. Flex позволяет описывать интерфейс web-приложения на MXML – декларативном языке описания и настройки свойств визуальных элементов интерфейса, основанном на XML. Логика web-приложения пишется на ActionScript – полноценном объектно-ориентированном языке программирования, который является одним из диалектов ECMAScript. Результатом компиляции является файл SWF, предназначенный для выполнения в браузере (на платформе Flash Player, которая существует в виде плагина к браузеру) или как самостоятельное приложение (на платформе AIR).

Flex-framework включает возможности локализации, стилизации приложения, разработки модульного приложения, встроенные валидаторы и форматоры текстовых полей — все те инструменты, которые нужны разработчикам приложений, работающих online. Также Flex предоставляет полные мультимедийные возможности Flash Platform, такие как потоковое мультимедиа, возможность получить доступ к веб-камере и микрофону пользователя, бинарные сокеты, обширные возможности сетевых коммуникаций (HTTP-запросы, веб-сервисы, встроенный формат

сериализации AMF), оперирование координатами трехмерного пространства).

Вся разработка во Flex ориентирована на применение готового набора расширяемых компонентов, внешний вид которых позволяет гибко настраивать CSS, что облегчает задачу разработчика.

Flex SDK является бесплатным инструментарием с июня 2007 года с открытым исходным кодом, распространяемым на условиях Mozilla Public License. Для работы с процедурами и классами этого фреймворка можно использовать как бесплатные (Eclipse WTP IDE, FlashDevelop IDE) так и платные (Flex Builder IDE, IntelliJ IDEA IDE, Aptana Studio IDE, PowerFlasher FDT IDE) среды разработки.

Flex приложения предоставляют возможность реализации клиент-серверного взаимодействия на основе бинарного формата обмена данными — AMF(Action Message Format). AMF используется для сериализации структурированных данных, таких как объекты Action Script или XML, и обмена сообщениями между Adobe Flash клиентом и удалённым сервисом. AMF более экономичен по трафику по сравнению с XML и позволяет передавать типизированные объекты.

Как известно огромную роль в жизненном цикле программного обеспечения играет фаза тестирования. Тестирование — один из важнейших этапов контроля качества в процессе разработки ПО. Автоматизированное тестирование является его составной частью. Оно использует программные средства для выполнения тестов и проверки их результатов, что помогает сократить время тестирования и упростить его процесс, а также может дать возможность выполнять определенные тестовые задачи намного быстрее и эффективнее чем это может быть сделано вручную. Автоматизация тестирования даёт следующие преимущества:

а) Автоматизация — выполнение сложно воспроизводимых вручную тестов, таких как нагрузочное и стресс-тестирование;

б) Меньшие затраты на поддержку — когда автоматические скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время чем на проведение того же объема тестирования вручную;

в) Повторяемость — все написанные тесты всегда будут выполняться однообразно, то есть исключен «человеческий фактор». Тестировщик не пропустит тест по неосторожности и ничего не напутает в результатах;

г) Быстрое выполнение — автоматизированному скрипту не нужно сверяться с инструкциями и документациями, это сильно экономит время выполнения;

д) Отчеты — автоматически рассылаемые и сохраняемые отчеты о результатах тестирования.

Если сложить все вышесказанное, то больший объем тестирования, может быть достигнут с меньшими усилиями, давая при этом возможность улучшать как качество, так и продуктивность.

Однако использование AMF вызывает ряд трудностей для реализации автоматизации функционального и нагрузочного тестирования взаимодействия сервера и Flex клиента, связанных с бинарной природой протокола. Так как amf сообщение представляет собой совокупность байтов, разработчикам и тестировщикам трудно считывать и изменять содержащуюся в нём информацию.

Отдельной проблемой является нагрузочное тестирование таких приложений — имитация работы с приложением большого количества пользователей за счёт запуска набора тестов в несколько потоков. Сложность состоит в поддержке уникальности идентификатора сессии каждого клиента. Идентификатор клиента выделяется строго одному подключенному SWF-файлу и прошит в передаваемых данных. Стандартный порядок тестирования — запись запросов пользователя с использованием прокси-сервера и запуск сохранённого сценария в несколько потоков — в данном случае будет неэффективным, так как все перехваченные запросы будут иметь одинаковый идентификатор клиента и тесты пройдут только в 1 потоке, а остальные будут возвращены с ошибкой о неправильной сессии.

Таким образом, можно сформулировать наличие противоречия между необходимостью проведения функционального и нагрузочного тестирования и сложностью его реализации на технлогии Flex и поставить следующие задачи на дипломную работу:

а) разработка программного обеспечения, в значительной степени снижающего сложность осуществления функционального и нагрузочного тестирования взаимодейтсвия клиента и сервера по AMF-протоколу;

б) разработка методики функционального и нагрузочного тестирования с использованием предложенного ПО.

1 Анализ существующих решений

На сегодняшний день существует ряд различных программных комплексов, предоставляющих возможность для проведения функционального, нагрузочного, регрессионного тестирования различного рода приложений. Проведём обзор наиболее распространённых и доступных утилит и укажем их функционал по обеспечению тестирования в области Flex технологий.

1.1 Обзор утилит для тестирования Flex приложений

1.1.1 HP QuickTest Professional

HP QuickTest Professional (QTP) — один из инструментов автоматизации функционального тестирования, является флагманским продуктом компании HP в своей линейке. Для разработки автоматизированных тестов QTP использует язык VBScript(Visual Basic Scripting Edition) — скриптовый язык программирования, интерпретируемый компонентом Windows Script Host. Он широко используется при создании скриптов в операционных системах семейства Microsoft Windows. QTP поддерживает ряд технологий, среди которых есть и Macromedia Flex. Поддержка Flex осуществляется засчёт установки плагина, предоставляемого компанией Adobe (Flex QTP add-in).

Чтобы приступить к тестированию приложение необходимо скомпилировать, создав для него HTML-оболочку, и развернуть его либо локально, либо на web-сервере. Создание тестов в QTP осуществляется следующим образом: приложение открывается в браузере и все действия, совершаемые пользователем с пользовательским интерфейсом приложения записываются в виде строчек Visual Basic скрипта. QTP поддерживает запись большинства наиболее часто используемых во Flex приложениях событий (операций), связанных с пользовательским интерфейсом. Однако часть из них, например некоторые атомарные операции, игнорируются во время записи тестов. Пользователь имеет возможность добавить их в текст скрипта вручную. Для проверки правильности выполнения теста пользователь задаёт ожидаемые значения для выполняемых операций —

checkpoints. Во время автоматического прогона тестов запускать браузер уже не нужно, достаточно лишь указать HTML страницу, используемую для тестов. Для выявления причины возникновения ошибок в тестах или каких-либо других неполадок можно обратиться к логам Flash Player или настроить и включить логирование в QTP. Возможность прогона тестов в несколько потоков в QTP отсутствует

Для проведения тестов с HP QuickTest Professional необходимо использовать браузер Internet Explorer версии 6 или выше. HP QuickTest Professional работает с операционными системами семейства Windows и является платным программным обеспечением.

1.1.2 IBM Rational Functional Tester

IBM Rational Functional Tester является средством автоматизированного регрессивного тестирования, предназначенным для тестирования Java, .NET, Web-приложений, включая Flex-приложения, и терминальных приложений на платформах Windows и Linux. Functional Tester поддерживает два языка сценариев: Java и Visual Basic.NET. Для тестирования Java-приложений в программу Functional Tester включена открытая среда разработки Eclipse. Установка дополнительных компонентов не требуется. Если требуется использовать язык сценариев Visual Basic.NET, перед установкой IBM Rational Functional Tester необходимо установить Visual Studio.NET.

Перед началом тестирования приложение загружается в браузере, затем все действия,совершаемые пользователем записываются в виде Java или Visual Basic.NET скрипта (в зависимости от настроек). Пользователь может редактировать скрипты, а также задавать ожидаемые результаты выполнения той или иной операции для проверки работы приложения. Как и QTP, IBM Rational Functional Tester во время записи тестов добавляет компоненты приложения в свой набор объектов. Далее тесты можно запускать в автоматическом режиме. Также в IBM Rational Functional Tester имеется возможность многократного запуска тестов с различным набором данных. Если стандартного набора объектов(таких как кнопки, текстовые поля и т.д.) недо-

статочно, существует возможность самостоятельно добавить во фреймворк необходимые объекты. Это возможно благодаря Rational Functional Tester proxy software development kit (SDK), который имеет удобное API и довольно подробную сопроводительную документацию. Также IBM Rational Functional Tester упрощает механизм регрессионного тестирования. Functional Tester использует усовершенствованную технологию ScriptAssure для того, чтобы «изучить» контрольные характеристики пользовательского интерфейса, что позволяет идентифицировать те же самые средства управления в новой версии, несмотря на внесенные изменения. Эти характеристики сохраняются в объектной карте, совместный доступ к которой могут получить различные скрипты и участники проекта. Благодаря этой карте изменения, внесенные в характеристики распознавания объекта, будут отражены во всех скриптах тестирования, что существенно упрощает обслуживание.

Rational Functional Tester поддерживает операционные системы Windows 2000, XP, Vista, 7, Linux. Является платным программным обеспечением.

1.1.3 NeoLoad

NeoLoad — профессиональный инструмент нагрузочного тестирования веб-приложений, в том числе и Flex, работающий на многих платформах, в том числе Windows, Solaris, Linux. Осуществляя моделирование большого числа пользователей, которые обращаются к приложению, NeoLoad делает проверку надежности и производительности приложения при разных нагрузках.

Тестирование Flex приложений осуществляется засчёт записи AMF трафика во время взаимодействия пользователя с приложением. Все перехваченные запросы отображаются в xml формате, что позволяет редактировать содержащиеся в них данные. Далее записанные тесты могут быть запущены в несколько потоков.

NeoLoad является платным программным обеспечением с закрытым исходным кодом.

1.1.4 Apache JMeter

JMeter — инструмент для проведения нагрузочного тестирования, изначально разрабатывался как средство тестирования web-приложений, в настоящее время он способен проводить нагрузочные тесты для JDBC-соединений, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP и TCP. В JMeter реализованы возможность создания большого количества запросов с помощью нескольких компьютеров при управлении этим процессом с одного из них, логирование результатов тестов и разнообразная визуализация результатов в виде диаграмм, таблиц и т. п.

Ключевое понятие в JMeter —- план тестирования. План тестирования приложения представляет собой описание последовательности шагов, которые будет исполнять JMeter. План тестирования может содержать:

а) Группы потоков (Thread Groups) — элемент, позволяющий конфигурировать многопоточный запуск тестов

б) Контроллеры — позволяют создаватьтесты со сложной логической структурой.

