САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра Технологии Программирования

Иванов Кирилл Андреевич

Выпускная квалификационная работа бакалавра

Управление данными в многофилиальной системе тестирования знаний

Направление 010400

Прикладная математика и информатика

Научный
руководитель:

ст. преподаватель

Севрюков С.Ю.

Санкт-Петербург

2016

Оглавление

Введение

Центр языкового тестирования СПбГУ осуществляет государственное тестирование иностранных граждан по русскому языку, истории России и основам законодательства Российской Федерации. С 1 января 2015 комплексный экзамен является обязательным для иностранных граждан, желающих оформить разрешение на работу, патент, разрешение на временное проживание или вид на жительство. За год эксперты Центра тестирования СПбГУ проэкзаменовали более 200 000 мигрантов из стран ближнего и дальнего зарубежья.

Центром языкового тестирования была проведена масштабная работа по разработке тестовых заданий, созданию и внедрению программного обеспечения для проведения тестирования, совершенствованию организационного аспекта — оптимизации процедуры тестирования и анализа результатов, а также сокращению сроков выдачи сертификатов до трёх рабочих дней. Решение поставленных задач стало возможным во многом благодаря автоматизации процесса проведения экзамена.

Экзамен проводится с помощью программного комплекса «СТКПлюс». Это современный программный комплекс, позволяющий выполнять задания субтестов, оценивать результаты, автоматически формировать тест из пула вопросов, редактировать задания, регистрировать кандидатов и организовать документооборот, а также получать различные виды статистических данных по любой имеющейся информации о кандидатах и выполненным ими заданиям. На компьютере выполняются субтесты «Лексика. Грамматика», «Чтение», «Аудирование», «Говорение», «История России», «Основы законодательства РФ».

Система тестирования является многофилиальной. Под филиалом подразумевается класс, где установлен программный комплекс.

Сам программный комплекс выполнен в виде двух модулей для запуска на Windows системах — модуль тестирования и модуль управления. Первый модуль устанавливается на каждый компьютер, на котором тестируемый будет отвечать на вопросы. Второй модуль устанавливается на сервере филиала и служит для управления тестированием и анализа данных.

Следующим звеном в комплексе выступает центр управления (центр обработки данных). Он находится в Санкт-Петербурге и должен собирать все данные о тестировании со всех классов. Однако в текущей реализации системы передача данных в центр не автоматизирована и проводится вручную сотрудниками системы. При увеличении числа филиалов, это становится сложной задачей.

Передача файлов устных ответов с компьютера для тестирования на сервер филиала также обладает рядом недостатков. Устные ответы тестируемых записываются и сохраняются сразу на сервере филиала в общую сетевую папку, предварительно созданную на сервере филиала. Из-за ограничения на количество подключений к общей сетевой папке в некоторых ОС Windows, тестирование невозможно провести одновременно с более 20 участниками. Также есть проблемы с отказоустойчивостью. Например, при записи ответа может возникнуть сбой в сети, и ответ не будет утерян.

На рисунке 1 схематично представлена архитектура программного комплекса.

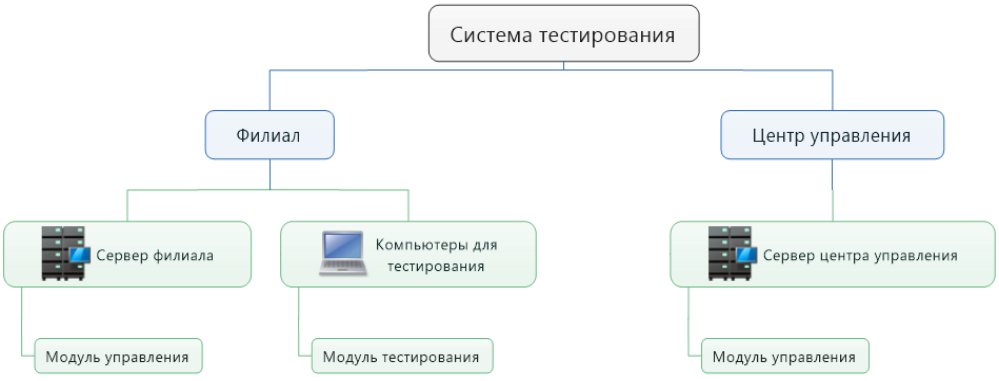

Рис. 1. Архитектура СТКПлюс

Постановка задачи

Целью данной работы является улучшение существующего варианта управления данными в многофилиальной системе тестирования знаний СТКПлюс.

Для достижения данной цели были поставлены следующие задачи:

1. Ознакомиться с инструментами для автоматизированного управления конфигурациями;
2. Изучить архитектуру системы СТКПлюс;
3. Ознакомиться с существующими инструментами по синхронизации файлов;
4. Разработать архитектуру прототипа решения управления файлами;
5. Разработать прототип приложения по управлению файлами;
6. Провести тестирование приложения по управлению файлами;
7. Ознакомиться с инструментами для синхронизации баз данных; Сделать обоснованный выбор инструмента.

Обзор литературы

В этом разделе приведен обзор некоторых книг и статей, наиболее активно используемых автором при написании работы.

Создание архитектуры программы или как проектировать табуретку [9]

Статья описывает принципы построения программного обеспечения с «хорошей» архитектурой. Автор указывает на то, что в основе хорошей архитектуры лежит декомпозиция программы на подсистемы (функциональные модули, сервисы, слои, подпрограммы) и организация их взаимодействия друг с другом и внешним миром. Далее автором описаны преимущества, которые можно получить благодаря декомпозиции. Также в статье содержится информация о том, как правильно производить декомпозицию программы и как ослабить связность между модулями. В конце статьи автором приводится список источников, используемых им при написании, а также рекомендуемый список литературы и ресурсов.

Pro Sync Framework [14]

Книга раскрывает тонкости при работе с Microsoft Sync Framework — многофункциональной платформой синхронизации, предназначенной для создания решений по синхронизации данных между приложениями, сервисами и устройствами. Книга учит работать со встроенными инструментами синхронизации Sync Framework, а также учит создавать свои, для тех типов данных, которые по умолчанию не поддерживаются.

Training Kit Exam 70-462: Administering Microsoft SQL Server 2012 Databases [15]

Книга является подробным руководством по управлению базами данных Microsoft SQL Server 2012. В ней содержатся пошаговые инструкции для установки и настройки SQL Server 2012, создания объектов баз данных, секционирования, зеркального отображения баз данных, создания

моментальных снимков баз данных, реализации доставки журналов, репликации и работы с такими компонентами, как Database Mail, Service Broker, SQL Server Agent и Full Text Search и другое. Благодаря этой книге можно подготовиться к экзамену 70-462 от Microsoft.

Глава 1. Автоматизация развертывания и настройки филиала

Развертывание системы на удаленный филиал, а также ее администрирование, осуществляется администраторами системы посредством подключения к серверу филиала с помощью программного обеспечения для удалённого контроля компьютеров TeamViewer.

По мере увеличения числа филиалов их администрирование таким образом становится сложной задачей. Для ее решения часто применяются инструменты по управлению конфигурациями. Инструменты такого характера позволяют составлять поэтапные алгоритмы развертывания окружения системы и вносить изменения в конфигурации используемых компонент.

Для того, чтобы разрабатываемое решение по управлению данными имело возможность управляться такой системой, автором был проведен анализ одного из самых актуальных средств конфигурационного управления — Puppet.

1.1. Особенности Puppet

Puppet — клиент-серверная система, которая состоит из сервера-управления и подчиненных узлов. На сервер управления устанавливается Puppet Master, на подчиненные узлы Puppet Agent.

Описание каждого узла хранится на сервере-мастере в текстовом файле (манифест в терминологии Puppet). Каждые полчаса (по умолчанию) узел подключается к серверу по зашифрованному каналу и получает от него описание конечного состояния, сверяет его с текущим и, если оно и/или описанное состояние изменилось, производит переконфигурирование системы.

1.2. Настройка Puppet

Минимальные системные требования для установки сервера Puppet Master следующие [1]:

1. Двухъядерный процессор;
2. 1 GB оперативной памяти;
3. ОС Debian, Fedora, Ubuntu.

Для работы сервера достаточно внести в его конфигурацию информацию об адресе сервера в сети. Запуск сервера производится командой puppet master.

В отличие от Puppet Master, Puppet Agent может быть установлен на Windows систему. Минимальные системные требования совпадают с системными требованиями самой ОС. С учетом того, что на серверах филиала установлена ОС Windows, все дальнейшее описание Puppet Agent ведется для нее.

