Здравствуйте уважаемая комиссия…итд

Тема моей магистрской диссертации – интеграция системы дистанционного банковского обслуживания и платежной системы «Рапида». По роду деятельности фирма в которой я работаю занимается автоматизацией банковских услуг.

Актуальность данной работы прежде всего заключается в том что сегодня платежи от физических лиц являются объектом особого внимания со стороны банковского розничного бизнеса, а в особенности это касается платежей в адрес торгово-сервисных предприятий или поставщиков услуг. Платежи совершаются частными лицами различными способами и по разнообразным каналам — наличными и безналичными средствами, через платежные системы и системы электронных денег, через банкоматы и операционные кассы банков, узлы почтовой связи и т. д. и т. п. Не секрет, что львиная доля этих платежей вносится наличными. Парадокс заключается в том, что даже если товар приобретается в онлайн режиме через тот или иной интернет-магазин, то в подавляющем большинстве случаев оплата производится наличными курьеру. Проблема может быть решена банковским сообществом путем предоставления физическим лицам удобного и защищенного инструмента для проведения оплат необходимых товаров и услуг безналичным способом в онлайн режиме.

В своем проекте я попыталась сравнить существующие архитектуры распределенных информационных систем применительно к банковской сфере. Также в своей работе на базе одной из существующих архитектур реализован модуль взаимодействия системы дистанционного банковского обслуживания с платежной системой «Рапида».

Немного терминологии:

Дистанционное банковское обслуживание (ДБО) — общий термин для технологий предоставления банковских услуг на основании распоряжений, передаваемых клиентом удаленным образом (то есть без его визита в банк), чаще всего с использованием компьютерных и телефонных сетей. Для описания технологий ДБО используются различные в ряде случаев пересекающиеся по значению термины: Клиент-Банк, Банк-Клиент, Интернет-Банк, Система ДБО.

Платёжная система — совокупность правил, процедур и технической инфраструктуры, обеспечивающих перевод стоимости от одного субъекта экономики другому. Обычно подразумевается, что через платёжные системы осуществляется перевод денег. С юридической точки зрения в большинстве случаев происходит перевод долга: средства, которые платёжная система должна одному из клиентов, она становится должна другому клиенту. Когда первый клиент передаёт платёжной системе свои деньги, то фиксируется сумма такой передачи, то есть сумма долга перед первым клиентом.

В своей научно-исследовательской части я сделала обзор современных средств и методов интеграции распределенных систем и попыталась спроецировать данные решения на область банковской автоматизации.

Постановка задачи

Рассмотрим взаимодействие клиента с банком при использовании им системы дистанционного обслуживания. Система ДБО обеспечивает клиенту полное управление своим счетом включая перевод средств, выпуск карт и операции со счетом. Для возможности осуществления этих действий система ДБО интегрируется с автоматизированной банковской системой (абс), разрабатываемой каждым банком отдельно. С другой стороны мы хотим обеспечить дистанционную оплату услуг, предоставляемых различными платежными системами. И если систему ДБО мы рассматриваем как фиксированную, с определенным протоколом взаимодействия, то каждая платежная система имеет свои протоколы, свои правила подключения и обработки платежей.И если необходима связь с несколькими платежными сервисами, то для осуществления ее нужен промежуточный модуль, содержащий правила и функции для взаимодействия с ДБО и поддерживающий модули для интеграции с платежными системами. Таким консолидатором выступает информационная система, разработанная в фирме где я работаю. В рамках данной системы создан единый шлюз к системе ДБО через механизм SOAP-запросов. Соответственно одной из моих задач в магистерской диссертации было разработать в рамках данной интеграционной системы шлюз к платежной системе «рапида». Как задача на научно-исследовательскую часть стояла задача анализа, классификации и оценки существующих подходов к интеграции систем применительно к данной области и выбор оптимальной архитектуры для подобной системы.

Основными наиболее популярными методами интеграции приложний являются обмен файлами. Файлы— это универсальный механизм хранения данных, встроенный в любую операционную систему и поддерживающийся любым языком программирования. Таким образом, один из самых простых способов интеграции приложений может быть основан на использовании файлов. Наиболее существенное преимущество файлов заключается в том, что разработчики интеграционного решения не нуждаются в дополнительных сведениях о внутренней реализации интегрируемых приложений. Один из наиболее существенных недостатков передачи файла заключается в возможности рассинхронизации интегрируемых систем вследствие низкой частоты обмена информацией.

Общая база данных обеспечивает согласованность хранящейся в ней информации. Все попытки изменения одного и того же фрагмента данных из нескольких различных источников будут пресекаться администратором транзакций базы данных. Одной из наибольших трудностей, присущих рассматриваемому стилю интеграции, является разработка схемы общей базы данных. С возрастанием числа обращений к общей базе данных последняя

может превратиться в узкое место интеграционного решения, что приведет к снижению производительности приложений и увеличению вероятности блокировки данных.

