Министерство образования и науки Российской Федерации
(МИНОБРНАУКИ РОССИИ)
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (ТГУ)
Факультет информатики
Кафедра теоретических основ информатики

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП
к.т.н., доцент
_____ А.Л.Фукс
«__8__» ___06___ 2016 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С ДОКУМЕНТАМИ В ПОТОКЕ РАБОТ ДЛЯ ПЛАТФОРМЫ VDOM DESKTOP APPLICATIONS

по основной образовательной программе подготовки бакалавров
направление подготовки 02.03.02 – Фундаментальная информатика и информационные технологии

Жуков Игорь Андреевич

Руководитель ВКР,
Старший преподаватель
_____ М.С. Овсянников
подпись
«__08__» ___июня___ 2016 г.

Автор работы
студент группы №__1421__
_____ И.А. Жуков
подпись

Томск – 2016

Реферат

Дипломная работа, 42 страницы, 25 рисунков, 15 источников, 2 приложения.

ЭЛЕКТРОННЫЙ ДОКУМЕНТООБОРОТ, QT, VDOM ТЕХНОЛОГИИ, API, PROSHARE.

Цель: разработка приложения для работы с документами в контексте потока работ.

Результаты работы: разработано приложение для работы с документами в потоке работ. Приложение принято заказчиком и внедрено.

Область применения: предприятия и госучреждения, нуждающиеся в системе электронного документооборота.

Содержание:

Введение

Важной составляющей бизнес процессов является движение и обработка документов. Задача автоматизации бизнес процессов с применением ЭВМ возникла в семидесятые годы XX века и актуальна по настоящее время. Для автоматизации движения и обработки документов были созданы системы электронного документооборота. Система электронного документооборота "представляет собой многопользовательскую клиент-серверную (или трехзвенную) автоматизированную систему, используемую в процессе управления организацией с целью обеспечения выполнения этой организацией своих функций" [1].

Современные программные продукты помимо электронного документооборота предоставляют более широкий набор функций таких как: планирование задач внутри предприятия, корпоративная электронная почта, списки контактов предприятия и многое другое. Такие системы называются системами управления корпоративным информационным контентом (англ. Enterprise content management) "ECM — это управление документами и другими типами контента, а также их хранение, обработка и доставка в масштабах предприятия" [2]. Одной из таких систем является разработка компании VDOM Box Group комплект программ ProSuite.

Для хранения документов в ProSuite используется ProShare. Осуществлять электронный документооборот при помощи этого приложения возможно, но вследствие того, что это не является основной задачей ProShare, для пользователя эта работа неудобна. Основные трудности для пользователя:

- Web-приложение может одновременно отображать только один каталог.
- Просмотреть документ возможно только загрузив его на локальный компьютер.
- Без расширений, для перемещения файла из одной папки в другую необходимо удалить файл в изначальной папке и повторно загрузить документ в систему в другую папку.

Целью данной работы является разработка приложения для работы с документами в контексте потока работ. Поток работ (англ. Workflow) – "это полная или частичная автоматизация бизнес-процесса, при которой документы, информация или задания передаются от одного участника бизнес-процесса к другому для выполнения действий согласно набору руководящих правил" [3].

Для достижения указанной цели необходимо решить следующие задачи:

- Обзор существующих систем электронного документооборота.
- Реализация программного продукта согласно требованиям заказчика.
- Тестирование разработанного программного продукта.

1. Обзор существующих систем электронного документооборота

Рассмотрим распространённые существующие решения. При рассмотрении будет обращать на основной функционал и платформы, для которых доступны клиенты.

1.1 Microsoft SharePoint Online

SharePoint – разработка корпорации Microsoft. Первая версия была выпущена в 2001 году. Предоставляет возможности управление документами, электронной почтой и контактами, создания сайтов группы (англ. team site) и персональное облако. Данное приложение имеет клиенты под мобильные операционные системы: iOS, Android, Windows 10 mobile и web-приложение.