Для работы с установленным Puppet Master достаточно указать в конфигурационном файле Puppet Agent имя узла и адрес сервера с Puppet Master.

1.3. Создание манифеста

Как было указано выше, описание каждого узла хранится текстовом файле — манифесте. Для описания конечного состояния узлов Puppet использует собственный язык описания на базе Ruby — DSL (“Domain Specific Language”). Ключевым элементом языка являются так называемые ресурсы, с помощью которых происходит описание того, к какому виду должен быть приведен один из компонентов ОС.

Как следует из документации СТКПлюс, для развертывания системы на сервере филиала, необходимо применить следующие шаги:

1. Установить SQL Server 2014 Express WT;

2. Настроить разрешающее правило в брандмауэре для порта 1433;
3. Создать обычного локального пользователя disk с паролем disk;
4. Настроить общую папку (рекомендуется C:/share/) на чтение и запись для пользователя disk;
5. Создать сетевой диск (буква диска - T:/) на созданную общую папку командой: net use T: \\pc-name\share /SAVECRED;
6. Включить протокол TCP/IP в диспетчере конфигурации сервера и в его свойствах, во вкладке ip-адреса, в секции IPAll в поле “TCP-порт” указать 1433, поле “Динамические TCP-порты” сделать пустым;
7. Выполнить заданный скрипт БД.

Автором работы был написан манифест для автоматизации данных шагов. С манифестом можно ознакомиться в приложении 1.

1.4 Выводы

После знакомства с Puppet можно сделать следующие выводы:

1. Большинство шагов по развертыванию решается через командную строку Windows;
2. Приложение, подлежащее настройке Puppet, должно хранить свою конфигурацию в открытом виде, например, в текстовом файле настроек.
3. Разработанный в ходе изучения манифест Puppet позволяет автоматизировать процесс развертывания филиала.

Глава 2. Управление аудио-ответами

2.1. Исследование вопроса

В системе СТКПлюс помимо вопросов, требующих от тестируемого выбор варианта ответа среди предложенных, есть также вопросы подразумевающие устный ответ. При этом речь тестируемого записывается и сохраняется в отдельный файл. С целью контроля дисциплины во время тестирования, речь записывается и в вопросах, не требующих устного ответа.

В системе СТКПлюс, аудио-ответы должны быть доступны на локальном сервере управления филиалом и на сервере центра управления. В первом случае для проверяющих, которые прослушивают ответы и выставляют оценки, во втором — для аудита качества проводимого тестирования и выборочной повторной проверки результатов.

Однако в текущей реализации системы передача файлов в центр обработки данных отсутствует, а аудио-ответы сразу сохраняются на сервере филиала. Для этого встроенным средствами Windows на нем создается общая сетевая папка. На компьютере тестируемого она подключается как виртуальный диск, на который модуль тестирования записывает свои данные.

Однако у такой реализации есть следующие недостатки:

1. Низкая отказоустойчивость;
2. Ограничение на количество подключений к общей сетевой папке.

Низкая отказоустойчивость. Сервер филиала может зависнуть во время записи речи и аудио-файл будет безвозвратно утерян. Аналогичная ситуация может произойти и в случае сбоя в канале связи.

*Ограничение на количество подключений к общей сетевой папке***.** Все версии Windows, кроме северных, имеют ограничение на количество одновременных подключений к общей сетевой папке. Например, в ОС

Windows 7 это ограничение равно 20. Ввиду этого, тестирование не может продводиться с более, чем 20 участниками.

На не серверных версиях ОС Windows доступ к общей сетевой папке могут иметь до 20 устройств [2].

На основе указанного выше, автором предлагается выполнить следующее:

1. Модифицировать текущей способ передачи файлов на сервере филиала;
2. Реализовать синхронизацию файлов между сервером филиала и центром обработки данных.

Основная идея для модификации существующего способа сохранения файлов на сервере филиала заключается в том, чтобы сначала записывать речь на компьютер тестируемого, а уже впоследствии передавать эти данные на сервер филиала. На рисунке 2 представлена иллюстрация этой концепции.

Рис. 2. Схема модифицированного способа передачи файлов внутри филиала

Таким образом обе поставленные задачи подразумевают под собой передачу файлов по двум маршрутам:

1. Между компьютером тестируемого и сервером филиала;
2. Между сервером филиала и сервером центра обработки данных.

2.2. Обзор инструментов для синхронизации файлов

Следует отметить, что на рынке есть немало приложений, позволяющих решать задачу синхронизации между двумя каталогами. Но использование готовых решений не оправдано в рамках системы СТКПлюс, поскольку требуется плотная интеграция в существующую архитектуру системы. Однако с целью формирования представления о лучших практиках в области синхронизации файлов автором был выполнен анализ существующих инструментов. Предлагается сравнить их по следующим критериям:

1. Количество поддерживаемых протоколов передачи;
2. Синхронизация по расписанию;
3. Управление синхронизацией из другого приложения;
4. Возможность синхронизации независимо от статуса пользователя, т. е. работа в условиях, когда пользователь не произвел вход в систему;
5. Отказоустойчивость

Exiland Backup Professional

Официальный сайт: http://www.exiland-backup.com.

Распространение: Shareware.

Описание: Exiland Backup - это программа резервного копирования, разработанная компанией Exiland Software. Как утверждает разработчик, утилита может использоваться как дома, так и в организациях на рабочих ПК и серверах. И хотя программа предназначена для создания резервных копий, она поддерживает также и синхронизацию каталогов. Работа с Exiland Backup

основана на заданиях. Задание включает в себя параметры синхронизации - папка-источник, папка-получатель и т.д. [3]

Для организации, где необходимо копировать данные с некоторого количества машин, разработчики Exiland Backup предлагает настроить копирование данных на всем выбранном множестве машин, либо настроить на каждом ПК выкладывание локальных файлов на сетевой диск или сервер. [4]

Поддерживаемые протоколы:

1. SMB;
2. FTP.

Синхронизация по расписанию: присутствует.

Возможность синхронизации независимо от статуса пользователя: присутствует. Программа может быть запущена как служба Windows.

Управление синхронизацией из другого приложения: управлять синхронизацией из другого приложения можно из командой строки. Однако поддерживается лишь запуск определенного задания синхронизации. Создать новое задание или импортировать задание из файла из командной строки невозможно.

Отказоустойчивость: в случае критической ошибки сеанс синхронизации будет перезапущен.

SmartSync Pro

Официальный сайт: http://www.smartsync.com/ru/smartsyncpro/

Распространение: Shareware

Описание: Как уверяет разработчик, SmartSync Pro – это комплексное решение для бэкапа и синхронизации, предназначеное для индивидуальных и корпоративных пользователей компьютеров. Данная программа отличается простым и приятным интерфейсом, качественно локализована на русский

язык. Активная поддержка российских пользователей связана с тем, что руководитель SmartSync Software Дмитрий Ситников сам является программистом из России. В отличие от Exiland Backup вместо заданий присутствуют профили, суть которых аналогична. [5]

Поддерживаемые протоколы:

1. SMB;
2. FTP;
3. WebDAV.

По всем остальным требованиям программа полностью идентична Exiland Backup.

SyncBackPro

Официальный сайт: http://allwaysync.com/

Распространение: Shareware

Описание: Еще одна программа для создания резервных копий (бэкапов) от компании 2BrightSparks, которая поддерживает синхронизацию папок. [6]

Поддерживаемые протоколы:

1. SMB;
2. FTP;
3. MTP (Media Transfer ProtocolMedia Transfer Protocol);
4. Также поддерживаются облачные хранилища: Google Drive, OneDrive, Box, Dropbox, Amazon S3, Amazon Cloud Drive, Azure, Google Storage, Office 365 (OneDrive for Business and SharePoint), SugarSync и файлы на почтовом сервисе.

Управление синхронизацией из другого приложения: управлять синхронизацией из другого приложения можно из командой строки. Поддерживается как запуск определенного задания синхронизации, так и

импорт заданий. Однако формат, в котором они хранятся - закрытый, и возможности отредактировать файл задания текстовым редактором нет.

Возможность синхронизации независимо от статуса пользователя: Присутствует. Но в отличие от рассмотренных ранее приложений, установить SyncBackPro, как службу нельзя. Однако можно добавить задание по синхронизации во встроенный планировщик Windows.

В остальном программа аналогична предыдущим вариантам.