в) Слушатели — отображают результаты выполнения тестов

г) Соответствия — позволяют сравнивать полученные результаты с ожиданиями пользователя

д) Сэмплеры — основные элемнты тест-плана, в которых формируется тело запроса, тестовый шаг.

На данный момент в JMeter нет отдельных средств для тестирования Flex приложений, однако есть возможность записи http запросов, содержащих в себе тело AMF сообщения, с помощью прокси-сервера. Далее перехваченные запросы могут быть перенесены в план тестирования и запущены в режиме нагрузочного тестирования. Созданные в JMeter тесты могут быть запущены с помощью систем сборок проектов Maven и Ant. Также JMeter свободно интегрируется со многими серверами сборок, такими как Jenkins и Hudson.

JMeter является бесплатным кросс-платформенным Java-приложением с открытым исходным кодом и удобным API. Существует боль-

шое количество плагинов к JMeter, значительно расщиряющих его базовую функциональность.

2 Проектирование модуля

2.1 Требования к разрабатываемому модулю

2.1.1 Формулировка требований

Сформулируем ряд функциональных требований к разрабатываемому модулю. Требования должны быть составлены таким образом, чтобы результат их реализации удовлетворял одной из поставленных на дипломный проект задаче — разработанное ПО должно в значительной степени снижать сложность осуществления функционального и нагрузочного тестирования взаимодейтсвия клиента и сервера по AMF-протоколу.

а) Разрабатывамый модуль должен предоставлять возможность записи действий пользователя, производимых с тестируемым приложением(проксирование запросов). Проксирование является важным функциональным требованием, так как на этапе подготовки тестирования позволит создавать тестовый сценарий, на основе действий пользователя с приложением.

б) Программный модуль должен предоставлять возможность отправки AMF сообщений через http соединение по указанному url

в) Программный модуль должен полностью поддерживать протокол AMF с целью кодирования, декодирования и замены сообщений. Все полученные или отправленные сообщения должны предоставляться пользователю в человекочитаемом формате.

г) Реализация программного модуля должна полностью поддерживать основные прнципы нагрузочного тестирования имитирующего работу большого количества пользователей одновременно.

д) Разрабатываемое приложение должно интегрироваться с системами автоматической сборки проектов, такими как maven и ant

е) Возможность запуска тестов из командной строки. Данная возможность позволяет выполнять тесты в автоматическом режиме в системах непрерывной интеграции, что является необходимостью, особенно в тех случаях, когда над разными частями тестируемой системы разработчики трудятся независимо и необходимо выполнении частых авто-

матизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. Часто производители тестовых фреймворков предоставляют свои продукты для проведения интеграционного тестирования, однако зачастую они узко заточены под конкретный круг задач, поэтому нас будет интересовать возможность запуска тестов на таких инструментах интеграции, как Jenkins и Hudson, которые на данный момент широко распротранены и имеют множество плагинов, позволяющих значительно расширить их существующую функциональность.

ж) Кроссплатформенность. Созданная нами тестовая утилита должна работать под управлением различных операционных систем.

з) Условия распростронения продукта. В идеале программное обеспечение должно быть бесплатным и иметь открытый исходный код с возможность создания собственных расширений, это является одним из основных критериев отбора, так как тестировщики и разработчики всегда должны иметь возможность доработки и усовершенствования существующего функционала тестового фреймворка, чтобы адаптировать его под специфику работы тестируемого ими приложения.

2.1.2 Анализ требований

Проанализируем сформулированные требования к разрабатываемому модулю. Для наглядности и удобства сведём их в таблицу. Для классификации требований были выбраны следующие критерии:

а) Приоритет — критерий оценки полезности реализации требования для конечного пользователя, важности для достижения поставленных перед проектом целей.

б) Трудоёмкость — критерий отображает предварительную оценку сложности реализации требования, количество привлекаемых для этого ресурсов.

в) Риск — интегральный критерий, введением которого предпринимается попытка оценить во-первых возможность невыполнения требования, во-вторых возможную ошибку в оценке по двум предыдущим критериям (в первую очередь трудоёмкости).

Для каждого из критериев введена шкала из трёх уровней: низкий, средний, высокий. Размерность шкалы выбрана минимальной исходя из соображений простоты и наглядности. Так как размер проекта и количество предъявляемых к нему требований незначительны, то таких шкал вполне достаточно для исчерпывающей классификации требований а также принятия проектных и организационных решений.

2.2 Архитектура модуля

3 Программная реализация модуля

3.1 Выбор технических средств решения задачи

Как показали результаты обзора существующих тестовых фреймворков, на данный момент нет решения польностью удовлетворяющего поставленным требованиям. Однако многие рассмотренные программные комплексы уже предоставляют часть необходимого нам функционала, поэтому для решения поставленной на дипломный проект задачи нет необходимсти создавать тестовую утилиту с нуля, разумнее будет выбрать одно из существующих решений, и доработать его. Чтобы выбрать из предложенного разнообразия подходящий продукт, составим таблицу, где укажем, в какой степени каждый тестовый фреймворк удовлетворяет поставленным требованиям.

ТУТ ТАБЛИЦА

Анализ показал, что оптимальным выбором, в наибольшей степени удовлетворяющим поставленным требованиям, является Apache JMeter. Именно на базе его функционала будет реализовано решение задачи дипломногот проекта.

Разработка модуля будет осуществляться с использованием языка Java, что обусловлено одним из требований технического задания - кросплатформенность приложения. Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор, что обеспечивает полную независимости байт-кода от операционной системы и оборудования и позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Также выбранный нами тестовый фреймворк Apache JMeter является стопроцентным Java приложением, поэтому для интеграции с ним целесообразно использовать именно этот язык.

На текущий момент основным средством обеспечения взаимодействия Flex клиентов с Java приложениями является технология BlazeDS — серверная Java-технология для передачи данных, поддерживающая

AMF протокол. BlazeDS является бесплатным приложением с открытым исходным кодом,разработано компанией Adobe. В силу распространённости BlazeDS, одно из основных функциональных требований к реализации модуля—обработка AMF сообщений— будет решаться с помощью средств именно этой техноогии.

В качестве инструмента автоматизации сборки проектов был выбран Apache Maven фреймворк для автоматизации сборки проектов, специфицированных на XML-языке POM (Project Object Model).Основныvb преимуществами Maven являются:

а) Независимость от OS. Сборка проекта происходит в любой операционной системе. Файл проекта один и тот же.

б) Управление зависимостями. Редко какие проекты пишутся без использования сторонних библиотек(зависимостей), которые зачастую тоже в свою очередь используют библиотеки разных версий. Мавен позволяет управлять такими сложными зависимостями, что позволяет разрешать конфликты версий и в случае необходимости легко переходить на новые версии библиотек.

в) Возможна сборка из командной строки. Такое часто необходимо для автоматической сборки проекта на сервере (Continuous Integration).

г) Хорошая интеграция с средами разработки. Основные среды разработки на java легко открывают проекты которые собираются с помощью maven. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке. Как следствие - если с проектом работают в разных средах разработки, то maven удобный способ хранения настроек. Настроечный файл среды разработки и для сборки один и тот же - меньше дублирования данных и соответственно ошибок.

д) Декларативное описание проекта. В файлах проекта pom.xml содержится его декларативное описание, а не отдельные команды.

Эффективность разработки программного обеспечения в любом современном проекте подразумевает возможность вести разработку параллельно с другими участниками проекта.Для оптимизации совместной работы над дипломным проектом было принято решение о размеще-

нии всех файлов проекта в репозитории системы контроля версий Git. Git—это быстрая, масштабируемая, распределенная система управления версиями с большим набором команд, которые обеспечивают как операции верхнего уровня, так и полный доступ к внутренним механизмам.

В качестве среды разработки бал выбрана IntellijIdea. http://www.ozon.ru/context/detail/id/2331312/

3.2 Реализация поддержки AMF протокола

3.2.1 Общие сведения о BlazeDS

BlazeDS -серверная Java-технология для передачи данных. Предоставляет ряд сервисов, которые позволяют приложениям клиента взаимодействовать с сервером, а также осуществяет передачу данных между несколькими клиентами, подключенными к серверу BlazeDS, в режиме реального времени.BlazeDS приложение состоит из двух частей: клиентского приложения и серверного J2EE web-приложения. Данная архитектура представлена на следующем рисунке

ТУТ РИСУНОК

Клиентское приложение BlazeDS обычно представляет собой Adobe Flex или AIR приложение. В его состав входят:

а) Пользовательский интерфейс приложения. Создаётся с помощью Flex SDK;

б) Один или несколько компонентов BlazeDS:

1) RemoteObject — компонент, предоставляющий клиентскому приложению доступ к методам Java-объектов на стороне сервера;
2) HTTPService — компонент, позволяющий клиентскому приложению с помощью http запросов взаимодействовать с JSP, сервлетами, ASP страницами через сервер BlazeDS;
3) WebService — компонент, предназанченный для взаимодействия с веб-сервисами;
4) Producer — компонент-отправитель сообщений, предназанчен для взаимодействия с сервером сообщений;

5) Consumer — компонент-получатель сообщений, предназанчен для взаимодействия с сервером сообщений.

в) Набор каналов. На стороне клиента определяются каналы,которые инкапсулируют соединение между Flex клиентом и сервером BlazeDS. Для клиентского приложения задаётся набор каналов, упорядоченных по предпочтению. Flex компонент пытается подключиться по первому каналу, указанному в списке, и в случае неудачи выбирает следующий канал и т.д, до тех пор пока соединение не будет установлено, либо список каналов не кончится.Flex клиенты могут использовать различные типы каналов, такие как AMFChannel и HTTPChannel. AMFChannel использует бинарный AMF протокол, а HTTPChannel - небинарный формат AMFX (AMF, преобразованный в XML). Выбор канала зависит от ряда факторов, например от типа создаваемого приложения, формата передачи данных, требуемого размера сообщений.

Архитектура сервера BlazeDS представлена на ...

Тут РИСУНОК

BlazeDS сервер базируется на технологии J2EE. Взаимодействие клиента и сервера BlazeDS происхоит следующим образом : Flex клиент посылает запрос по определённому каналу, далее запрос направляется в соответствующий каналу компонент endpoint, который является точкой обработки получаемых сервеом сообщений различного типа. Затем сообщение декодируется и проходит через ряд Java-объектов - MessageBroker , Service object, Destination object и Adapter object. Adapter object либо обрабатывает запрос локально, либо связывается с какой-либо backend системой или удалённым сервером. После запроса происходит обратный процесс.

Таким образом для обеспечения поддержки AMF протокола в разрабатываемом программном обеспечении, необходимо будет реализовать модуль, осуществляющий сериализацию сообщений перед их отправкой от клиента к серверу и наоборот, десериализацию полученных клиентом сообщений сервера.