Microsoft SharePoint Online является частью продукта SharePoint, входящего в Microsoft Office. Следовательно, является проприетарным программным обеспечением, требующим ежемесячной подписки. При этом серверная часть SharePoint Server распространяется бесплатно для серверов под управлением Windows Server 2012 R2, что ограничивает применимость продукта.

Скриншот интерфейса приведён в [4] и на рисунке 1.

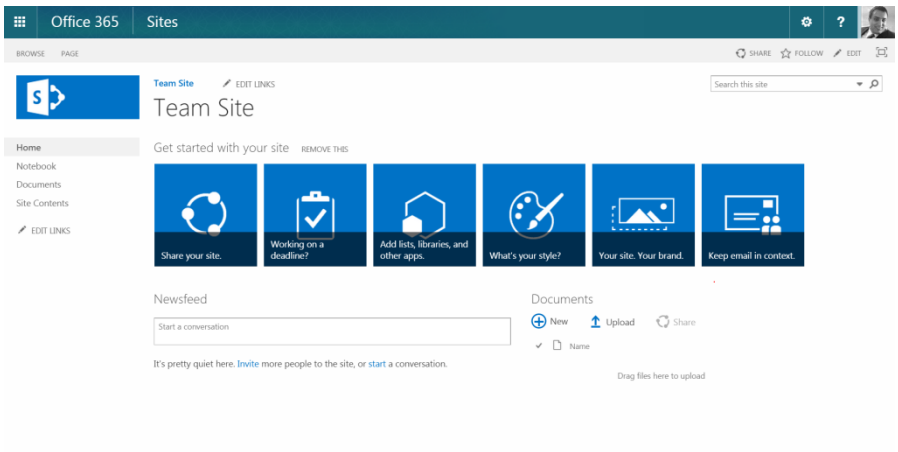

Рисунок 1 — Интерфейс приложения Microsoft Share Point Online

1.2 IBM Notes

IBM Notes – разработка корпорации IBM. Первая версия была выпущена в 1989 году. Предоставляет возможность работы с электронной почтой, календарями, списками дел, управление документами и контактами, а также мгновенную передачу сообщений. Существует клиент для операционных систем Windows, OS X, Linux и для мобильных платформ: Android, Blackberry, iOS, Windows 10 Mobile. Notes является проприетарным программным обеспечением.

Серверная часть для IBM Notes – IBM Domino использует NoSQL базу данных. Сервер может быть запущен как на Windows, так и на Linux серверах.

IBM Notes/Domino так же предоставляет фреймворк для разработки модулей.

Скриншот интерфейса приведён в [5] и на рисунке 2.