По сути, удаленный вызов процедуры представляет собой применение принципов инкапсуляции данных к интеграции приложений. Если приложение нуждается в получении или изменении информации, поддерживаемой другим приложением, оно обращается к нему посредством вызова функции. Таким образом, каждое приложение самостоятельно обеспечивает целостность своих данных и может изменять их формат, не затрагивая при этом оставшиеся приложения.

Оптимальное интеграционное решение должно обладать характеристиками передачи файла, обеспечивая частый, асинхронный обмен малыми порциями данных с уведомлением получателя и механизмом повторной передачи. В отличие от общей базы данных сведения о представлении и хранении информации должны быть скрыты от интегрируемых приложений. Кроме того, приложение должно иметь возможность удаленного вызова процедуры другого приложения с гарантированным отсутствием сбоя.

Асинхронный обмен сообщениями устраняет большинство недостатков распределенных систем. Для передачи сообщения не требуется одновременной доступности отправителя и получателя. Более того, сам факт асинхронного обмена данными побуждает разработчиков к созданию компонентов, не требующих частого удаленного взаимодействия.

Далее рассмотрим в качестве критериев для выбора решения по интеграции особенности реализаци и и требования к интеграции банковских систем. Одним из ключевых требований является слабая связанность модулей, отвечающих за неосновные функции системы. Это значит, что например отказ одной подсистемы не повлечет за собой отказ основных систем. Также особенностью является обязательное наличие средств информационной безопасности. Это правило, в особенности при взаимодействии с платежными системами регламентируется законом N161-ФЗ «О национальной платежной системе».

Банковские операции зачастую должны происходить с определенной частотой, которая является недостижимой для например систем интеграции, основанных на файловом обмене. Также важно и то чтобы систему можно было легко адаптировать под изменяющиеся условия законодательства в России, и легко добавлять новые возможности.

Таким образом, исходя из озвученных требований, можно выбрать близкую к оптимальной архтектуру построения информационных систем – Сервисно-ориентированная архитектура.

Термином SOA называется архитектура (парадигма построения) информационных систем, основаннных на взаимодействии слабосвязанных программ, называемых сервисами. Сервисы имеют стандартизированный интерфейс и протокол взаимодействия. Таким образом, исходя из этого определения можно выбрать технологии, подходящие под данную архитектуру: это удаленный вызов процедур и асинхронный обмен сообщениями.

Далее рассмотрим прикладную задачу, которую требуется решить в рамках магистерской диссертации. НКО «Рапида» - платежная система, ждет запросов на указанный адрес. Параметры можно передавать http-методом get согласно рекомендациям RFC и протоколу взаимодействия. Кодировка запросов ..

Канал к платежной системе защищается по протоколу SSL, с использованием клиентского и серверного сертификатов (двусторонняя аутентификация) с исползованием сертификата цифровой подписи формата X509.

Основные типы запросов, обрабатываемые пс рапида: запрос на выгрузку каталога услуг. Служит для загрузки каталога из платежной системы. Запрос является синхронным и выполняется по таймеру раз в сутки.

Запрос на проверку реквизитов платежа является также синхронным и выполняется на начальном этапе проведения платежа. Запрос синхронный, так как выполняется достаточно быстро.

Запрос на проведение платежа является асинхронным. Согласно протоколу взаимодействия с рапидой данный запрос может возвращать несколько раз статус платежа в обработке, в этом случае необходимо отправлять повторный запрос с теми же реквизитами. В этом случае обработка запроса является асинхронной и скрыта от пользователя.

Запрос на отмену платежа является синхронным.

Согласно этому протоколу, для связки платежной системы и системы дбо необходимо реализовать 2 типа запросов, синхронный и асинхронный, а также задействовать механизм гарантированной доставки сообщений и обеспечить транзакционность обработки. Этим требованиям удовлетворяют технологии построения приложений на базе веб-сервисов, в связке с сервис-шиной предприятия (доставка сообщений для модуля ДБО), а также удаленный вызов процедур для синхронных вызовов ПС Рапида и технология очередей сообщений JMS – для асинхронного взаимодействия с платежной системой.

На следующем слайде показана логическая архитектура общей интеграционной системы, на базе которой выполнялась моя работа. Основными ее компонентами является база данных Oracle 11g, с сервером приложений Weblogic 10.3.3 На сервере приложений на модуле oracle service bus базируется вся бизнес-логика обработки запросов от системы дбо и приложения-шлюзы к платежным системам. Также на сервере расположено ядро системы, отвечающее за логику и работу с базой данных. Системы, связанные с данной системой – дбо банка, отправляющее запросы в систему по протоколу SOAP, и платежные системы, инициатором запроса к которым выступает наша система. С платежными сервисами мы общаемся по протоколу SSL.

Физическая архитектура системы показана на следующем слайде. Как видно на схеме, запрос идет из локальной сети банка. Канал связи может быть IPSec VPN , в этом случае просто выстраивается подсеть, либо SSL-шифрование, тогда выполняются правила маршрутизации по определенным портам. В нашей сети, границей которой является маршрутизатор Microtik с белым ip-адресом, выполняется прокидывание запроса на определенные порты сервера приложений, где работают сервисы. Взаимодействие с платежными системами инициируется всегда нами и происходит с использованием SSL.