3.2.2 Сериализация и десериализация сообщений

Рассмотрим подробнее спецификацию AMF.

AMF - бинарный формат, используемый для сериализации структурированных данных, таких как объекты Action Script или XML, и обмена сообщениями между Adobe Flash клиентом и удалённым сервисом. Action Message Format более экономичен по трафику по сравнению с XML и позволяет передавать типизированные объекты. Первая версия протокола, AMF0, была применена в Flash Player 6 в 2001 и оставалась неизменной в реализациях ActionScript 2.0 в Flash Player 7 и Flash Player 8. В Flash Player 9 был внедрён Action Script 3.0 с усовершенствованной виртуальной машиной ActionScript Virtual Machine (AVM+). В новой версии протокола, AMF3, оптимизирован формат сообщений и введена поддержка новых типов. Далее будет рассматриваться именно обновлённая версия протокола.

AMF протокол поддерживает следующие типы данных:

а) undefined - любой неспецифицированный протоколом формат данных;

б) null;

в) boolean;

г) integer -в ActionScript 3.0 integer представляет собой 29х битное число переменной длины без знака или 28битное со знаком. Если выделенных битов не достаточно для представления числа, то AVM+ сериализует его как тип double;

д) double - 8ми байтовое число с плавающей точкой в формате IEEE 754 с порядком байтов от старшегок младшему (network byte order)

е) string - сторка в кодировке UTF-8, литерал или ссылка;

ж) xml-doc - данные в xml формате в кодировке UTF-8. При сериализации преобразуются в строку;

з) date - сериализуется как число миллисекунд, прошедших с 00:00:00 первого января 1970 в часовом поясе UTC, данные о локальном часовом поясе не посылаются;

и) array - массивы данных;

к) object - amf3 обрабатывает Action Script объекты, а также разработанные пользователем классы, которые можно разделить на следующие группы:

1) анонимные - Action Script объекты или экземпляры пользовательских классов, не имеющие зарегистрированных в протоколе псевдонимов. В процессе десериализации анонимные объекты будут интерпретироваться как Object;
2) типизированные - экземпляры классов с зарегистрированными в протоколе псевдонимами;
3) динамические - экземпляры классов, общедоступные переменные которых могут динамически изменяться во время выполнения кода;
4) сериализуемые - экземпяры классов, реализующих интерфейс flash.utils.IExternalizable, которые полностью контролируют сериализацию своих членов.

л) xml - в Action Script 3.0 введена поддержка XML синтаксиса E4X;

м) byte-array - в Action Script 3.0 введён новый формат хранения массива байтов, ByteArray. AMF3 сериализует данные этого типа с использованием 29ти битного значения, передаваемого в качестве длины массива байтов.

AMF сообщение содержит информацию об отдельной транзакции. Оно определяют вызываемую удалённую операцию, операцию, выполняемую клиентом в случае успеха или ошибки запроса, и используемые при этом данные. Структуры сообщений, представляющих запросы клиента и ответы сервера, одинаковы.

Первым полем AMF сообщения является target URI, которое описывает, какая операция, функция или метод должны быть вызваны. Спецификация AMF не определяет точного формата этого поля — его формат зависит от конкретной реализации используемого сервера.

Вторым полем AMF сообщения является response URI, определяющее имя операции, которая будет вызвана в зависимости от ответа на запрос клиента.

Третье поле AMF сообщения — длина тела сообщения в байтах.

Четвртым заключительным полем AMF сообщения является тело сообщения. Оно содержит фактические данные, связанные с выполняемой операцией. Если сообщение является запросом клиента, то оно должно содержать параметры, передаваемые удалённому методу. Если сообщение является ответом сервера, то оно должно содержать либо ответ на запрос, либо сообщение об ошибке.

Отправка AMF сообщений осуществляется пакетами, какждый из которых имеет следующую структуру:

а) версия AMF протокола;
б) количество заголовков пакета;
в) заголовки пакета;
г) число сообщений пакета;
д) сообщения пакета;

В библиотеке BlazeDS существует класс, соответсвующий структуре AMF пакетов — ActionMessage. Чтобы получить экземляр ActionMessage из входного потока данных, используется класс AmfMessageDeserializer. Следующий код демонстрирует этот процесс:

ТУТ КОД

Метод initialize(SerializationContext context, InputStream in, AmfTrace trace) устанавливает контекст для чтения данных из указанного входного потока. Метод readMessage(ActionMessage m, ActionContext context) считывает десериализованные данные в переданный ему в качестве параметра экземпяр ActionMessage.

Пример десериализованного сообщения представлен на Рис. 3.1.

3.2.3 AMF клиент

В BlazeDS существует механизм, Java AMF Client, позволяющий совершать удалённые вызовы методов и обрабатывать ответы сервера. Преимущество использования Java AMF Client заключается в том, что сериализация и десериализация AMF сообщений, отправляемых клиентом и сервером, а также установка http соединения, полностью обеспечивается данной технологией.

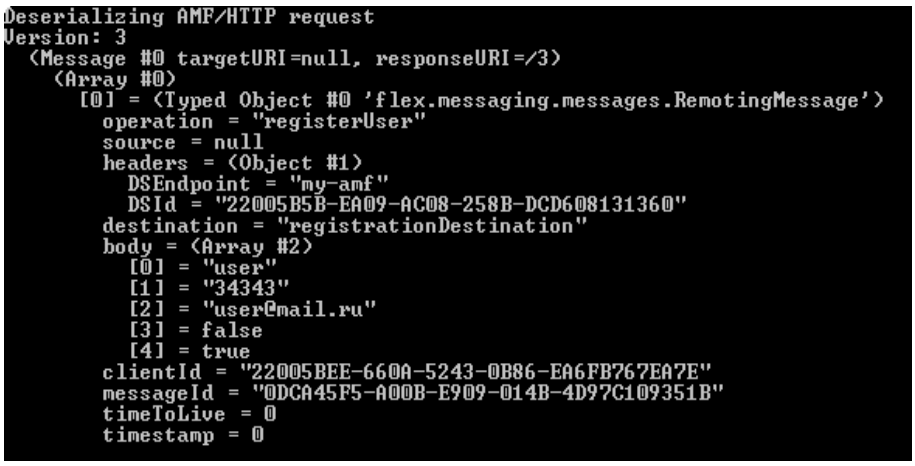

Рисунок 3.1 — Десериализованное сообщение

Классы, реализующие функционал Java AMF Client, находятся в пакете flex.messaging.io.amf.client. Основным классом является AMFConnection. Пример его работы показан ниже.

ТУТ ПРИМЕР КОДА

AMFConnection устанавливает соединение с удалённым объектом по указанному URL с помощью метода connect(). В случае успешной установки соединения метод call() отправляет AMF запрос пользователю, в качестве параметров метод принимает имя вызываемого на стороне сервера метода и его параметры, представленные в виде массива объектов.

В данном примере возможно появление двух видов исключительных ситуаций:

а) ServerStatusException — в случае появления сообщения об ошибке от сервера.

б) ClientStatusException — в случае ошибки установки соединения с сервером или при непредвиденном разрыве соединения.

3.3 Интеграция с JMeter

3.3.1 Общая архитектура JMeter

Структура проекта JMeter изображена на ...

ТУТ РИСУНОК

а) bin — содержит в себе .bat и .sh файлы для запуска JMeter, файл ApacheJMeter.jar и файлы настроек;

б) docs — директория, содержащая документацию по проекту;

в) extras — дополнительне фалйлы для утилиты ant;

г) lib — jar файлы библиотек, используемых в JMeter;

д) lib/ext — jar файлы ядра и отдельных компонентов JMeter;

е) src — исходные коды JMeter;

ж) test — юнит-тесты;

з) xdocs — xml файлы для документации (JMeter генерирует документацию из xml).

Рассмотрим подробнее содержимое директории src:

а) component — директория, содержащая общие для различных протоколов элементы, такие как визуалайзеры, соответствия и т.д;

б) core — ядро JMeter, содержит базовые интерфейсы и абстрактные классы;

в) examples — примеры, демонстрирующие использование компонентов фреймворка;

г) functions — стандартные функции, используемые всеми компонентами;

д) jorphan — утилитные классы;

е) monitor — элементы мониторинга сервера Tomcat 5;

ж) protocol — содержит реализации компонентов Jmeter для различных протоколов.

В архитектуре JMeter ядро, содержащее в себе интерфейсы и абстрактные классы,а также базовый функционал, отделёны от конкретных реализаций компонентов для различных протоколов. Это сделано для того, чтобы разработчики могли добавлять поддержку новых протоколов без сборки всего приложения. Таким образом, для того, чтобы добавить в JMeter элементы тестирования Flex приложений, нужно будет переопределить несколько базовых компонентов JMeter, собрать jar файл модуля, и поместить его в директорию lib/ext — новый функционал будет автоматически подхвачен JMeter.

Прежде чем приступить к созданию различных компонентов, опишем ряд общих правил их реализации, которые необходимы для того,

чтобы элемент правильно работал в среде JMeter. В основном это относится к графическому интерфейсу пользователя (GUI).

В JMeter код GUI элемента отделён от функционального кода элемента, поэтому реализуя новый компонент следует создавать отдельные классы для рабочего функционала и графического представления. GUI элемент, в зависимости от его предназанчения, должен расширять один из представленных ниже абстрактных классов.

а) AbstractSamplerGui
б) AbstractAssertionGui
в) AbstractConfigGui
г) AbstractControllerGui
д) AbstractPostProcessorGui
е) AbstractPreProcessorGui
ж) AbstractVisualizer
з) AbstractTimerGui

Следующим шагом является реализация метода getResourceLabel().Этот метод должен возвращать имя ресурса, представляющего компонент.

Чтобы GUI создаваемого вами тестового элемента соответсвовало стилю JMeter, следует добавить в него стандартную рамку JMeter, которая создаётся следующим образом — setBorder(makeBorder()). Также необходимо создать панель с именем элемента методом makeTitlePanel(). Обычно её располагают в верхней центральной части панели элемента.

Важным пунктом является реализация метода public void configure(TestElement el), который отвечает за отображение параметров тестового элемента в GUI. Первой строчкой в методе должен быть вызов super.configure(e), что обеспечит выполнение некоторых стандартных действий, как например отображение имени элемента. Обратное действие — передача данных из GUI в тестовый элемент — обеспечивается методом public void modifyTestElement(TestElement e), который также нужно реализовать. В имплиментацию этого метода тоже включается строчка super.configureTestElement(e).