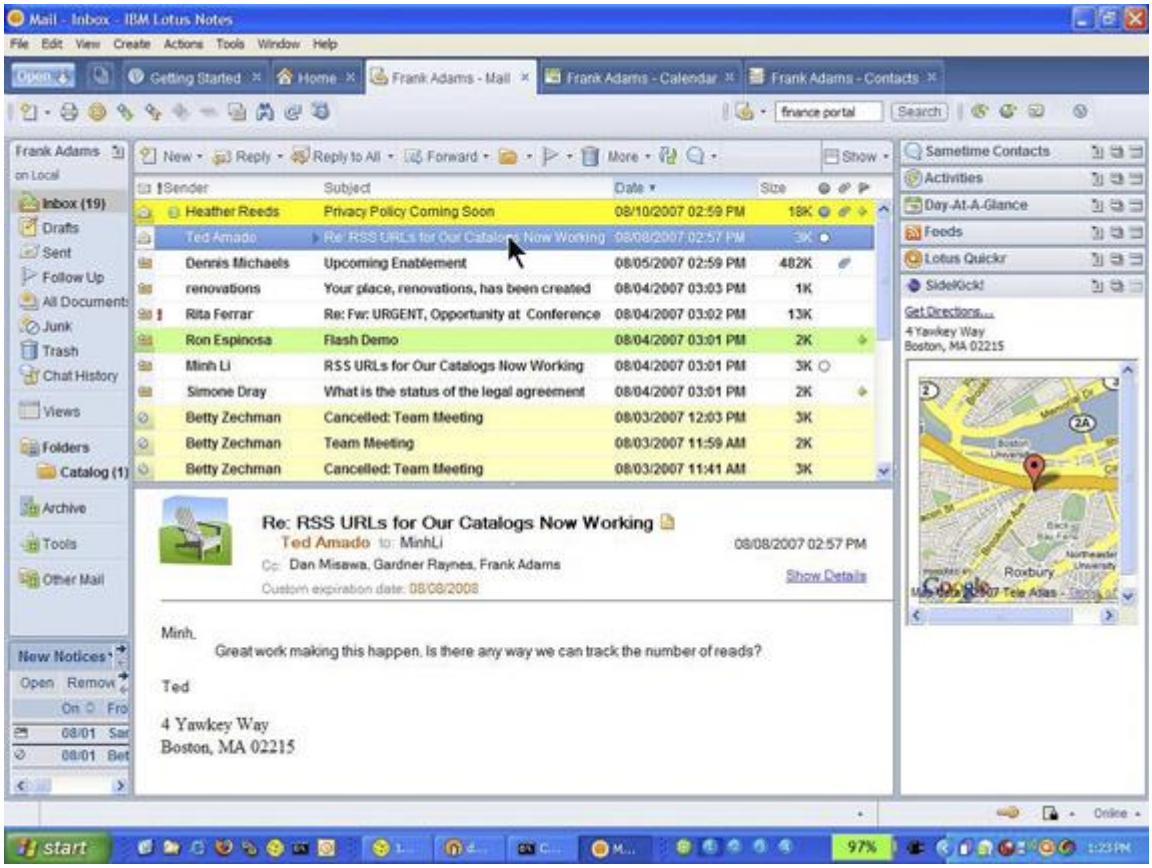

Рисунок 2 — Интерфейс приложения IBM Notes

1.3 Directum

Directum — ECM-система, разработанная российской компанией Directum. Система занимает примерно 25% рынка ECM –систем в России.

В состав системы входят модули:

- Управление электронными документами.
- Управление деловыми процессами.
- Управление договорами.
- Управление совещаниями.
- Канцелярия.
- Управление взаимодействием с клиентами.

Клиент для работы с Directum – web-приложение. Так же существует клиенты для мобильных операционных систем: iOS, Android, Windows 10 mobile.

Основным недостатком Directum является зависимость от платформы Windows.

Скриншот интерфейса приведён в [6] и на рисунке 3.