GoodSync

Официальный сайт: http://www.goodsync.com/ru

Распространение: shareware

Описание: Данная утилита примечательна тем, что она создана разработчиками популярной программы для сохранения паролей RoboForm [7].

Поддерживаемые протоколы:

1. SMB;
2. FTP;
3. MTP (Media Transfer ProtocolMedia Transfer Protocol);
4. WebDAV.
5. Также поддерживаются облачные хранилища: Google Drive, OneDrive, Box, Dropbox, Amazon S3, Amazon Cloud Drive, Azure, Google Storage, Office 365 (OneDrive for Business and SharePoint), SugarSync и собственная служба GoodSync Connect

Отдельно стоит рассказать про GoodSync Connect [8]: это P2P-сеть, которая позволяет синхронизировать данные на нескольких компьютерах через Интернет. Таким образом, не нужно соединять компьютеры в локальную сеть или обходиться для переноса USB-устройством, при этом файлы не будут

храниться в «облаке». Чтобы использовать GoodSync Connect, нужно сконфигурировать компьютеры, создав учетную запись в системе.

Управление синхронизацией из другого приложения: управлять синхронизацией из другого приложения можно из командой строки. Поддерживается как запуск определенного задания синхронизации, так и импорт заданий. В отличие от предыдущего варианта формат заданий представляет собой текстовой файл, содержимое которого напоминает xml-документ. На официальном сайте есть документация данному формату.

Синхронизация по расписанию: присутствует.

Возможность синхронизации независимо от статуса пользователя: присутствует. Программа может быть запущена как служба Windows.

Выводы

Все программы справляются со своей задачей, различия кроются в деталях. В целом, можно сделать следующие выводы:

1. Все предложенные варианты позволяют запустить задание/профиль из командной строки. Однако только GoodSync хранит свои задания в отрытом формате и позволяет сделать импорт задания из командной строки. Это важное преимущество при использовании систем управления конфигурациями;
2. Разработчики по-разному реализует возможность синхронизации независимо от статуса пользователя. Exiland Backup Professional, GoodSync, SmartSync Pro устанавливаются как служба, а SyncBackPro использует планировщик заданий Windows. При разработке приложения можно также применить один из этих способов;
3. Для организации, где необходимо копировать данные с некоторого количества машин, разработчики Exiland Backup предлагает настроить на каждом ПК выкладывание локальных файлов на сетевой диск или сервер. Такое архитектурно решение можно использовать и в собственном решении.

Эти выводы будут использованы при построении собственного решения.

2.3. Архитектура решения

Одним из главных критериев при разработке больших систем считается задача снижения сложности, т. е. декомпозиция программы на части (функциональные модули, сервисы, слои, подпрограммы) и организация их взаимодействия друг с другом и внешним миром (Рис. 3). Это дает возможность получить все те преимущества, которые обычно соотносятся с понятием хорошая архитектура: [9]

1. Масштабируемость (Scalability) — возможность расширять систему и увеличивать ее производительность, за счет добавления новых модулей;
2. Ремонтопригодность (Maintainability) — изменение одного модуля не требует изменения других модулей;
3. Заменимость модулей (Swappability) — модуль легко заменить на другой;
4. Возможность тестирования (Unit Testing) — модуль можно отсоединить от всех остальных и протестировать / починить;
5. Переиспользование (Reusability) — модуль может быть переиспользован в других программах и другом окружении;
6. Сопровождаемость (Maintenance) — разбитую на модули программу легче понимать и сопровождать.

Рис. 3. Декомпозиция программы на части

В рамках задачи было принято решение разработать два модуля:

1. Модуль передачи файлов внутри филиала;
2. Модуль синхронизации файлов с центром управления.

Реализация модулей будет осуществляться на языке C# и платформе .NET. Такое решение было принято по нескольким причинам, основная из которых связана с тем, что существующие модули системы СТКПлюс уже реализованы на языке C# и платформе .NET.

Модуль передачи файлов внутри филиала

Как было выявлено в ходе обзора инструментов синхронизации файлов, в рамках филиала есть два варианта организации передачи данных:

1. На каждом компьютере тестируемого установлен модуль передачи данных, который закачивает аудио-ответы на сервер филиала;
2. Модуль передачи данных установлен лишь на сервере филиала и забирает данные с компьютеров для тестирования.

Был выбран второй вариант. Основная причина выбора — при такой реализации сервер всегда сам будет контролировать очередность загрузки. Благодаря этому, какой бы протокол не использовался, ограничение на количество подключений не играет роли.

Передача файлов в модуле осуществляется через открытие общего доступа к файлам с помощью “общих папок” Windows на компьютере тестируемых. Это решение не требует установки дополнительного ПО на компьютере для тестируемого [10].

Запрос о необходимости передачи файла исходит из модуля тестирования, сразу после его записи на компьютере тестируемого. Запрос включает в себя следующую информацию:

1. Адрес компьютера в сети, с которого необходимо произвести передачу файла;
2. Имя файла и путь до него.
3. Приоритет файла — целое число: 1 или 0.

1 - используется для ответов, где подразумевался устный ответ.

0 - используется для всех остальных ответов.

Файлы с приоритетом 1 модуль передачи файлов загружает раньше, чем файлы с приоритетом 0 при наличии нескольких файлов с разными приоритетами в очереди. Приоритет 1 используется для ответов, где подразумевается устный ответ. Приоритет 0 используется для всех остальных ответов.

Также запрос о необходимости передачи файла может исходить и из модуля управления. Если при попытке воспроизведения аудио-ответа модуль управления не находит соответствующий файл на сервере филиала, он отправляет запрос модулю передачи файлов для его получения.

При успешной передачи файла или при возникновении ошибки, модуль передачи файлов отдает ответ о статусе передачи.

Для реализации запросов применена библиотека SignalR от Microsoft. SignalR построена по клиент-серверной технологии и предоставляет простой API для создания функционала, который позволяет вызывать методы на стороне клиента из серверного кода и наоборот. В качестве протокола обмена

сообщениями в SignalR используется WebSocket [11]. В рассматриваемом случае сервером SignalR служит модуль передачи файлов, а клиентом — модуль тестирования. Для отправки запросов с модуля тестирования, он был модифицирован.

Еще одной важной особенностью модуля передачи файлов является поддержка многопоточной передачи файлов. В настройках модуля можно указать сколько файлов одновременно можно копировать. Это особенность является важной при копировании файлов с нескольких компьютеров.

Всю информацию о передачи файлов модуль добавляет в лог-файлы.

Сам модуль оформлен в виде службы, для работы независимо от статуса пользователя. Настройки модуля хранятся в реестре. Можно указать следующие настройки:

1. Директория для хранения лог-файлов;
2. Адрес сервера SignalR;
3. Путь для сохранения аудио-ответов;
4. Тайм-аут передачи одного файла;
5. Число потоков для передачи.

Благодаря разработанной архитектуре в случае отказа работы модуля передачи файлов внутри филиала, тестирование не прервется. Файлы будут и дальше сохраняться на компьютере тестируемого, но не будут переданы на сервер филиала. Когда при проверке результатов файл не будет найден на сервере филиала, модулю передачи файлов от модуля управления будет отправлен запрос на его передачу.

Как видно из описанной архитектуры, недостатки существующего решения передачи файлов в предлагаемом автором решении отсутствуют.

Модуль синхронизации файлов с центром управления

Модуль синхронизации файлов с центром управления также установлен на сервере филиала. Он закачивает файлы на удаленный сервер центра управления. Передача файлов проходит также через открытие общего доступа к файлам с помощью “общих папок”. Само открытие общей папки производится уже в центре управления. Необходимо учесть, что сервер филиала и сервер центра управления находятся не в одной сети, поэтому перед передачей файлов модуль подключает VPN соединение между ними.

Связь между сервером филиала и центром управления не постоянна, поэтому модуль синхронизирует данные, когда она есть. Попытки синхронизации проводятся по расписанию.

Есть два пути реализации возможности запуска сеанса синхронизации по расписанию и независимо от статуса пользователя:

1. Оформить приложение в виде службы ОС и использовать планировщик внутри приложения;
2. Добавить задание по запуску приложения в планировщик заданий Windows.

Автором был выбран первый вариант.