Другой необходимый в создании GUI метод — public TestElement createTestElement(). Он должен новый экземпляр тестового элемента и передавать его в метод modifyTestElement(TestElement) в качестве параметра.

Все вышеобозначенные правила разграничения GUI и функционала элемента облегчают разработчикам реализацию графического интерфейса. Хотя программист и не освобождается от процедуры создания и размещения компонентов графического интерфейса, всё взаимодействие GUI и соответствующего ему тестового элемента обеспечивается JMeter.

3.3.2 Реализация модуля отправки AMF сообщений

Согласно одному из требований технического задания, разрабатываемое программное обеспечение должно предоставлять возможность отправки AMF сообщения серверу. Данное требование осуществляется реализацией такого компонента JMeter, как сэмплер(Sampler). Сэмлеры в JMeter предназначены для отправки запросов серверу и ожидания ответов от них.

Чтобы добавить свой собственный сэмплер в JMeter, необходимо создать класс, наследующий абстрактный класс AbstractSampler и реализующий метод public SampleResult sample(Entry e). Именно этот метод будет вызываться у каждого сэмплера, входящего в тест план во время его прогона. Метод sample должен осуществлять подключение к серверу, ожидание ответа от него и возвращать экземпляр класса SampleResult, содержащий в себе информацию о выполнении запроса.

В рамках решения этой задачи был создан класс AmfRPCSampler. В методе sample данного класса производится вызов метода удалённого объекта на стороне сервера с помощью AMF клиента (алгоритм работы AMF клиента описан в разделе 3.2.3). Метод возвращает экземпляр класса AmfRPCSamplerResult, в который по ходу выполнения метода sample записывается информация о передаваемых в запросе данных, ответ сервера, а также статус выполнения запроса (в случае, если отсет сервера получен и не содержит сообщений обошибках, запрос считается успешно выполненным).

Также метод sample класса AmfRPCSampler отвечает за одно из важных технических требований к модулю — возможность проведения полноценных нагрузочных тестов. В JMeter имитация работы с приложением большого числа пользователей одновременно осуществляется засчёт запуска тест-плана в несколько потоков и реализуемый сэмплер должен обеспечивать уникальность идентификатора сессии AMF клиента с сервером для каждого потока, выполняющего тест-план, а также гарантировать его сохранность в рамках одного потока. Данная задача решается засчёт введения в классе AmfRPCSampler статической переменной ThreadLocal<AMFConnection> AMF_CONNECTION_THREAD_LOCAL. Работа с данной переменной происходит следующим образом. В методе sample проверяется текущее значение AMF_CONNECTION_THREAD_LOCAL вызовом метода AMF_CONNECTION_THREAD_LOCAL.get(), который возвращает экземпяр AMFConnection для текущего потока. Если соединение AMFConnection для данного потока уже было создано, то дальнейшая работа осуществляется с полученным экземпляром AMFConnection. Если вызов AMF_CONNECTION_THREAD_LOCAL.get() возвращает null, то создаётся новый экземпляр AMFConnection, производится попытка подключения к серверу и значение AMFConnection записывается в AMF_CONNECTION_THREAD_LOCAL. Таким образом все элементы AmfRPCSampler, запускаемые одним потоком, будут работать с одним и тем же экземпляром AMFConnection, а остальные потоки будут использовать другие соединения.

Разработанный AmfRPCSampler также имеет графический интерфейс пользователя, реализованный в классе AmfRPCSamplerGui. AmfRPCSamplerGui создан согласно правилам реализации GUI в JMeter и расширяет класс AbstractSamplerGui. Интерфейс содержит следующие компоненты для ввода данных, необходимых для вызова метода объекта на стороне сервера:

а) endpointUrlField — поле для URL, по которому отправляется запрос;

б) amfCallField — поле для имени процедуы, которая должна быть вызвана;

в) parametersPanel — таблица параметров вызываемого метода.

3.3.3 Реализация прокси-сервера

Создание прокси-сервера необходимо для возможности записи трафика между приложением и сервером и дальнейшего использования записанных запросов в тест-плане. В рамках модуля реализован прокси сервер, который запускается по адресу localhost:port (порт указывается ползователем), обрабатывает полученные http запросы и отправляет их адресату. В случае, если тело полученного http запроса содержит AMF сообщение, происходит его десериализация с использованием AmfMessageDeserializer и создаётся экземпляр AmfRPCSampler, в который записывается информация из AMF сообщения. Все созданные во время работы прокси-сервера AMF запросы могут быть перенесены в тест план.

GUI прокси-сервера содержит поле для ввода порта, на котором сервер будет запущен, а также кнопки для запуска и остановки сервера.

3.4 Руководство пользователя

3.4.1 Установка JMeter

Для начала необходимо скачать с сайта производителя zip архив, содержащий все необходимы для установки файлы, а затем распаковать его на диске. Место установки JMeter далее будем называть JMETER_HOME.

3.4.2 Установка модуля amf-translator

Чтобы подключить к JMeter модуль amf-translator, достаточно добавить jar-файл приложения в каталог JMETER_HOME/lib/ext.

3.4.3 Запуск приложения

Приложение запускается с помощью файла jmeter.bat или ApacheJMeter.jar, находящихся в каталоге JMETER_HOME/bin.

3.4.4 Настройка прокси-сервера

После запуска в окне приложения с левой стороны нам доступно дерево элементов. Чтобы создать прокси сервер с поддержкой протокола AMF, необходимо правой кнопкой мыши кликнуть по элементу WorkBench, а затем добавить элемент AMF Proxy Server (WorkBench > Add > Non-Test Elements > AMF Proxy Server).

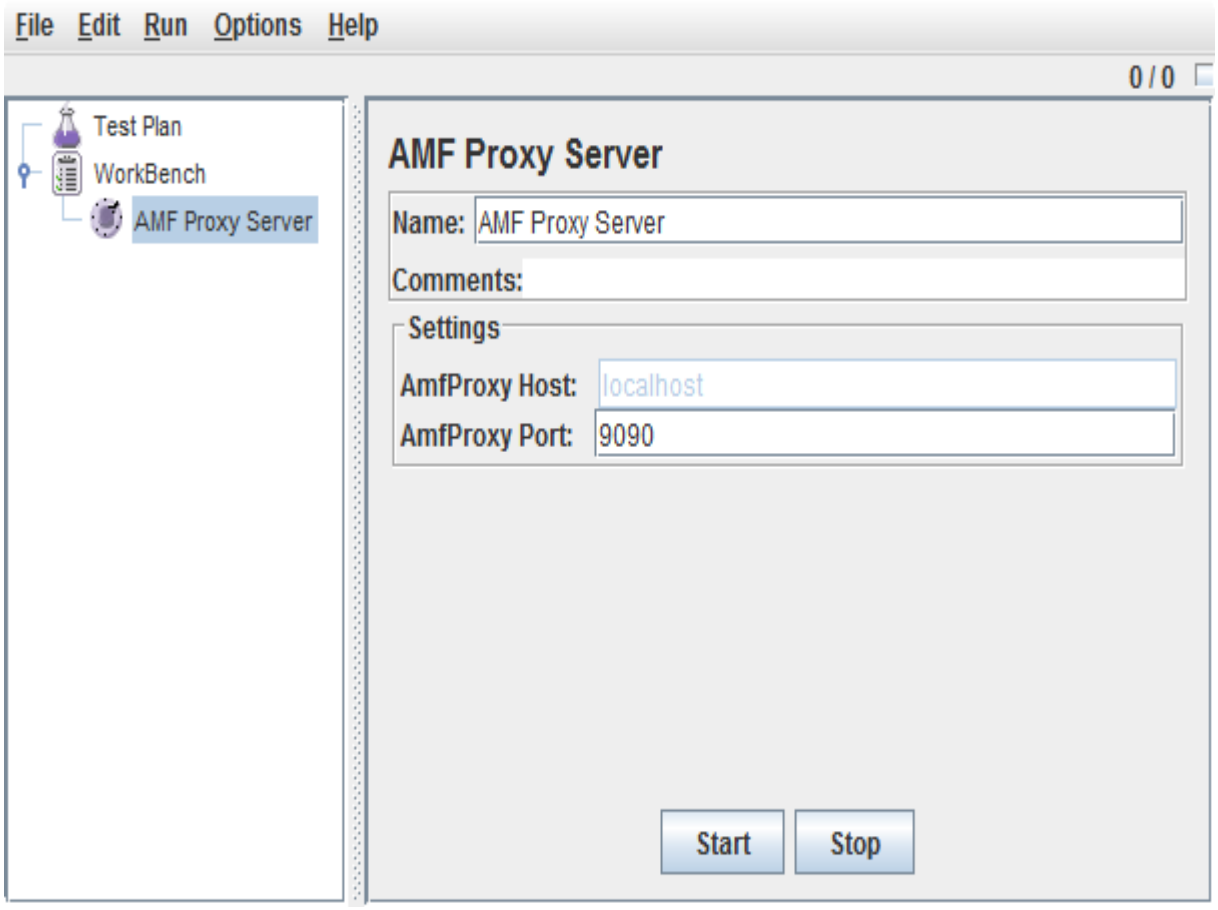

Рисунок 3.2 — Настройка прокси-сервера

В поле AmfProxy Port необходимо указать номер порта, который будет слушать наш прокси сервер. Если указать, например, 9090, то прокси-сервер будет запущен на localhost:9090. Затем точно такие же настройки прокси-сервера устанавливаются в браузере, с помощью ко-

торого будет производиться тестирование. Также стоит убедиться, что указанный Вами порт уже не занят другим приложением.

3.4.5 Запись тестового сценария

После того, как в AMF Proxy Server установлены все необходимые параметры, нажимается кнопка Start, запускающая прокси-сервер.