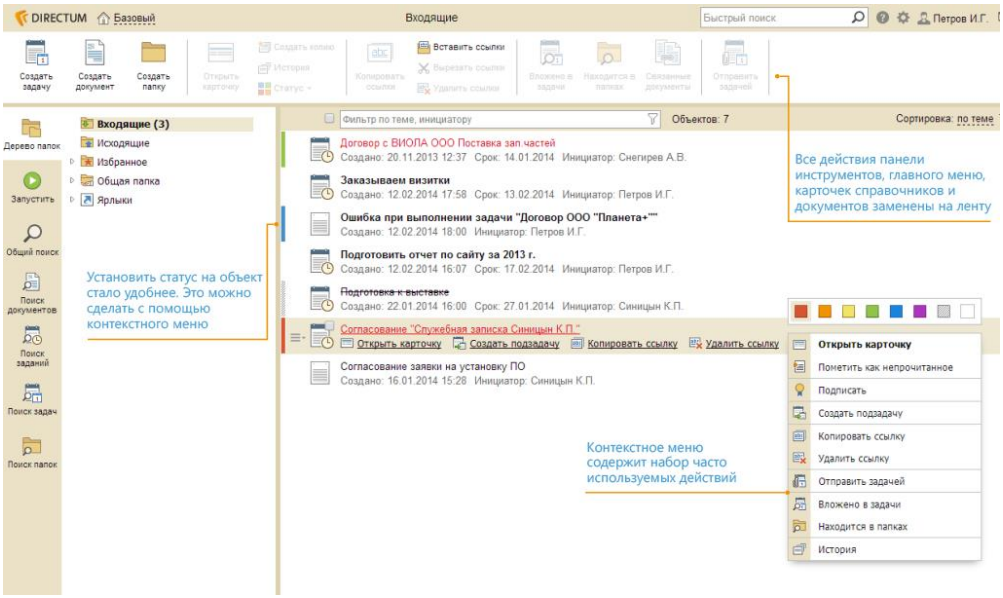

Рисунок 3 — Интерфейс приложения Directum

1.4 Eye on File Online

Eye on File Online – система электронного документооборота, созданная французской компанией BGSoft. Предоставляет возможность хранения документов, редактирования метаданных и полнотекстовый поиск по документам.

Система разработана на технологии VDOM. Используется Embedded SQLite в качестве базы данных. Клиентом является web-приложение.

Скриншот интерфейса приведён в [7] и на рисунке 4.

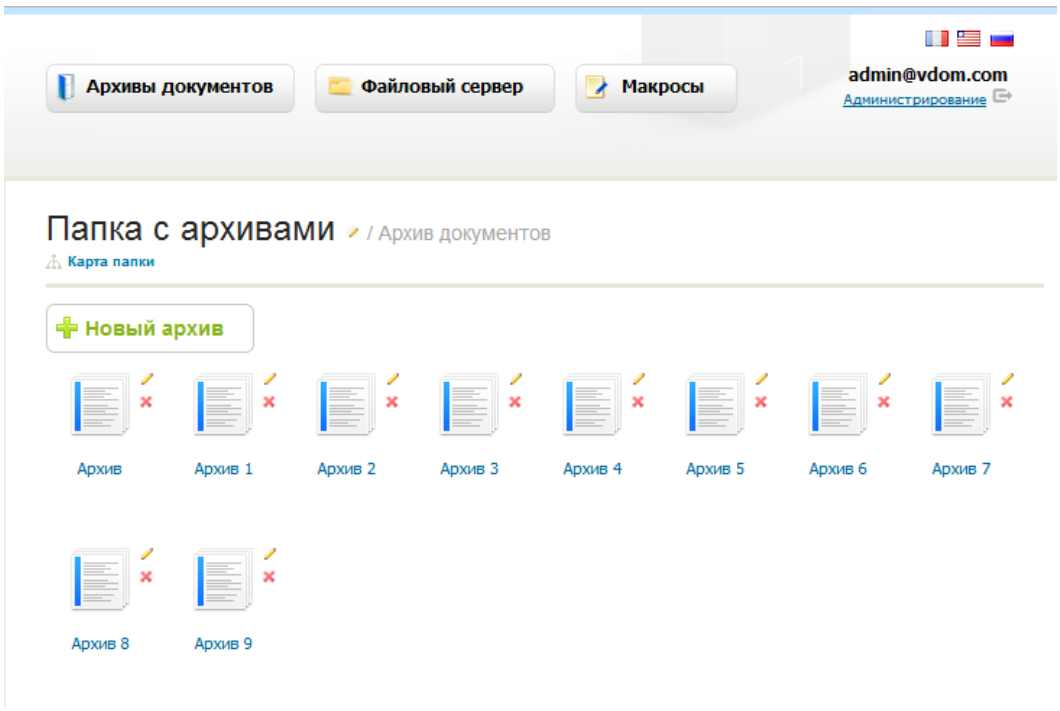

Рисунок 4 — Интерфейс приложения Eye on File Online

Кроме рассмотренных систем существуют и множество других. Системы Microsoft SharePoint Online и IBM Notes были рассмотрены исходя из их распространённости на мировом рынке. Также сыграла свою роль и то, что оба продукта разработаны гигантами IT-индустрии: Microsoft и IBM соответственно. Система Directum популярна на российском рынке. Eye on Files Online хоть и не является широко распространённой, но именно эта система использовалась в компании, заказавшей разработку.