Для .NET есть несколько библиотек, которые умеют синхронизировать две папки с файлами. Одной из них является RoboSharp - “обертка” над утилитой robocopy [12]. Она разработана для отказоустойчивого копирования каталогов и деревьев каталогов. Она обладает возможностью копирования всех (или выборочных) NTFS атрибутов и свойств, имеет дополнительный код для перезапуска при применении с сетевым соединением в случае его разрывов. Robocopy доступен как стандартный компонент в ОС семейства Windows. RoboSharp поддерживает все опции по работе с Robocopy. Однако у него есть существенный недостаток - привязанность к языку интерфейса операционной системы. Дело в том, что вывод утилиты выводится на том

языке, которой стоит в настройках ОС. Ввиду этого, многие возможности библиотеки не работают, в частности — отчет о скопированных файлах.

Более оправданным при разработке модуля оказалось использование библиотеки Microsoft Sync Framework. Она состоит из четырех основных компонентов: исполняющей среды, сервисов метаданных, провайдеров и участников синхронизации. [13]

Исполняющая среда Sync Framework предоставляет инфраструктуру для синхронизации данных между двумя источниками. Также поставляется SDK, который разработчики могут расширять для реализации собственных провайдеров.

Сервисы метаданных предоставляют инфраструктуру для хранения метаданных синхронизации, которые содержат информацию, используемую в сеансе синхронизации.

Метаданные синхронизации включают сведения о версиях, обнаружении изменений и др. Эти метаданные можно также использовать при проектировании разработке собственных провайдеров.

Провайдеры синхронизации применяются для синхронизации данных между репликами или конечными точками.

Реплика — это единица синхронизации; она используется для обозначения реального хранилища данных.

Участником (participant) называют источник, откуда можно извлечь данные, подлежащие синхронизации.

Провайдер синхронизации — компонент, который участвует в процессе синхронизации и обеспечивает синхронизацию данных одной реплики с данными других реплик. Должен быть один провайдер на каждую реплику.

Для синхронизации данных запускается сеанс синхронизации. В этом сеансе приложение подключается к провайдерам синхронизации для источника и получателя для передачи данных между репликами.

Рис. 4. Архитектура MS Sync Framework

По умолчанию в MS Sync Framework уже есть провайдер для синхронизации файлов — FileSyncProvider. Архитектура MS Sync Framework схематично представлена на Рис. 4 [14].

Свои настройки модуль синхронизации хранит в реестре. Можно указать следующие настройки:

1. Директория для хранения лог-файлов;
2. Адрес сервера SignalR;
3. Путь, где хранятся аудио-ответы на филиале;
4. Путь, где хранятся аудио-ответы в центре
5. Тайм-аут одного сеанса синхронизации;

Более подробно с реализацией модулей можно ознакомиться в приложении 2.

2.4. Обновленный сценарий развертывания системы

С учетом нового решения по управлению файлами, развертывание нового филиала и настройка компьютеров для тестирования должна проводиться следующим образом:

1. Настроить сервер базы данных на сервере филиала;

2. Установить и настроить модифицированный модуль управления;

   - В конфигурационном файле DataEditor.config необходимо указать строку для подключения к базе;

   - В конфигурационном файле App.Config необходимо указать адрес сервера SignalR модуля передачи файлов;

3. Установить модуль передачи файлов внутри филиала. В реестре в разделе настроек модуля казать адрес сервера SignalR и местоположение, куда необходимо сохранять аудио-ответы с компьютеров для тестирования;

4. На каждом компьютере для тестирования настроить общую папку в сети и доступ к ней;

5. Установить модуль тестирования на каждый компьютер для тестирования. В конфигурационном файле App.Config указать имя компьютера в сети, локальную папку для сохранения и адрес сервера SignalR модуля передачи файлов;

6. Создать подключение VPN на сервере филиала для связи с центром средствами Windows;

7. На сервер филиала установить модуль синхронизации файлов с центром управления. В реестре в разделе настроек модуля указать параметры подключения VPN, указать местоположение аудио-ответов на сервере филиала и адрес общей папки в сети центра.

После этого система готова к работе. Данный сценарий является более трудоемким, чем тот, что применяется в системе СТКПлюс, однако все вышеперечисленные шаги можно автоматизировать средствами управления конфигурациями, например, Puppet.

2.5. Нагрузочное тестирование

Автором было проведено нагрузочное тестирование **модуля передачи файлов внутри филиала**. Целью нагрузочного тестирования модуля является выявление следующих показателей:

1. Средний интервал времени между добавлением файла в очередь и окончанием его загрузки;
2. Средний интервал времени между началом загрузки файла и окончанием.

Тестирование модуля проводилось в одном из существующих филиалов системы тестирования СПбГУ. Сценарий тестирования был следующий:

1. Система СТКПлюс была развернута в соответствии с обновленным сценарием на 5 компьютеров в качестве компьютеров для тестирования и на сервер филиала.
2. На компьютеры тестируемых в общую папку было загружено по 3 файла объемом 1,5 Мбайт (средний объем аудиофайла в действующих филиалах);
3. Далее эти файлы были одновременно добавлены в очередь (сначала первый файл с первого компьютера, затем первый файл со второго и так до третьего файла с пятого компьютера).

В таблице 1 приведены результаты тестирования. Для большей наглядности результаты тестирования также приводятся в виде графика (Рис. 5.).

Количество потоков	1	2	3	4	5	6	7
Средний интервал времени между добавлением файла в очередь и окончанием его загрузки (мс)	841	750	683	570	565	600	622
Средний интервал времени между началом загрузки файла и окончанием (мс)	49	100	130	155	202	311	374

Табл. 1. Результаты нагрузочного тестирования модуля передачи файлов внутри филиала

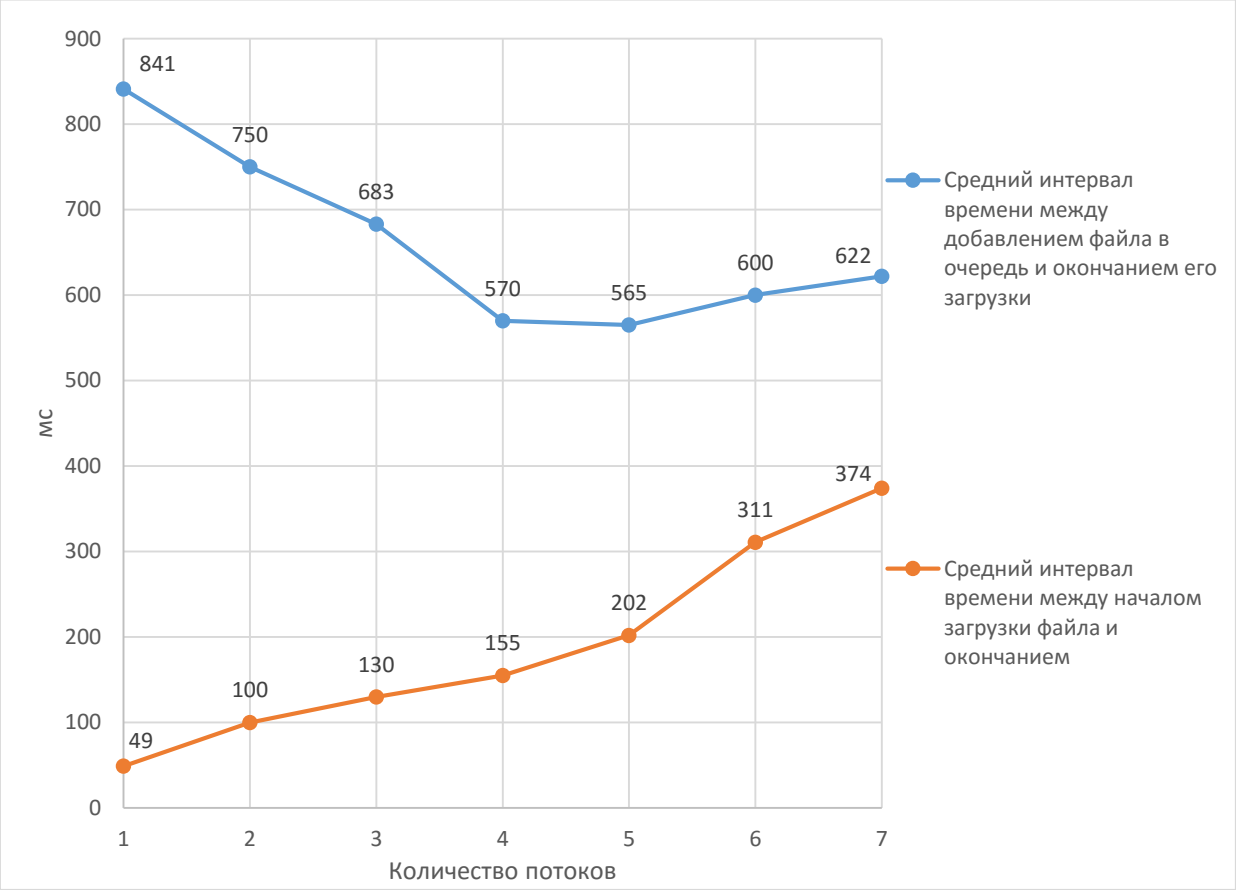

Рис. 5. Результаты нагрузочного тестирования модуля передачи файлов внутри филиала

Как видно из полученных результатов, минимальный средний интервал времени между добавлением файла в очередь и окончанием его загрузки достигается в случае использования 5 потоков копирования. Для выбора оптимального числа потоков для любого филиала требуется более детальное тестирование. Максимальный средний интервал времени загрузки файла оказался 374 мс.. Из этого можно сделать вывод, что в настройках модуля передачи файла необходимо установить тайм-аут не меньше этого значения. Для более точного выбора также требуется более детальное тестирование и учет максимальных по объему файлов.