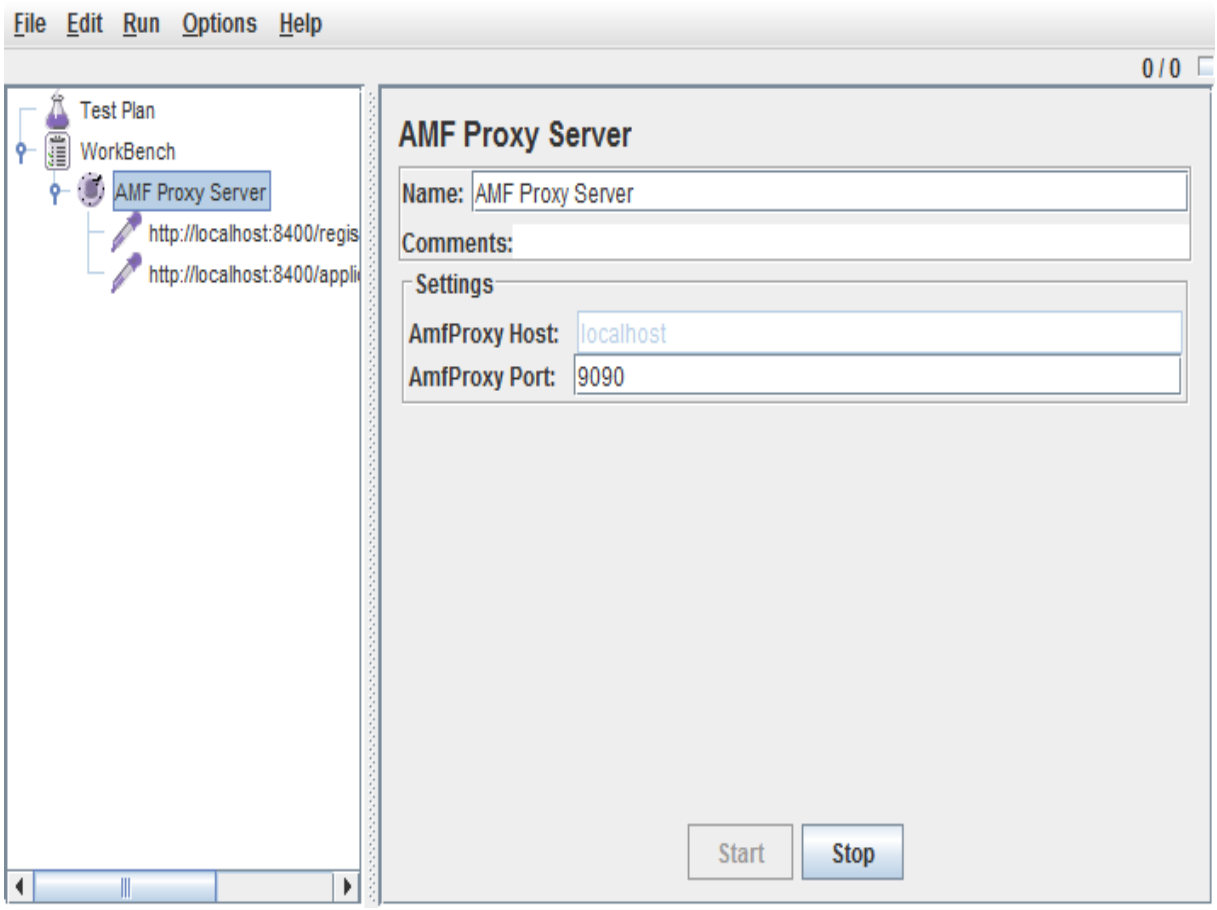

Рисунок 3.3 — Запуск прокси-сервера

Затем тестируемое приложение открывается в браузере, для которого также применены соответсвующие настройки прокси-сервера, и пользователь может выполнять с Flex приложением необходимые операции, которые будут записываться AMF Proxy Server в виде элементов AMF RPC Sampler и в дальнейшем могут быть перенесены в тест-план. Чтобы завершить запись тестовых запросов, необходимо нажать кнопку Stop. После завершения записи тестов, все перехваченные запросы отображаются в дереве элементов JMeter в качестве дочерних элементов AMF Proxy Server.

3.4.6 Создание тест-плана

Создание тест-плана в JMeter осуществляется следующим образом. В первую очередь добавляется группа потоков - Thread Group (Test Plan > Threads (Users) > Thread Group).

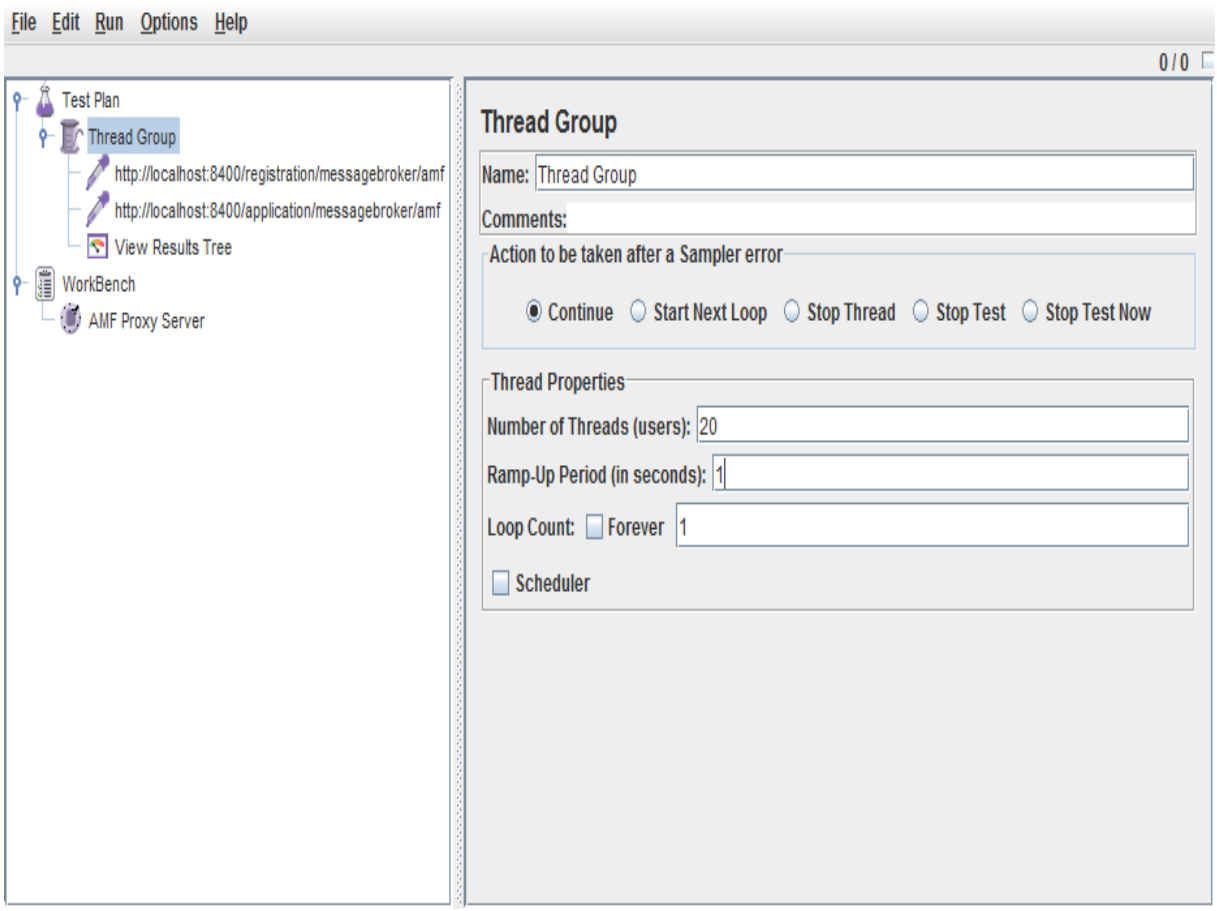

Рисунок 3.4 — Создание тест-плана

Этот элемент является ключевым в тест-плане JMeter, именно его функционал отвечает за реализацию нагрузочного тестирования — многопоточного запуска последовательности тестовых шагов. В данном элементе задается следующее:

а) Действия, которые будут производиться в случае, если в тест выполняется с ошибкой (Action to be taken after a Sampler error);

б) Число потоков, в которое будут запускаться шаги тест-плана (Number of Threads);

в) Интервал, в течение которого будет запущено указанное в предыдущем параметре число потоков (Ramp-Up Period);

г) Число повторений набора тестов (Loop Count);

д) Расписание запуска тестов (Scheduler).

Затем в Thread Group в качестве дочерних элементов добавляются шаги тестов, которые будут запускаться с указанными характеристиками. В нашем случае мы переносим элементы AMF RPC Sampler, записанные с помощью прокси.

AMF RPC Sampler представляет собой форму для вызова процедур java-объектов на стороне сервера. Чтобы вызвать метод удалённого объекта, необходимо заполнить следующие поля:

а) Endpoint Url - URL, по которому отправляется запрос

б) AMF Call - имя удалённого объекта и процедуы, которая должна быть вызвана; Например, если мы хоти вызвать у объекта registrationDestionation метод registerUser, в этом поле следует написать registrationDestination.registerUser;

в) Request Parameters - параметры метода, вызываемого у объекта (если они существуют).

На Рис. 3.5 представлен интерфейс AMF RPC Sampler.

Помимо сэмплера в тест-план также следует добавить визуалайзер результатов, чтобы иметь возможность отслеживать ход тестового сценария (Thread Group > Add > Listener). JMeter предлагает большой выбор таких элементов, для примера будем использовать View Results Tree. После того как план сформирован, он может быть сохранён. (File > Save Test Plan As...)

3.4.7 Запуск тестов

Чтобы запустить содержимое элемента Test Plan, необходимо выбрать в основном меню Run > Start. После заврешения прогона тестов результаты их выполнения можно наблюдать в View Results Tree.

В правой части визуалайзера отображается дерево элементов тест плана и статус их выполнения - если элемент подсвечен зелёным цветом, то шаг выполнен успешно. В случае с AMF RPC Sampler это значит, что соединение с сервером было установлено и вызов метода удалённого объ-