2. Анализ требований и проектирование

2.1 Основные требования

Согласно требованием заказчика основной функционал приложения должен включать:

- Отображение pdf документов.
- Механизм обновления и синхронизации состояния хранилищ.
- Механизм блокирования документа.
- Генерация формы документа.
- Проверка значений компонентов формы.

Помимо этого также были определены и нефункциональные требования:

- Отображение логической структуры хранилищ.
- Пользовательский интерфейс с перемещаемыми виджетами (Dockable UI).

Также требованием заказчика была платформа разработки – VDOM Desktop Applications.

Исходя из списка требований, можно сделать вывод, что существующие решения, рассмотренные в главе 1, не подходят для решения поставленной задачи.

2.2 Платформа VDOM Desktop Applications

Платформа VDOM Desktop Applications предоставляет возможность запуска приложений на локальной машине. Для запуска приложений используется VDOM Application Launcher. Каждое приложение хранится на сервере в формате XML. При каждом запуске приложение скачивается с сервера. Для разработки приложений для платформы VDOM Desktop Applications используется интегрированная среда разработки VDOM Script IDE.

VDOM Application Launcher и VDOM Script IDE написаны на языке программирования C++ с использованием библиотеки Qt.

“Qt — кроссплатформенный инструментарий разработки ПО на языке программирования C++.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода. MOC позволяет во

много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код” [8].

Движком приложений является QtScript. QtScript основан на скриптовом языке ECMAScript стандарта ECMA-262. Все компоненты Qt также сигналы и слоты поддерживаются в QtScript.

2.3 ProShare

ProShare – онлайн система управления файлами. В ProShare существует система виртуальных хранилищ, которая позволяет изменять местонахождение документа в системе, при этом, не перемещая его физически. Существует три вида хранилищ: виртуальное хранилище (англ. virtual storage), контейнер (англ. container), хранилище документов (англ. document storage).

Структура хранилищ является иерархической. На вершине иерархии находится виртуальное хранилище. Каждое хранилище может содержать множество контейнеров. Каждый контейнер может содержать множество хранилищ документов. Хранилище документов может содержать множество документов. Один документ может принадлежать только одному хранилищу документов. Также в ProShare с каждым документом могут быть связаны некоторые данные.

В контексте функционала ProShare поток работ может быть представлен одним сценарием. Различные участники потока работ могут быть представлены хранилищами документов.

Пример web-интерфейса ProShare приведён на рисунке 5.

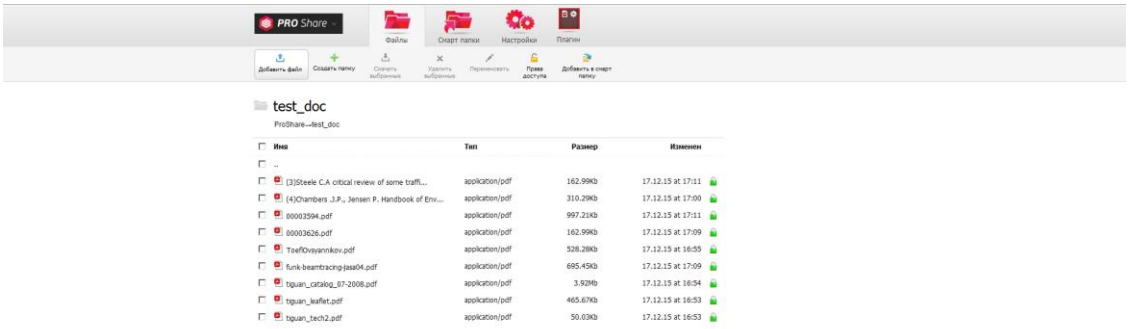

Рисунок 5 — Web-интерфейс приложения ProShare

2.4 Взаимодействие с сервером

Для взаимодействия с сервером ProShare используется ProShare API.

“API (интерфейс программирования приложений, интерфейс прикладного программирования) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах” [9].

Для обмена данными используется формат JSON.

“JSON (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999. JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными.

- Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
- Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме” [10].