2.6. Стресс-тестирование

Стресс-тестирование — один из видов тестирования программного обеспечения, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования. Автором были протестированы оба модуля.

В ходе стресс-тестирования была оценена модулей при возникновении следующих ситуаций:

1. Файл, подлежащий копированию, заблокирован;
2. Файл, подлежащий копированию, был удален;
3. Отсутствие связи при передачи файлов;
4. Превышение лимита одновременных подключений к общей сетевой папке компьютера-получателя (только для модуля синхронизации файлов с центром управления).

Для блокировки файлов была написана специальная утилита.

Модуль передачи файлов внутри филиала во всех трех случаях показал свою возможность обрабатывать исключительную ситуацию и добавлять в лог данные об ошибке. Наряду с этим модуль отдает клиентам

Signal R информацию об ошибке. Далее модуль продолжает работу и способен принимать новые запросы на передачу.

Модуль синхронизации файлов с центром управления во всех случаях также показал свою возможность обрабатывать исключительную ситуацию и добавлять в лог данные об ошибке. После этого модуль продолжает работу. Следует отметить, что сервис метаданных помечает файл, при копировании которого произошла ошибка как не отправленный. Поэтому при следующем сеансе синхронизации будет попытка повторной отправки файла.

2.7. Достигнутые результаты

Итогом работы, описанной в данной главе, являются следующие результаты:

1. Изучено существующее решение по управлению файлами в системе СТКПлюс;
2. Проведен обзор существующих инструментов по синхронизации файлов;
3. Разработан прототип архитектуры решения по управлению файлами;
4. Разработан прототип модуля передачи файлов внутри филиала;
5. Разработан прототип модуля синхронизации файлов с центром управления;
6. Проведено тестирование разработанных модулей.

Глава 3. Синхронизация баз данных

3.1. Исследование вопроса

Для работы системы СТКПлюс на сервере филиала должна быть установлена СУБД Microsoft SQL Server 2014 Express. Настройка СУБД проводится путем запуска начального скрипта. Данный скрипт создает базу данных и заполняет ее необходимыми данными. Эту базу использует, как модуль управления, так и модуль тестирования.

Предполагается, что в центре управления также установлен Microsoft SQL Server и развернута база данных системы. База данных на сервере центра управления, объединяет данные с серверов филиалов (Рис. 5). Однако в данный момент в системе СТКПлюс такой функционал не реализован. Автору работы было предложено ознакомиться с инструментами для синхронизации баз данных и сделать обоснованный выбор инструмента для синхронизации данных между филиалом и центром управления.

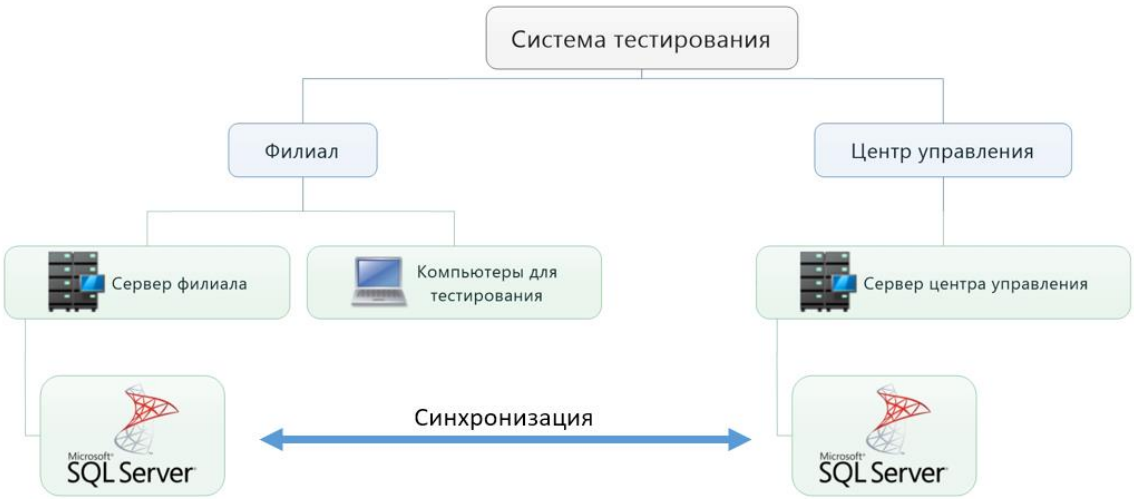

Рис. 6. Синхронизация баз данных между филиалом и центром управления

3.2. Выбор решения для синхронизации баз данных

В системе СТКПлюс можно выделить несколько основных требований к решению для синхронизации баз данных:

1. Двунаправленная синхронизация. Изменения на сервере филиала должны быть отправлены на сервер центра управления и наоборот;
2. Синхронизация более двух баз;
3. Работа в условиях отсутствия постоянной связи с центром управления;
4. Возможность передавать изменения в схеме базы;
5. Работоспособность системы в процессе синхронизации;

Автором работы были рассмотрены следующие инструменты:

1. Репликация SQL Server;
2. Доставка журналов транзакций;
3. Microsoft Sync Framework.

Репликация SQL Server

Встроенное в SQL Server средство для синхронизации баз данных. Репликация представляет собой набор технологий копирования и распространения данных и объектов баз данных между базами данных, а также синхронизации баз данных для поддержания согласованности. Используя репликацию, можно распространять данные в различные расположения, а также удаленным или мобильным пользователям по локальным или глобальным сетям посредством коммутируемого соединения, по беспроводным соединениям и через Интернет [15].

Microsoft SQL Server предоставляет три основных типа репликации:

1. Репликация моментальных снимков. При репликации моментальных снимков сервер передает подписчику целиком набор данных в указанное время, перезаписывая данные подписчика при

каждой операции копирования. Однако репликация моментальных снимков не затрагивает всю базу данных, как резервное копирование и восстановление, копируются только определенные статьи от издателя к подписчикам.

2. Репликация транзакций. В этом случае создается исходная полная копия данных, а все последующие копии передают только измененные данные. Репликация транзакций использует журнал транзакций, чтобы выполнить над конечными данными те же транзакции, что и над исходными.

3. Репликация слиянием. Репликация слиянием начинается с полного копирования данных от издателя к подписчикам. При последующем изменении данных на любом сервере процесс репликации производит эти изменения, разрешает конфликты, которые могли произойти из-за изменений, и переносит изменения на все серверы

Ввиду того, что из трех типов репликации требованию двунаправленной синхронизации соответствует только последний, дальше будет рассматриваться только репликация слиянием.

Доставка журналов транзакций

Доставка журналов транзакций - еще одно встроенное в SQL Server средство для синхронизации баз. Принцип работы следующий: на сервере-источнике регулярно производится резервное копирования журналов транзакций. Затем эти копии журналов автоматически передаются на получателей, на которых существует копия базы источника, и автоматически восстанавливаются [15].

Microsoft Sync Framework

В отличие от предыдущих вариантов, Sync Framework является не готовым решением, а библиотекой, предоставляющей API-интерфейс, который позволит создать приложение для синхронизации баз. [14].

Итоги сравнения инструментов в соответствии с установленными требованиями приведены в таблице 2. Символом «+» отмечено соответствие требованию, символом «-» обратное.