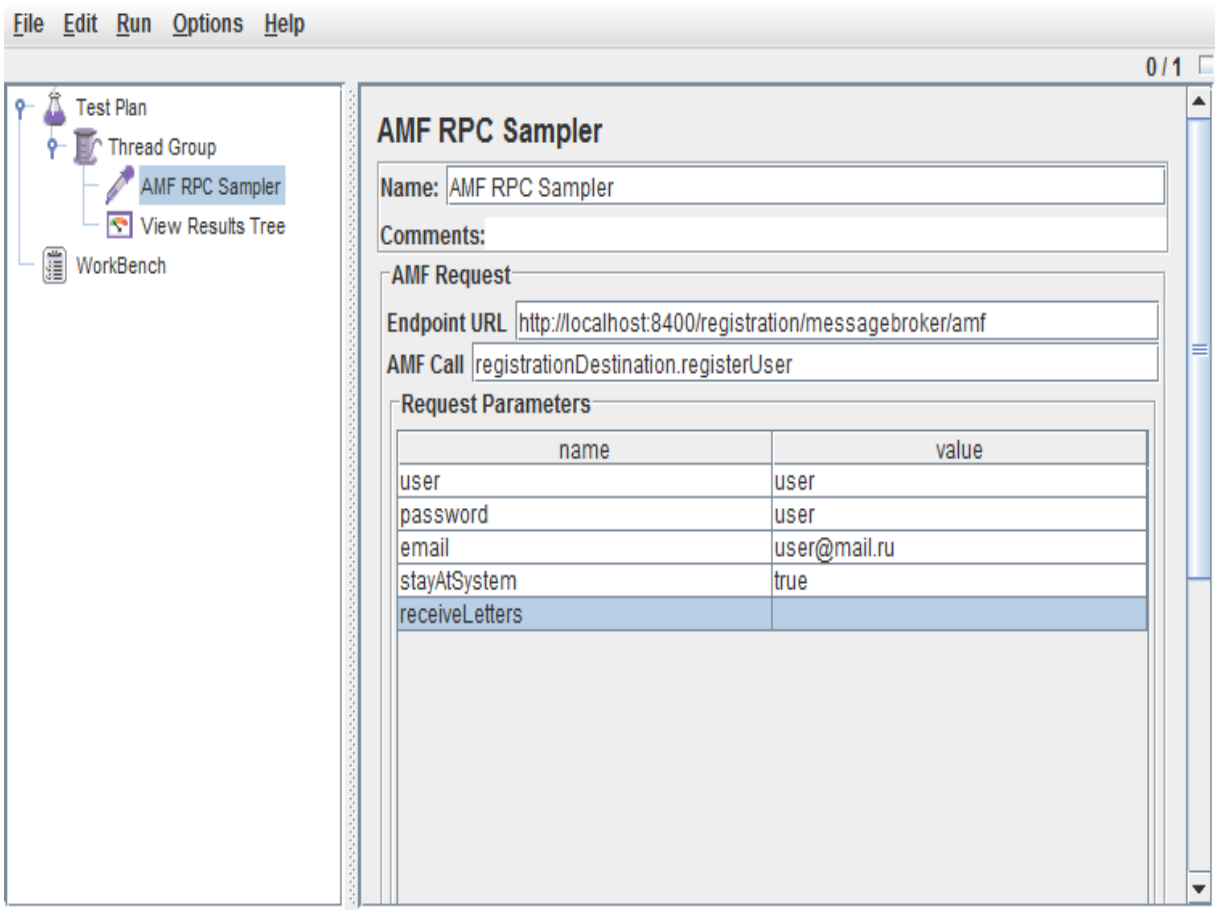

Рисунок 3.5 — Интерфейс элемента AMF RPC Sampler

екта прошёл без ошибок. Иначе тест не считантся пройденным и элемент подсвечивается красным цветом. Также на вкладках слева предоставляется возможность просмотра отправленного запроса и полученного от сервера ответа.

3.5 Методика тестирования

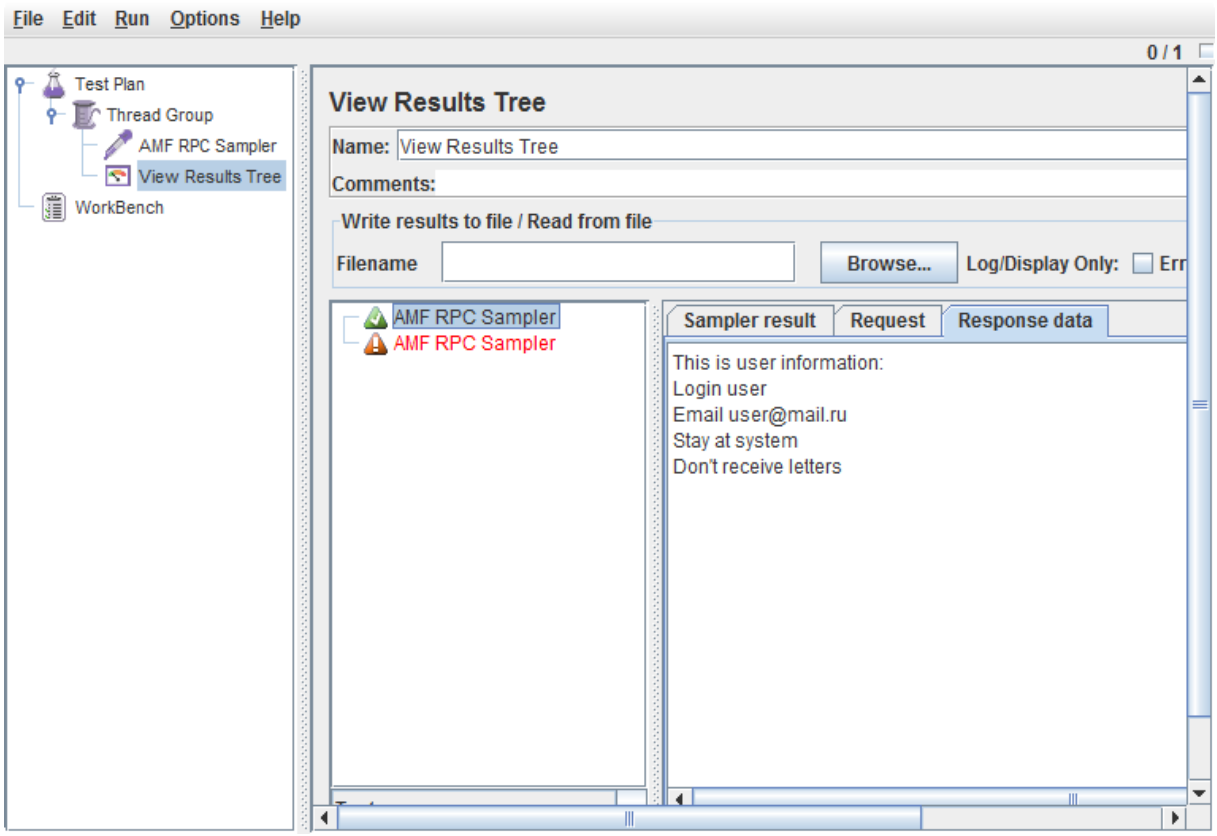

Рисунок 3.6 — Отображение результатов тестов в визуалайзере View Results Tree

4 Технико-экономическое обоснование

4.1 Концепция экономического обоснования

На сегодняшний день одним из наиболее перспективных направлений в разработке web-приложений является концепция Rich Internet Application (в дальнейшем RIA) — это приложения, доступные через сеть Интернет, обладающие особенностями и функциональностью традиционных настольных приложений. Одной из наиболее распространенных технологий разработки RIA является Adobe Flex. Flex приложения предоставляют возможность реализации клиент-серверного взаимодействия на основе бинарного формата обмена данными — AMF(Action Message Format). AMF более экономичен по трафику по сравнению с XML и позволяет передавать типизированные объекты. Как известно огромную роль в жизненном цикле программного обеспечения играет фаза тестирования. Автоматизированное тестирование является его составной частью. Оно использует программные средства для выполнения тестов и проверки их результатов, что помогает сократить время тестирования и упростить его процесс, а также может дать возможность выполнять определенные тестовые задачи намного быстрее и эффективнее чем это может быть сделано вручную. Однако использование AMF вызывает ряд трудностей для реализации автоматизации функционального и нагрузочного тестирования взаимодействия сервера и Flex клиента, связанных с бинарной природой протокола. Так как amf сообщение представляет собой совокупность байтов, разработчикам и тестировщикам трудно считывать и изменять содержащуюся в нём информацию. Отдельной проблемой является нагрузочное тестирование таких приложений — имитация работы с приложением большого количества пользователей за счёт запуска набора тестов в несколько потоков. Выходом из сложившейся ситуации является разработка программного обеспечения, в значительной степени снижающего сложность осуществления функционального и нагрузочного тестирования взаимодействия клиента и сервера по AMF-протоколу, а также разработка методики функционального и нагрузочного тестирования использованием предложенного ПО.

Целью технико-экономического обоснования является определение экономической целесообразности реализации проекта. Этапами ТЭО являются:

а) трудоемкость и календарный план выполнения НИР;

б) смета затрат на проведение НИР;

в) комплексная оценка эффективности НИР.

4.2 Рынок и план маркетинга

4.2.1 Сегментирование рынка

Сегментирование рынка состоит в выделении сегментов анализируемого рынка и оценки спроса на продукцию в каждом сегменте рынка.

Итак, выделим основные сегменты:

а) Компании, осуществляющие разработку программного обеспечения.

б) Компании, специализирующиеся на внешнем тестировании программного обеспечения.

в) Программисты, осуществляющие разработку ПО в частном порядке (для личных или коммерческих целей).

Проведем анализ требований различных групп потенциальных покупателей к продукции:

а) технический уровень, качество и надежность в эксплуатации, уровень послепродажного обслуживания;

б) технический уровень, качество и надежность в эксплуатации, уровень послепродажного обслуживания;

в) цена продукции, качество и надежность в эксплуатации.

4.2.2 Продвижение товара

Для эффективного продвижения товара, а так же для поддержания спроса необходима реклама, предоставление скидок, демо-версии для привлечения новых клиентов. Поскольку продукт является специализированным, следует давать рекламу в журналы, газеты, размещать

на сайтах и форумах той же тематики. Скидки, как правило, стимулируют уже существующих клиентов, поскольку важно, привлекая новых, не потерять уже существующих пользователей.

4.3 Производство продукта

Проводится расчет затрат, связанных с проведением НИР. Основные статьи калькуляции приведены ниже:

а) Материалы
б) Спецоборудование
в) Расходы на оплату труда
г) Отчисления на социальные нужды
д) Затраты по работам, выполняемым сторонними организациями
е) Командировочные расходы
ж) Прочие прямые расходы
з) Накладные расходы

4.3.1 Статья «Материалы»

На статью относятся затраты на сырье, основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, необходимые для выполнения конкретной НИР с учетом транспортно-заготовительных расходов. Калькуляция расходов по статье «Материалы» приведена в табл. 4.1.

Таблица 4.1 — Материалы

Наименование материалов	Коли-чество, шт	Цена, р.	Сумма, р.
Бумага формата А4	1	120	120
Заправка картриджа для принтера	1	300	300
Записываемый диск CD-RW	1	50	50
		Итого:	470
		Транспортные расходы, 15%:	70
		Всего:	540

4.3.2 Статья «Спецоборудование»

На статью «Спецоборудование» относятся затраты на приобретение (или изготовление) специальных приборов, стендов, другого специального оборудования, необходимого для выполнения конкретной НИР. Для данной НИР дополнительного спецоборудования не требуется.