На следующем слайде показан фрагмент базы данных системы, отвечающий за связку банков с определенными платежными системами.

Следующий фрагмент базы данных – услуги в системе. Все услуги в итоге находятся в одной таблице services и связаны с каталогом определенного банка.

На текущем слайде показана таблица платежей, в которую периодически записываются все авансовые платежи клиентов. Таблица достаточно часто обновляется и не имеет индексов.

Диаграмма запроса каталога услуг представлена на следующем слайде. Инициатором запроса к пс выступае наша система, и после получения ответа с каталогом выполняется преобразование его в наш формат, сохранение в базу данных каталога для поставщика. По запросу от банка на обновление мы формируем каталог для банка, состоящий из услуг разных платежных систем и отдаем в банк.

При запросе на платеж выполняются следующие действия: запрос со статусом NEW поступает из платежной системы, и идет на проверку реквизитов. После успешной проверки платеж сохраняется в базу данных, а пользователю выводится предложение об окончательном подтверждении оплаты. После этого при поступлении от пользователя этого подтверждения выполняется смена статуса плаетежа, запрос в платежную систему и соответственно обработка ответа и смена статуса. Для более ясной картины диаграмма статусов платежей показана на следующем слайде. Из начального состояния платеж может уйти на проверку, далее на подтверждение, откуда он либо повторяется несколько раз, либо отправляется в очередь ошибок либо достигает какого-то финального статуса.

//видео

На этапе рабочего проектирования были разработаны модули системы и структура пакетов показана на слайде. Основным компонентом является модуль клиента к платежной системе , он связан со всеми остальными. Также в данную диаграмму включены пакеты для модульного тестирования. Пакеты, оканчивающиеся на model представляют собой модель данных (orm-отображение базы данных в объекты)

На следующем слайде показана диаграмма классов спроектированной системы. Основным классом является класс клиента , который реализует основные функции объявленные в его интерфейсе. Класс клиента вызывается модулями, где зашита бизнес-логика обработки платежа.

Классы, показанные на следующей диаграмме показывают формализованный ответ платежной системы. Я использовала библиотеку Jaxb для представления ответа в виде объекта.

Отдельно нужно сказать про защиту канала связи с платежной системой. Для обеспечения защиты необходимо было реализовать двухстороннюю авторизацию по протоколу SSL. Диаграмма этого процесса показана на рисунке. Как видно…после клиентских и серверных helo сервер отсылает клиенту сертификат для проверки а также алгоритмы шифрования и методы сжатия, о которых предстоит договориться. Клиент в ответ посылает свой сертификат, а также поддерживаемые им алгоритмы компрессии и шифрования. Далее после взаимоной аутентификации и выбора совместных параметров установление соединения прекращается, и начинают передаваться данные.

Для генерации клиентского ключа и сертификата я использовала утилиту OpenSSL, которая позволяет создавать запрос на сертификат по формату x509v3. При этом создается файл запроса на сертификат, который содержит информацию о владельце сертификата. Сам ключ, сгенерированный при этом процессе никуда не передается.

Но чтобы использовать сертификат в коде необходимо его поместить в доверенное хранилище – стандартизированный файл формата pkcs#12. Создание хранилища показано на слайде.

После написания модулей-связок с рапидой встает вопрос о том как система дбо узнает о наличии нового модуля .напомню, что система дбо связана с нашей информационной системой посредством сервис-шины oracle service bus. Сервис-шина хорошо подходит для интеграции корпоративных приложений и осуществляет передачу сообщения между сервисами.

Для настройки нового модуля мне было необходимо дописать на сервис-шине небольшой прокси-сервис к своему модулю. Маршрутизация с использованием osb выполнянется достаточно просто:…

После разработки всех основных модулей необходимо было выполнить функциональное/нагрузочное тестирование . Для этого я применяла утилиту jmeter, позволяющую создавать тест-проекты для самых разных типов взаимодействия. На первом скриншоте вы видите настройки сценария тестирования: количество выполняемых потоков, период повтора запроса, действия при неудаче и т.д. На втором скриншоте – сам запрос с тестовыми данными и адрес сервиса.

Результаты тестирования показаны на следующем слайде. Часть запросов намеренно выполнялась с неверными тестовыми данными чтобы удостовериться что сценарий работает. Выводы по тестир.

Таким образом основными результатами моей магистерской диссертации стали:

1. Сравнительный анализ методов построения распределенных систем и применение его к области банковской автоматизации
2. Проектирование модуля взаимодействия с платежной системой «рапида»
3. Диаграммы классов/последовательностей UML
4. Реализация спроектированного модуля на языке Java с использованием сервис-шины
5. Отладка и тестирование системы.

В настоящее время данная подсистема введена в тестовую эксплуатацию.

Спасиба))