	Репликация слиянием SQL Server	Доставка журналов транзакций	Microsoft Sync Framework
Двунаправленная синхронизация.	+	-	+
Синхронизация более двух баз	+	+	+
Работа в условиях отсутствия постоянной связи с центром управлени	+	+	+
Возможность передавать изменения в схеме базы	+	+	-
Работоспособность системы в процессе синхронизации	+	-	+

Табл. 2. Итоги сравнения инструментов синхронизации баз данных

Анализ решений позволил сделать следующие выводы:

1. Из всех рассмотренных вариантов, всем требованиям удовлетворяет лишь репликация слиянием;
2. Доставка журналов транзакция больше подходит для резервирования базы данных;
3. Репликация слиянием и доставка журналов транзакций являются готовыми решениями, а Sync Framework требует написание приложения.

В итоге было принято решение синхронизировать базы с применением репликации слиянием.

3.3. Применение репликации

Для именования компонентов архитектуры репликации SQL Server используется терминология издательской отрасли.

- Издатель(publisher) — хранит главную копию базы данных, на которой настроена публикация;
- Подписчик (subscriber) — база данных, которая получает реплицируемые данные.
- Распространитель(distribution) — экземпляр MS SQL server настроенный для сбора транзакция с публикаций и их распространения на подписчики. Распространитель так же имеет базу данных для хранения реплицируемых транзакций;
- Статья (*article*) — это данные из какого-либо объекта на издателе. Подписаться на статью нельзя. Но статьи можно поместить в публикацию и настроить на нее подписку;
- Публикация (*publications*) — это основная единица репликации. Подписчики подписываются именно на публикацию. Публикация состоит из одной или нескольких статей, которые для целей публикации рассматриваются как единое целое. Подписаться на часть публикации нельзя
- Подписка (*subscription*) — это запрос на получение публикации. Подписка определяет, какую публикацию запрашивает подписчик и с какой частотой будут передаваться данные. Подписки могут быть двух видов: *принудительная* (*push*), когда передачу информации инициирует издатель, и *запрашивающая* (*pull*), когда в качестве инициатора выступает подписчик.

Предлагается рассмотреть применение репликации в процессе развертывания базы данных нового центра управления и нового филиала. Предполагается, что на сервере центра управления установлен SQL Server

2014 Standart. Сервер центра управления будет издателем, т.е. хранить главную копию базы, а сервер филиала подписчиком.

Анализ объектов базы данных СТКПлюс

Настройка СУБД внутри филиала в системе СТКПлюс проводится путем запуска начального скрипта. Данный скрипт создает базу данных и заполняет ее необходимыми данными. Прежде, чем приступать к применению репликации, необходимо разобраться какие именно объекты создает этот скрипт.

Важно отметить, что система СТКПлюс написана с использованием фреймворка eXpressApp Framework (XAF) от компании DevExpress. Это мощный инструмент для разработки бизнес-приложений. Благодаря eXpressApp Framework разработчик концентрируется на бизнес функциональности приложения, в то время как такие элементы приложения как права доступа, дизайнер отчетов, стандартный настраиваемый графический интерфейс, поддержка Web интерфейса и другие уже реализованы в XAF. Для доступа к объектам базы данных в XAF реализована прослойка между базой и “кодом программы” - ORM библиотека eXpress Persistent Objects (XPO). Помимо базового функционала ORM библиотек XPO предоставляет такие возможности, как:

- “оптимистичный параллелизм” [16];
- иерархия между объектами базы [17];
- отложенное удаление объектов [18].

Для реализации этих и других возможностей XPO требуется следующее “расширение“ таблиц, соответствующих бизнес-сущностям системы [19]:

1. Добавлен столбец *GCRecord* в каждую таблицу для отложенного удаления;
2. Добавлен столбец *OptimisticLockField* в каждую таблицу для оптимистичного параллелизма;

3. Создана таблица *XPObjectType* для реализации иерархии между объектами базы;

4. В каждую таблицу добавлен столбец глобального уникального идентификатора (GUID) для идентификации каждой строки.

Для аутентификации в модуле управления использован модуль XAF - Security Module [20]. Данный модуль использует следующие таблицы:

- *SecuritySystemRole*. Содержит список ролей пользователей;
- *SecuritySystemTypePermissionsObject*. Разрешения ролям на доступ к моделям;
- *SecuritySystemMemberPermissionsObject*. Разрешения на доступ к отдельным полям модели;
- *SecuritySystemUser*. Список пользователей;
- SecuritySystemUserUsers_SecuritySystemRoleRoles. Назначение пользователям ролей;
- *SecuritySystemObjectPermissionsObject*. Критерии для разрешений SecuritySystemTypePermissionsObject;
- SecuritySystemRoleParentRoles_SecuritySystemRoleChildRoles. Для вложения ролей в роли.

Пользовательские настройки отображения бизнес-сущностей содержатся в следующих таблицах [21]:

- *ModelDifference;*
- *ModelDifferenceAspect*.

Таблица *ModuleInfo* содержит ожидаемые версии модулей XAF.

Таблицы *IDGeneratorTable* и *ServerPrefix* используются в XAF классом *DistributedIdGeneratorHelper*, который предназначен для генерации “дружелюбных“ идентификаторов [22].

Все остальные таблицы соответствуют бизнес-сущностям системы.

Помимо перечисленных выше объектов, начальный скрипт создает следующие объекты базы:

1. Учетная запись SQL Server – SqlExaminerUser;
2. Пользователь в базу данных – SqlExaminerUser;
3. Роль базы данных — examinee;
4. Добавление SqlExaminerUser в роль examinee;
5. Ряд хранимых процедур, функций и представлений.

Выбор объектов базы для репликации

Поскольку центр управления должен хранить данные всех филиалов, начальный скрипт необходимо применить к нему.

Далее необходимо выбрать какие объекты, создаваемые скриптом, необходимо добавить в репликацию. Для того, чтобы в дальнейшем через репликацию можно было передавать изменения в структуре в базе, предлагается добавить все поддерживаемые репликацией объекты:

1. Таблицы;
2. Хранимые процедуры;
3. Представления;
4. Пользовательские функции.

Следует отметить, что все таблицы, участвующие в репликации слиянием, должны содержать столбец глобального уникального идентификатора (GUID) для идентификации каждой строки. В таблицах, создаваемых скриптом, такой столбец присутствует у всех таблиц, за исключением следующих:

1. *ModuleInfo;*
2. *ServerPrefix;*
3. *XPObjectType.*

Также все таблицы должны иметь более одного столбца. Такому требованию не удовлетворяет таблица *CentralServer*. Предлагается добавить в нее столбец с произвольным типом и названием.

Отметим объекты и действия, которые создает начальный скрипт, и которые невозможно реплицировать:

1. Учетная запись SQL Server – SqlExaminerUser;
2. Пользователь в базу данных – SqlExaminerUser;
3. Роль базы данных — examinee;
4. Добавление SqlExaminerUser в роль examinee.

Настройка центра управления

Далее производится настройка репликации на сервере центра управления. Репликацию можно настроить как через графическую оболочку SQL Server Management Studio, так и через хранимые системные процедуры SQL Server [23].

В итоге автором был написан новый скрипт для развертывания базы на сервере центра управления на основе скрипта для развертывания базы на сервере филиала. Новый скрипт создает все те же данные, что и начальный скрипт для филиала, однако таблицы, создаваемые новым скриптом, уже содержат столбцы глобального уникального идентификатора, в таблице CentralServer добавлен столбец с произвольным типом и названием. Также данный скрипт настраивает сервер центра управления, как издателя для репликации слиянием и добавляет все объекты, которые можно реплицировать.

Настройка филиала

Как было указано, настройка СУБД внутри филиала в системе СТКПлюс проводится путем запуска начального скрипта. Данный скрипт создает базу данных и заполняет ее необходимыми данными.

Однако с использованием репликации данный шаг необходимо модифицировать, т. к. после добавления подписчика издателю, на него будут скопированы все данные, выбранные для репликации у издателя.

Также заметим, что репликацию можно сделать только в уже имеющуюся базу данных. Т. е. на подписчике база данных должна быть создана заранее.

В итоге для настройки нового филиала, автором был написан скрипт, который выполняет следующее:

1. Создает новую базу данных;
2. Создает объекты и действия, которые невозможно получить от издателя (перечислены в разделе «Выбор объектов базы для репликации»).

Добавление филиала, как подписчика производится на стороне издателя также путем запуска скрипта.