4.3.3 Статья «Расходы на оплату труда»

На статью относится заработная плата научных сотрудников, инженеров и прочего инженерно-технического персонала, непосредственно занятых выполнением конкретной НИР. Эти расходы складываются из основной и дополнительной заработной платы. Средняя зарплата за один рабочий день определяется, исходя из месячного оклада и среднего количества рабочих дней за месяц, принимаемого за 22 дня.
Основная зарплата рассчитывается по формуле:

$$C_{\text{зо}} = \frac{T \cdot C_{\text{зо.мес}}}{22}, \tag{4.1}$$

где Т – трудоемкость выполнения работ по НИР, $\text{С}_{\text{зо.мес}}$ – месячный оклад.
Дополнительная зарплата рассчитывается по формуле:

$$C_{\text{зд}} = \frac{C_{\text{зо}} \cdot \text{Н}_{\text{д}}}{100}, \tag{4.2}$$

где $C_{\text{зо}}$ – основная зарплата, $\text{Н}_{\text{д}}$ – норматив дополнительной зарплаты.
Производим расчет основной и дополнительной зарплат на основании следующих данных:

а) трудоемкость выполнения работ исполнителем $\text{Т}_{\text{исп}} = 106$ чел.-дн.;

б) трудоемкость выполнения работ СНС $\text{Т}_{\text{рук}} = 7$ чел.-дн.;

в) месячный оклад инженера $\text{С}_{\text{зо.мес.исп}} = 15\ 000$ р.;

г) месячный оклад руководителя $\text{С}_{\text{зо.мес.рук}} = 20\ 000$ р.;

д) норматив дополнительной зарплаты $\text{Н}_{\text{д}} = 12\%$.

$$\text{С}_{\text{зо.исп}} = \frac{106 \cdot 15000}{22} = 72272\text{р.}, \tag{4.3}$$

$$\text{С}_{\text{зо.рук}} = \frac{7 \cdot 20000}{22} = 6363\text{р.}, \tag{4.4}$$

$$\text{С}_{\text{зо}} = 72272 + 6363 = 78635\text{р.}, \tag{4.5}$$

$$\text{С}_{\text{зд.исп}} = \frac{72272 \cdot 12}{100} = 8672\text{р.}, \tag{4.6}$$

$$\text{С}_{\text{зд.рук}} = \frac{6363 \cdot 12}{100} = 763\text{р.}, \tag{4.7}$$

$$\text{С}_{\text{зд}} = 8672 + 763 = 9435\text{р.} \tag{4.8}$$

Общие расходы на оплату труда составляют 88 070 р.

4.3.4 Статья «Отчисления на социальные нужды»

На статью относят затраты, связанные с выплатой единого социального налога. Данная статья рассчитывается пропорционально зарплате в размере 26,2%:

а) федеральный бюджет – 20%;
б) фонд социального страхования – 3.2%;
в) фонд обязательного медицинского страхования – 2,8%;
г) страхование от несчастных случаев – 0,2%.

$$C_{\text{зд}} = \frac{P_{\text{от}} \cdot \text{Н}_{\text{сн}}}{100}, \tag{4.9}$$

где $\text{Н}_{\text{сн}}$ – суммарный норматив отчислений на социальные нужды – 26,2

$$\text{С}_{\text{сн}} = \frac{88070 \cdot 26{,}2}{100} = 23074\text{р.}, \tag{4.10}$$

4.3.5 Статья «Затраты по работам, выполняемым сторонними организациями»

Расходов на работы, выполняемые сторонними организациями, не существует.

4.3.6 Статья «Командировочные расходы»

На статью относятся расходы на все виды служебных командировок работников, выполняющих задания по конкретной НИР. В данной НИР расходов, связанных со служебными командировками, нет.

4.3.7 Статья «Прочие прямые расходы»

На статью относятся расходы на получение специальной научно-технической информации, платежи за использование средств связи и коммуникации, а также другие расходы, необходимые для проведения НИР. На всех этапах работы требуется выход в Интернет. Стоимость работы в Интернете составляет 350 р. в месяц. Затраты на Интернет на весь период составляют:

$$\text{С}_\text{и} = \frac{350}{22} \cdot 106 = 1686\text{р.}, \tag{4.11}$$

Также требуется использование телефона. Примем эти расходы ориентировочно равными $\text{С}_\text{т} = 300$ р. Следовательно, суммарно прочие прямые затраты составляют:

$$\text{С}_\text{пнз} = \text{С}_\text{и} + \text{С}_\text{т} = 1686 + 300 = 1986\text{р.}, \tag{4.12}$$

4.3.8 Статья «Накладные расходы»

В статью включаются расходы на управление и хозяйственное обслуживание НИР.

$$\text{С}_\text{нр} = \frac{P_\text{от} \cdot H_\text{нр}}{100}, \tag{4.13}$$

где $\text{Н}_\text{нр}$ – норма накладных расходов, равная 20%.

$$\text{С}_\text{нр} = 88070 \cdot \frac{20}{100} = 17614\text{р.} \tag{4.14}$$

На основании полученных данных в табл. 4.2 приведена калькуляция себестоимости разработки.

Таблица 4.2 — Смета затрат на проведение НИР

№ п/п	Наименование статьи	Сумма, р.
1	Материалы	540
2	Спецоборудование	—
3	Расходы на оплату труда	88070
4	Отчисления на социальные нужды	23074
5	Затраты по работам, выполняемым сторонними организациями	—
6	Командировочные расходы	—
7	Прочие прямые расходы	1986
8	Накладные расходы	17614
	Себестоимость НИР:	131 284

4.4 Организационный план проекта

Трудоемкость выполнения работы исполнителем составляет 106 чел.-дней, а руководителем 7 чел.-дней. Общая продолжительность выполнения данной НИР 113 дней (чуть больше 15 недель).

4.5 Комплексная оценка эффективности НИР

4.5.1 Научно-технический эффект разработки

Концепции и технологии, используемые при разработке web-приложений, постоянно развиваются и совершенствуются - оптимизируется использование ресурсов и времени, улучшаются возможности по отображению предоставляемой информации, динамичность и их интерактивность.

Flex является одним из самых распространённых и перспективных инструментов разработки web-приложений. Разрабатываемый в дипломном проекте программный модуль поможет обеспечить контроль качества Flex приложений, а также повысит эффективность тестирования за счёт автоматизации тестовых сценариев.

4.5.2 Экономический эффект

Выгоды, которые может получить потребитель от использования разрабатываемой продукции:

а) Повышение стабильности разрабатываемого приложения за счёт обеспечения качественного тестирования.

б) Повышение скорости тестирования за счёт автоматизации труда инженеров по качеству.

в) Понижение затрат на тестирование, т.к. внедрение разрабатываемого модуля позволяет отказаться от использования платных средств автоматизированного тестирования.

4.5.3 Расчет потребности в начальных инвестициях

Потребность в начальном капитале определяется средствами, израсходованных на НИР – 131 284 р., а также дополнительными средствами, необходимых для приобретения ПЭВМ и принтера. При стоимости ПЭВМ 20 000 р., сроке ее службы – 5 лет, времени копирования программного обеспечения 0,5 ч. потребность в ПЭВМ составит:

5 Охрана интеллектуальной собственности

5.1 Интеллектуальная собственность

Согласно определению интеллектуальной собственности, принятому в российском законодательстве, а также на основании определения Стокгольмской конференции от 14 июля 1967 г., компьютерные программы относятся к объектам интеллектуальной собственности. Компьютерным программам предоставляется охрана нормами авторского права как литературным произведениям в соответствии с Бернской конвенцией. В Российской Федерации вопросы предоставления правовой охраны программам регулируются Гражданским кодексом РФ, Часть 4 (ГК РФ Ч.4).

5.2 Программа для ЭВМ

Под программой для ЭВМ понимается «... представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определенного результата». Кроме того, в понятие программы для ЭВМ входят «...подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения» [2, ст. 1261]. С точки зрения программистов и пользователей программа для ЭВМ представляет собой детализацию алгоритма решения какой-либо задачи и выражена в форме определенной последовательности предписаний, обеспечивающих выполнение компьютером преобразования исходных данных в искомый результат.

Можно выделить следующие объективные формы представления программы для ЭВМ:

а) исходная программа (или исходный текст) — последовательность предписаний на алгоритмическом языке высокого уровня, предназначенных для автоматизированного перевода этих предписаний в последовательность команд в объектном коде;

б) рабочая программа (или объектный код) — последовательность машинных команд, т. е. команд, представленных на языке, понятном ЭВМ;

в) программа, временно введенная в память ЭВМ, — совокупность физических состояний элементов памяти запоминающего устройства ЭВМ (ОЗУ), сохраняющихся до прекращения подачи электропитания к ЭВМ;

г) программа, постоянно хранимая в памяти ЭВМ, — представленная на языке машины команда (или серия команд), выполненная в виде физических особенностей участка интегральной схемы, сохраняющихся независимо от подачи электропитания.

Исходная и рабочая программы представляются в электронном виде. Правовая охрана программ для ЭВМ распространяется только в отношении формы их выражения и «... не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования» [2, ст.1259, п. 5].

5.3 Авторское право на программу для ЭВМ

Предпосылкой охраноспособности программы для ЭВМ и базы данных является их творческий характер, т. е. они должны быть продуктом личного творчества автора. Творческий характер деятельности автора предполагается до тех пор, пока не доказано обратное [2, ст. 1257].

Момент возникновения авторского права является важнейшим юридическим фактом, который устанавливается в силу создания программы для ЭВМ. «Для возникновения, осуществления и защиты авторских прав не требуется регистрация произведения или соблюдение каких-либо иных формальностей» [2, ст.1259, п.4].

Закон устанавливает, что обнародование программы не является обязательным условием для возникновения прав на нее: «Авторские права распространяется как на обнародованные, так и на необнародо-

ванные произведения, выраженные в какой-либо объективной форме ...» [2, ст. 1259, п. 3].

Таким образом, только сам факт создания программы, зафиксированной в объективной форме, является основанием возникновения авторского права на нее.

Каждая составляющая понятия использования программы для ЭВМ имеет конкретное содержание, которое также определено законом:

а) воспроизведение — «... изготовление одного или более экземпляров произведения или его части в любой материальной форме, ... в том числе запись в память ЭВМ» [2, ст. 1270, п. 2, п.п.1];

б) распространение —- предоставление доступа к произведению «... путем продажи или иного отчуждения его оригинала или экземпляров» [2, ст. 1270, п. 2, п.п.2];

в) публичный показ — «... любая демонстрация оригинала или экземпляров произведения непосредственно ... либо с помощью технических средств в месте, открытом для свободного посещения, или в месте, где присутствует значительное число лиц ...» [2, ст. 1270, п. 2, п.п.3].

В целях оповещения о своих правах правообладатель «... вправе использовать знак охраны авторского права, который помещается на каждом экземпляре произведения и состоит из следующих элементов: латинской буквы С в окружности; имени или наименования правообладателя; года первого опубликования произведения» [2, ст. 1271].

Исключительные права на программу переходят по наследству в установленном законом порядке, и их можно реализовать в течение срока действия авторского права.