Основные API методы, которые будут использоваться:

- **login** – get-запрос. Используется для авторизации пользователя.
- **list_storages** – get-запрос. Используется для получения списка виртуальных хранилищ.
- **get_storage** – get-запрос. Используется для получения содержания виртуального хранилища.
- **get_doc_storage** – get-запрос. Используется для получения списка документов в хранилище документов.
- **select_document** – get-запрос. Используется для получения формы документа.
- **submit_single_document** – post-запрос. Используется для осуществления действия с одним документом.
- **submit_multi_documents** – post-запрос. Используется для осуществления действия с несколькими документами.
- **lock_document** – post-запрос. Используется для блокирования документа.
- **unlock_document** – post-запрос. Используется для разблокирования документа.
- **logoff** – post-запрос. Используется для выхода пользователя.

2.5 Просмотр документа

Для осуществления корректной работы с документом пользователю необходимо иметь возможность просматривать содержимое этого документа. Требовалось дать пользователю возможность просматривать содержимое документа прямо из приложения. Следует отметить, что в данной задаче все документы имеют формат PDF. Дадим определение этому формату.

“PDF (Portable Document Format) — межплатформенный формат электронных документов, разработанный фирмой Adobe Systems с использованием ряда возможностей языка PostScript. В первую очередь предназначен для представления полиграфической продукции в электронном виде” [11].

Обычно для отображения PDF документов используется сторонние приложения. Так же такие документы возможно просматривать с помощью web-браузеров. Для отображения документа в приложении существует две возможные реализации.

Первым вариантом является использование какой-либо сторонней библиотеке для рендеренга PDF документа в файлы изображений. В таком случае требуется или разработка стороннего приложения, которое будет принимать PDF документ и возвращать файлы изображений со страницами документа, или встраивать модуль рендеренга pdf документов в VDOM Application Launcher.

Были рассмотрены популярные библиотеки для рендеринга PDF документов: poppler и muPDF.

Библиотека poppler написана на языке программирования C++ и является свободным программным обеспечением. Главным недостатком является слабая поддержка операционной системы Windows. То есть существует реализация библиотеки для этой платформы, но в ней отсутствует значительное количество дополнительного функционала. Преимуществом же в контексте решаемой задачи является наличие интерфейса для Qt. Следовательно, это упрощает интеграцию в VDOM Application launcher, так как нет необходимости задумываться о преобразовании формата представления изображения в приложении.

Библиотека muPDF написана на языке программирования C. Эта библиотека имеет как свободно распространяемую версию, так и коммерческую. Основной недостаток библиотеки в том, что интерфейс для неё существует только для языка C. Следовательно, это значительно усложняет разработку модуля для VDOM Application launcher.

В результате было принято решения отказаться от такого варианта отображение PDF документа вследствие высокой сложности и трудозатратности. Для реализации через интеграцию модуля в VDOM Application launcher основная сложность заключатся в изучении структуры платформы. Реализация через стороннее приложение имеет чуть меньшую сложность, но при этом было бы необходимо иметь приложение для каждой платформы, для которой существует VDOM Application launcher. Трудозатратность заключается в том, что для реализации такого способа приходилось бы скачивать PDF документ с сервера для рендеренга.

Второй вариант – отображения документа при помощи виджета просмотра web документов. Для этого требуются или сторонней JavaScript библиотеки, или плагины.

Из сторонних библиотек на JavaScript самой популярной является PDF.js, разрабатывает некоммерческой организацией Mozilla Foundation. Принцип работы библиотеки - PDF документ отрисовывается в набор HTML5 страниц. Для этого библиотеке требуется URL документа. Основной проблемой этой библиотеки является слабая поддержка кроссдоменных запросов. Такая возможность не предусмотрена по умолчанию. Так как PDF.js запускается с такими же правами, как и любой другой JavaScript код, следовательно, это означает, что кросс доменные запросы не могут быть осуществлены. Но существует возможность обойти это ограничение при помощи таких систем, как CORS [12]. CORS (Cross-Origin Resource Sharing) – механизм, позволяющий получить доступ к кроссдоменным запросам [13]. В кон-

тексте решаемой задачи такое ограничение является очень существенным из-за того что в ProShare не предусмотрена работа с кроссдоменными запросами. Следовательно, для использования PDF.js требуется скачивание PDF документа с сервера и сохранение его на диске.