После создания подписчика, SQL Server создаст необходимое задание на SQL Server центра управления, которое можно запускать в нужное время или по расписанию для синхронизации данных.

3.4. Достигнутые результаты

Итогом работы, описанной в этой главе стало следующее:

1. Проанализированы существующие инструменты синхронизации баз;
2. Сделан обоснованный выбор в пользу репликации MS SQL Server;
3. Разработан начальный скрипт для развертывания базы в центре управления;
4. Разработан начальный скрипт для развертывания базы в филиале;
5. Разработан скрипт для добавления подписчика.

Полученные результаты могут быть в дальнейшем использованы при разработке конечного решения по синхронизации баз между филиалом и центром управления. В частности, при проектировании конечного решения, необходимо учесть, что филиал и центр управления находятся не в одной сети, и перед запуском сеанса синхронизации необходимо настроить VPN соединение.

Заключение

В рамках работы было изучено одно из самых актуальных средств по управлению конфигурациями — Puppet. Далее с помощью Puppet были автоматизированы развертывание и настройка филиала. Выводы, полученные после изучения, помогли лучше продумать архитектуру решения по управлению файлами.

Была изучена архитектура системы СТКПлюс. Просмотрены исходные коды, используемые библиотеки и т.д. В частности, был изучен существующий способ управления файлами в системе СТКПлюс и выявлены его недоставки.

Далее автор выполнил обзор по заданным критериями существующих инструментов для синхронизации файлов. Полученные выводы были использованы при построение собственного решения.

Был разработан прототип решения по управлению файлами в системе СТКПлюс. Полученное решение лишено недостатков существующего и обладает необходимым потенциалом гибкости. Решение включает в себя два модуля – модуль управления файлами внутри филиала и модуль синхронизации файлов с центром управления. Возможна независимая модификация отдельных модулей системы и распределенный запуск программного комплекса. Проведено тестирование разработанных модулей.

Далее был составлен обзор существующих инструментов по синхронизации баз данных. Сделан обоснованный выбор в пользу репликации MS SQL Server. Далее были проанализированы объекты базы данных системы и выбраны объекты для репликации. Разработаны скрипты для настройки центра управления и филиала. Полученные результаты могут быть в дальнейшем использованы при разработке конечного решения.

Список литературы и источников

1. Документация Puppet — https://docs.puppet.com/puppet/latest/reference/;
2. Условия лицензионного соглашения на использование программного обеспечения Microsoft Windows 7;
3. Документация Exiland Backup. https://exiland-backup.com/prod/backup-ru.pdf;
4. Страница покупки Exiland Backup Professional. https://exiland-backup.com/ru/backup-purchase.html
5. Документация SmartSyncPro. http://www.smartsync.com/downloads/manual.pdf
6. Документация SyncBackPro. http://www.2brightsparks.com/assets/pdf/SyncBackPro-PDF.pdf
7. Руководство пользователя GoodSync. http://www.goodsync.com/ru/manual
8. Документация GoodSync Connect. http://www.goodsync.com/goodsync-connect-manual
9. Создание архитектуры программы или как проектировать табуретку. https://habrahabr.ru/post/276593/
10. Открытие общего доступа к файлам с помощью общих папок. http://windows.microsoft.com/ru-ru/windows7/share-files-using-the-public-folders
11. Документация SignalR. https://github.com/SignalR/SignalR/wiki
12. Документация RoboSharp. https://robosharp.codeplex.com/documentation
13. Документация Microsoft Sync Framework. https://msdn.microsoft.com/ru-ru/library/bb902854%28v=sql.105%29.aspx
14. Rituraj Singh , Joydip Kanjilal. Pro Sync Framework, 2009
15. Thomas O. Training Kit Exam 70-462 Administering Microsoft SQL Server 2012 Databases, 2012
16. Документация DevExpress «Optimistic Concurrency». https://documentation.devexpress.com/#CoreLibraries/CustomDocument2028

17. Документация DevExpress «Inheritance Mapping». https://documentation.devexpress.com/#CoreLibraries/CustomDocument2019
18. Документация DevExpress «Deleting Persistent Objects». https://documentation.devexpress.com/#CoreLibraries/CustomDocument2026
19. Документация DevExpress «When and Why XPO Extends the Database Schema». https://documentation.devexpress.com/#CoreLibraries/CustomDocument2632
20. Сайт DevExpress, раздел «Rock-Solid Application Security». https://www.devexpress.com/Products/NET/Application_Framework/features_security.xml
21. Документация DevExpress «Model Difference Storages». https://documentation.devexpress.com/#eXpressAppFramework/CustomDocument113697
22. Сайт DevExpress, раздел «Support Center». https://www.devexpress.com/Support/Center/Example/Details/E4904
23. Основные понятия системных хранимых процедур репликации. https://msdn.microsoft.com/ru-ru/library/ms147302%28v=sql.120%29.aspx

Приложение 1. Код манифеста Puppet для развертывания филиала

```
node agent1{
	#options
	$shared_folder_user = "disk"
	$folder_to_share = "C:\\share"

	file{ 'C:\\from_puppetmaster':
	ensure = > 'directory',
	}

	file{ "C:\\from_puppetmaster\\test1":
	ensure = > "directory",
		source = > "puppet:///modules/shared_files/test1",
		recurse = > "true",
	}

	file{ "C:\\from_puppetmaster\\initial.sql":
	source = > "puppet:///modules/shared_files/initial.sql",
	}

	exec{ 'mssql':
	command = > 'C:\sql2014\SQLEXPRADV_x86_RUS\SETUP.EXE /Q /ACTION=Install
/FEATURES=SQL,SSMS /INSTANCENAME=MSSQLSERVER /SQLSVCACCOUNT=user /SQLSVCPASSWORD=YUP0Dh
/SQLSYSADMINACCOUNTS=user /AGTSVCACCOUNT="NT AUTHORITY\Network Service"
/IACCEPTSQLSERVERLICENSETERMS /TCPENABLED=1',
		creates = > 'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\Binn\sqlservr.exe',
	}

	exec{ 'open_port':
	command = > 'C:\Windows\System32\cmd.exe /c netsh firewall set portopening
protocol = TCP port = 1433 name = SQLPort mode = ENABLE scope = SUBNET profile =
CURRENT',
		subscribe = > Exec["mssql"],
		refreshonly = > true,
	}

	exec{ 'create_database':
	command = > 'C:\Windows\System32\cmd.exe /c sqlcmd -i
C:\\from_puppetmaster\\initial.sql',
		subscribe = > Exec["mssql"],
		refreshonly = > true,
	}

	exec{ 'add_user':
	command = > "C:\\Windows\\System32\\cmd.exe /c net user $shared_folder_user
$shared_folder_user /ADD",
		unless = > "C:\\Windows\\System32\\cmd.exe /c net user $shared_folder_user
>nul 2>&1 && (exit 0) || (exit 1)"
	}

	file{ 'create_folder_to_share':
	path = > $folder_to_share,
		ensure = > 'directory',
		owner = > $shared_folder_user,
		require = > Exec["add_user"],
	}

	exec{ 'share_folder':
	command = > "C:\\Windows\\System32\\cmd.exe /c net share share=$folder_to_share",
```

```
            unless = > "C:\Windows\System32\cmd.exe /c if exist \\\\${hostname}\\share
(exit 0) else (exit 1)",
            require = > File['create_folder_to_share'],
      }

      exec{ 'mount_folder':
      command = > "C:\Windows\System32\cmd.exe /c net use T: \\\\${hostname}\\share
/SAVECRED",
            unless = > "C:\Windows\System32\cmd.exe /c if exist  T: (exit 0) else (exit
1)",
            require = > Exec['share_folder'],
      }
      }
```

Приложение 2. Исходный код модулей по управлению файлами

Ниже частично приведены исходные коды некоторых классов разработанных модулей. Объем кода слишком велик и в рамках данной работы привести его полностью не представляется возможным.