Передача прав на материальный носитель не влечет за собой передачи каких-либо прав на программу для ЭВМ [2, ст.1227]. Иными словами, передача носителя информации (например, диска) с зафиксированной на нем программой третьему лицу не означает передачи каких-либо прав на эту программу.

5.4 Правообладание

Если человек (или группа людей) самостоятельно, по личной инициативе создал программу для ЭВМ, то он является одновременно и автором, и правообладателем созданного произведения, что позволяет ему по собственному усмотрению использовать эту программу (или базу данных) в личных целях, продавать, раздавать бесплатно, разрешать тиражировать и распространять или иным образом распоряжаться своими исключительными правами.

Кроме личных (неимущественных) прав автору "служебной" программы для ЭВМ (базы данных) принадлежит право на вознаграждение при условии использования работодателем созданных произведений или передачи исключительного права другому лицу. Размер и порядок выплаты этого определяется договором [2, ст. 1295, п. 2].

5.5 Нарушение прав на программу для ЭВМ и базу данных

Специфика программ для ЭВМ такова, что они очень уязвимы в смысле их незаконного использования (прежде всего, путем копирования и распространения копий). Незаконно изготовленные (скопированные) или используемые экземпляры программы для ЭВМ или базы данных называются контрафактными, а несанкционированное использование чужих программ или баз данных путем опубликования (выпуска в свет), воспроизведения (полного или частичного), распространения, иного использования считается нарушением исключительных прав на программы для ЭВМ или базы данных, т. е. нарушением авторского права.

5.6 Право на официальную регистрацию

В ст. 1262 ГК РФ Ч.4 закреплено право автора или иного правообладателя на государственную регистрацию программы для ЭВМ или базы данных: «Правообладатель в течение срока действия исключительного права на программу для ЭВМ может по своему желанию зарегистрировать такую программу для ЭВМ или такую базу данных в феде-

ральном органе исполнительной власти по интеллектуальной собственности». Исключение составляют программы для ЭВМ и базы данных, в которых содержатся сведения, составляющие государственную тайну.

Предусмотренная регистрация не является правообразующей и носит факультативный характер, т. е. с ней не связано возникновение прав на программу для ЭВМ, однако такая процедура представляется полезной по следующим соображениям.

а) Она является официальным уведомлением общественности о наличии у правообладателей прав в отношении рассматриваемых объектов.

б) Государственная регистрация содействует защите прав в случае возникновения конфликтных ситуаций при нарушении прав или установлении приоритета.

5.7 Процедура официальной регистрации

Процедура официальной регистрации программ для ЭВМ и баз данных в целом определена ст. 1262 ГК РФ Ч.4 и включает подачу заявки в федеральный орган исполнительной власти по интеллектуальной собственности (Роспатент), проверку поданных документов и собственно регистрацию. После поступления заявки на регистрацию в Роспатент проверяется наличие необходимых документов и их соответствие установленным требованиям. При положительном результате проверки сведения о программе для ЭВМ или базе данных вносятся, соответственно, в Реестр программ для ЭВМ или Реестр баз данных под уникальным регистрационным номером и выдается заявителю (здесь заявителем называют правообладателя, подавшего заявку на регистрацию программы или базы данных в Роспатент) свидетельство о государственной регистрации установленной формы, в котором указаны регистрационный номер объекта по Реестру, название программы или базы данных, имя или наименование правообладателя, фамилии авторов и дата регистрации. Сведения о зарегистрированных программах для ЭВМ и базах данных публикуются в официальном бюллетене Роспатента.

5.8 Заявка на официальную регистрацию

Состав заявки на официальную регистрацию программы для ЭВМ или базы данных (далее - Заявка) определен п. 2 ст. 1262 ГК РФ Ч.1, а также в Правилах составления, подачи и рассмотрения заявок на официальную регистрацию программ для электронных вычислительных машин и баз данных (далее - Правила).

Заявка должна относиться к одной программе или одной базе данных. При этом «Программа для ЭВМ, состоящая из нескольких программ для ЭВМ (программный комплекс), которые не могут быть использованы самостоятельно, регистрируется в целом (без регистрации каждой входящей в нее (него) программы для ЭВМ)» [3, п. 5]. Заявка должна содержать следующие документы:

а) заявление о государственной регистрации;

б) депонируемые материалы, идентифицирующие программу для ЭВМ, включая реферат;

в) документ, подтверждающий уплату государственной пошлины в установленном размере или основание для освобождения от уплаты государственной пошлины или уменьшения его размера.

В Правилах подробно описаны требования, предъявляемые к документам заявки.

Заявление на официальную регистрацию представляется отпечатанным на типографском бланке или в виде компьютерной распечатки согласно образцам, приведенным в приложениях к Правилам (формы РП и РП/ДОП).

В состав депонируемых материалов входит также реферат, который представляется в двух экземплярах отдельно от листинга программы для ЭВМ или описания структуры базы данных и не входит в их объем. Реферат должен содержать информацию, определенную в п.п. 18а) — 18и), п.21 и п.23 Правил, в полном объеме. При этом:

а) аннотация реферата должна содержать сведения, определенные п. 18г) Правил;

б) объем памяти указывается в Кбайтах или Мбайтах и определяется для программ как объем памяти, занимаемый исходным текстом программы (листингом).

5.9 Программный продукт и формы его продажи

Программный продукт — персонифицированная программа для ЭВМ или база данных, которая предназначена для самостоятельного использования конкретным пользователем в личных целях.

Коммерческая реализация (продажа) программного продукта связана с понятием использования программы для ЭВМ третьими лицами (пользователями) и осуществляется на основании лицензионного договора с правообладателем. Договор заключается в письменном виде и может определять следующие условия: способы использования, порядок выплаты вознаграждения и срок действия договора, а также территорию, на которой используется данный продукт [2, ст. 1235, 1236].

Одним из типов лицензионного договора на программу для ЭВМ является традиционный двухсторонний договор правообладателя —- лицензиара, с покупателем (пользователем) — лицензиатом, в котором определяется способы, сроки, территория использования программы или базы данных. Такие договоры составляются, как правило, при единичных продажах программного продукта, предназначенного для решения достаточно узких прикладных задач (научных, отраслевых и т. п.), при продажах программного продукта, требующего регулярного обновления и дополнения (некоторые базы данных), а также при передаче прав на тиражирование и распространение программ для ЭВМ или баз данных.

5.10 Договор на использование программы для ЭВМ

Текст договора должен содержать определенную исчерпывающую формулировку лицензионного соглашения между владельцем прав на программу для ЭВМ (далее — объект договора) и покупателем (приобретателем прав на использование объекта договора).

Заключение

В результате проделанной работы стало ясно, что ничего не ясно...

Приложение А Программный код модуля

Листинг А.1 — Код реализация декодера сообщений

```
1  package edu.leti.amf;
2
3  import flex.messaging.io.ClassAliasRegistry;
4  import flex.messaging.io.SerializationContext;
5  import flex.messaging.io.amf.ActionContext;
6  import flex.messaging.io.amf.ActionMessage;
7  import flex.messaging.io.amf.AmfMessageDeserializer;
8  import flex.messaging.io.amf.AmfTrace;
9  import flex.messaging.messages.AcknowledgeMessageExt;
10 import flex.messaging.messages.AsyncMessageExt;
11 import flex.messaging.messages.CommandMessageExt;
12 import org.slf4j.Logger;
13 import org.slf4j.LoggerFactory;
14
15 import java.io.IOException;
16 import java.io.InputStream;
17
18 /**
19  * Класс для сериализации и десериализации amf сообщений
20  *
21  * @author Tedikova O.
22  * @version 1.0
23  */
24 public class MessageDecoder {
25
26     private static final Logger logger =
    LoggerFactory.getLogger(MessageDecoder.class);
27
28     public MessageDecoder() {
29         ClassAliasRegistry registry = ClassAliasRegistry.getRegistry();
30         registry.registerAlias(AsyncMessageExt.CLASS_ALIAS,
    AsyncMessageExt.class.getName());
31         registry.registerAlias(AcknowledgeMessageExt.CLASS_ALIAS,
    AcknowledgeMessageExt.class.getName());
32         registry.registerAlias(CommandMessageExt.CLASS_ALIAS,
    CommandMessageExt.class.getName());
33     }
34
35     /**
36      * Метод считывает amf сообщение  из входного потока и преобразует его в
    экземпляр
37      * Action Message
38      *
```

```
39          * @param inputStream входной поток , содержащий amf сообщение
40          * @return сообщение, преобразованное в ActionMessage
41          * @throws IOException в случае ошибки чтениязаписи/
42          * @throws ClassNotFoundException в случае ошибки работы с классами
43          */
44         public ActionMessage getActionMessage(InputStream inputStream) throws
        IOException, ClassNotFoundException {
45             AmfMessageDeserializer deserializer = new AmfMessageDeserializer();
46             ActionMessage message = new ActionMessage();
47             ActionContext context = new ActionContext();
48             AmfTrace amfTrace = new AmfTrace();
49             SerializationContext serializationContext =
        SerializationContext.getSerializationContext();
50             deserializer.initialize(serializationContext, inputStream,
        amfTrace);
51             deserializer.readMessage(message, context);
52             logger.debug("Received message\n" + amfTrace);
53             return message;
54         }
55 }
```

Листинг А.2 — Интерфейс обработки http-запросов

```
1  package edu.leti.jmeter.proxy;
2
3  import org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase;
4  import org.apache.jmeter.samplers.SampleResult;
5  import org.apache.jmeter.testelement.TestElement;
6
7  /**
8   * Интерфейс, отвечающий за обработку http запросов.
9   *
10  * @author Tedikova O.
11  * @version 1.0
12  */
13 public interface SamplerDeliverer {
14      /**
15       * Метод обрабатывет полученный http запрос.
16       *
17       * @param sampler     http sampler, созданный на основе полученного запроса.
18       * @param subConfigs настройки тестового элемента.
19       * @param result      информация об обработке запроса.
20       */
21      public void deliverSampler(HTTPSamplerBase sampler, TestElement[]
        subConfigs, SampleResult result);
22 }
```

Листинг А.3 — Интерфейс прокси сервера

```
1  package edu.leti.jmeter.proxy;
2
3  /**
4   * Прокси представляет собой поток вычитывающий поток данных из сокета.
5   *
6   * @author Tedikova O.
7   * @version 1.0
8   */
9  public interface ProxyInterface {
10
11     /**
12      * Вызывается приложением для начала записи трафика.
13      */
14     void start();
15
16     /**
17      * Вызывается приложением для завершения записи трафика.
18      */
19     void stop();
20
21     /**
22      * Возвращает порт подключения прокси сервера.
23      */
24     int getLocalPort();
25 }
```