Модуль управления файлами внутри филиала

```
public partial class ExaminatorBranchFileTransferService : ServiceBase
{
    private static readonly object locker = new object();

    List<Job> jobQueue = new List<Job>();

    //settings
    string logPath;
    string signalRServerUrl;
    string branchAnswersPath;
    int jobTimeOut;
    int maxRunningJobs;

    public ExaminatorBranchFileTransferService()
    {
        InitializeComponent();
    }

    protected override void OnStart(string[] args)
    {
        Init();
     }

    //------часть кода опущена-------

    private int runningJobsCount
    {
        get
        {
            return jobQueue.Count(job => job.State == JobState.Running);
        }
    }

    private int notRunningJobsCount
    {
        get {
            return jobQueue.Count - runningJobsCount;
        }
    }

    private void updateRunningJobs()
    {
        for (int i = 1; i > 0; i--)
        {
            foreach (Job job in jobQueue.AsEnumerable().Reverse())
            {
                if (runningJobsCount >= maxRunningJobs) break;
                if ((job.State == JobState.New) && (job.Priority == i))
                {
```

```
                        runJob(job);
                    }
                }
            }
        }

        public void StartServer()
        {
            string url = signalRServerUrl;
            WebApp.Start<Startup>(url);
            GlobalHost.DependencyResolver.Register(typeof(FileSyncHub), () => new
FileSyncHub(this));
        }

        public void AddJobQueue(string sourceDirectory, string fileName, int priority)
        {
            jobQueue.Add(new Job {
                SourceDirectory = sourceDirectory,
                DestinationDirectory = branchAnswersPath,
                FileName = fileName,
                Priority = priority,
                State = JobState.New,
                AddedDateTime = DateTime.Now,
            });
            updateRunningJobs();
        }

        private async void runJob(Job job)
        {
            job.State = JobState.Running;
            Stopwatch jobStopWatch = new Stopwatch();
            jobStopWatch.Start();
            var task = Task.Run(() =>
            {
                try
                {
                    copyFile(job);
                    job.State = JobState.Completed;
                }
                catch (Exception e)
                {
                    job.State = JobState.Error;
                    job.ErrorMessage = e.Message;
                }
            });
            if (await Task.WhenAny(task, Task.Delay(jobTimeOut)) != task)
            {
                //if time out
                try
                {
                    string destFile = Path.Combine(job.DestinationDirectory,
job.FileName);
                    System.IO.File.Delete(destFile);
                }
                catch { }
                job.State = JobState.Error;
                job.ErrorMessage = "Time Out";
            }
            job.Elapsed = jobStopWatch.Elapsed;
            job.CompletedDateTime = DateTime.Now;
            string logMessage = Log(job);
            if (Debugger.IsAttached)
            {
                Console.WriteLine(logMessage);
```

```
            }
            jobQueue.Remove(job);
            updateRunningJobs();
        }

        private void copyFile(Job job)
        {
            string sourceFile = Path.Combine(job.SourceDirectory, job.FileName);
            string destFile = Path.Combine(job.DestinationDirectory, job.FileName);
            Directory.CreateDirectory(Path.GetDirectoryName(destFile));
            File.Copy(sourceFile, destFile, true);
         }

        //------часть кода опущена-------
       }
```

Модуль синхронизации файлов с центром управления

```
    public partial class ExaminatorCenterFileSyncService : ServiceBase
    {
        Scheduler centerScheduler;
        Logger logger;

        public ExaminatorCenterFileSyncService()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            Init();
        }

        //------часть кода опущена-------

       private void uploadFilesToCenter()
        {
            try
            {
                if (!Directory.Exists(branchAnswersPath))
                {
                    throw new Exception("BranchAnswersPath не существует");
                }

                if (!Directory.Exists(CenterAnswersPath))
                {
                    throw new Exception("CenterAnswersPath не существует");
                }
                FileSyncScopeFilter fileSyncScopeFilter = new FileSyncScopeFilter();

                string metadataDirectoryPath =
Path.Combine(Path.GetDirectoryName(System.Reflection.Assembly.GetEntryAssembly().Location
));
                string sourceMetaData = "source.metadata";
                string destinationMetaData = "destination.metadata";

                FileSyncProvider sourceProvider = new FileSyncProvider(branchAnswersPath,
fileSyncScopeFilter, FileSyncOptions.None, metadataDirectoryPath, sourceMetaData,
branchAnswersPath, null);
                FileSyncProvider destinationProvider = new
FileSyncProvider(centerAnswersPath, fileSyncScopeFilter, FileSyncOptions.None,
metadataDirectoryPath, destinationMetaData, centerAnswersPath, null);
```

```
                SyncOrchestrator syncOrchestrator = new SyncOrchestrator();
                syncOrchestrator.LocalProvider = sourceProvider;
                syncOrchestrator.RemoteProvider = destinationProvider;
                syncOrchestrator.Direction = SyncDirectionOrder.Upload;

                destinationProvider.AppliedChange += delegate(object sender,
AppliedChangeEventArgs e)
                {
                    string message =
                        e.ChangeType + " " +
                        Path.Combine(centerAnswersPath, e.NewFilePath);
                    Log(message);
                };

                destinationProvider.SkippedChange += delegate(object sender,
SkippedChangeEventArgs e)
                {
                    string message =
                        e.ChangeType + " skipped because " + e.SkipReason + " in " +
                        Path.Combine(centerAnswersPath, e.NewFilePath);
                    Log(message);
                };

                try
                {
                    syncOrchestrator.Synchronize();
                }
                catch (Microsoft.Synchronization.SyncException se)
                {
                    throw new Exception(se.Message);
                }

            } catch (Exception e)
            {
                string message = "error " + e.Message;
                Log(message);
            }

        }
```

Приложение 3. Скрипт для развертывания базы в центре управления

Приведена лишь часть скрипта. В зависимости от остальных настроек экземпляра SQL Server в центре управления некоторые параметры используемых процедур могут быть изменены.

```
-- Включение базы данных репликации
use master
exec sp_replicationdboption @dbname = N'Examinator', @optname = N'merge publish', @value
= N'true'
GO

-- Добавление публикации слиянием
use [Examinator]
exec sp_addmergepublication @publication = N'asd', @description = N'Публикация слиянием
базы данных "Examinator" от издателя "CANTTI-PC".', @sync_mode = N'native', @retention =
14, @allow_push = N'true', @allow_pull = N'true', @allow_anonymous = N'true',
@enabled_for_internet = N'false', @snapshot_in_defaultfolder = N'true',
@compress_snapshot = N'false', @ftp_port = 21, @ftp_subdirectory = N'ftp', @ftp_login =
N'anonymous', @allow_subscription_copy = N'false', @add_to_active_directory = N'false',
@dynamic_filters = N'false', @conflict_retention = 14, @keep_partition_changes =
N'false', @allow_synctoalternate = N'false', @max_concurrent_merge = 0,
@max_concurrent_dynamic_snapshots = 0, @use_partition_groups = N'false',
@publication_compatibility_level = N'100RTM', @replicate_ddl = 1,
@allow_subscriber_initiated_snapshot = N'false', @allow_web_synchronization = N'false',
@allow_partition_realignment = N'true', @retention_period_unit = N'days',
@conflict_logging = N'both', @automatic_reinitialization_policy = 0
GO


exec sp_addpublication_snapshot @publication = N'asd', @frequency_type = 4,
@frequency_interval = 14, @frequency_relative_interval = 1, @frequency_recurrence_factor
= 0, @frequency_subday = 1, @frequency_subday_interval = 5, @active_start_time_of_day =
500, @active_end_time_of_day = 235959, @active_start_date = 0, @active_end_date = 0,
@job_login = N'cantti-pc\cantti', @job_password = null, @publisher_security_mode = 1
exec sp_grant_publication_access @publication = N'asd', @login = N'sa'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'cantti-pc\cantti'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'NT SERVICE\Winmgmt'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'NT SERVICE\SQLWriter'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'NT
SERVICE\SQLSERVERAGENT'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'NT
Service\MSSQLSERVER'
GO
exec sp_grant_publication_access @publication = N'asd', @login = N'distributor_admin'
GO

-- Добавление статей слияния
use [Examinator]
exec sp_addmergearticle @publication = N'asd', @article = N'AdministrativeTool',
@source_owner = N'dbo', @source_object = N'AdministrativeTool', @type = N'table',
@description = N'', @creation_script = N'', @pre_creation_cmd = N'drop', @schema_option =
0x000000010C034FD1, @identityrangemanagementoption = N'none', @destination_owner =
```

```
N'dbo', @force_reinit_subscription = 1, @column_tracking = N'false', @subset_filterclause
= N'', @vertical_partition = N'false', @verify_resolver_signature = 1,
@allow_interactive_resolver = N'false', @fast_multicol_updateproc = N'true',
@check_permissions = 0, @subscriber_upload_options = 0, @delete_tracking = N'true',
@compensate_for_errors = N'false', @stream_blob_columns = N'true', @partition_options = 0
```