Самым популярным плагином для отображения PDF документов в браузере является Adobe PDF. Этот плагин поставляется вместе с приложениями Adobe Acrobat и Adobe Reader. Указанные приложения является кроссплатформенными. Главным преимуществом этого плагина – отсутствие проблемы кроссдоменных запросов. Следовательно, для отображения документа требуется только прямая URL ссылка на документ. Недостатком является необходимость наличия в системе установленного приложения Adobe Acrobat или Adobe Reader. Но этот недостаток не столь существенен за счёт того, что Adobe Reader – свободно распространяемое программное обеспечение.

В результате из рассмотренных вариантов был выбран вариант с плагином Adobe PDF.

2.6 Очередь

Что делать, если пользователю необходимо поработать с документом, который в данный момент заблокирован кем-то другим? Для этого был предусмотрен механизм очередей.

Очередь для каждого отдельного пользователя своя. Она хранится на стороне клиента. При выборе заблокированного документа, он добавляет в очередь к пользователю. При этом, пользователь может работать с другими документами. При синхронизации состояния хранилища документов, если документ из очереди окажется свободен, то на сервер будет послан запрос на блокирование этого документа текущим пользователем. Вследствие того, что очередь хранится на стороне клиента, документ из очереди будет заблокирован тем пользователем, для которого синхронизация состояния хранилища после разблокирования документа произойдёт раньше.

На рисунке 6 представлено схематическое изображение механизма очереди.

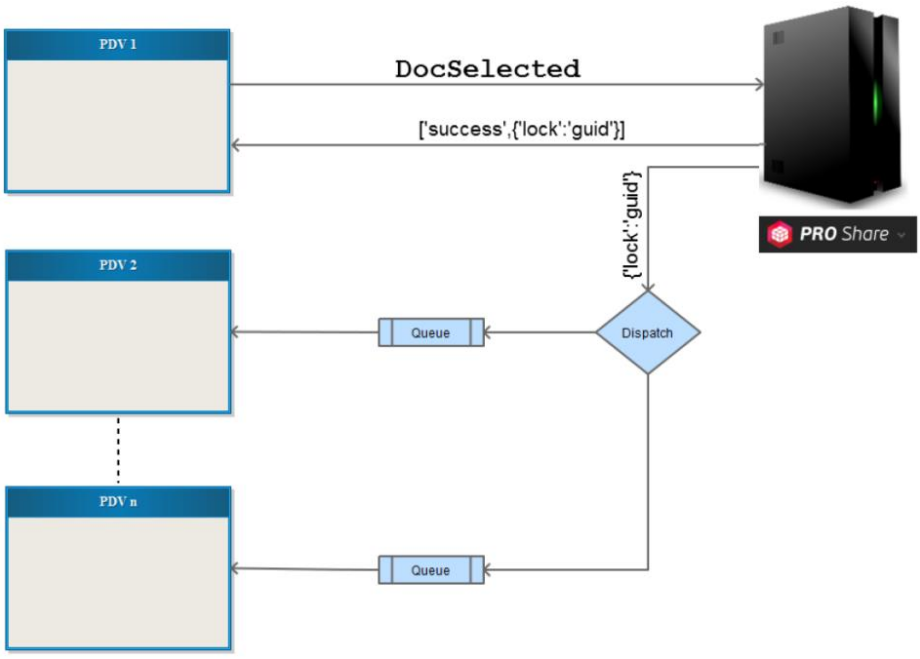

Рисунок 6 — Схематическое изображение механизма очереди

2.7 Архитектура приложения

Приложение построено по модели клиент-сервер. Сервером, как уже было сказано, является сервер ProShare. Клиент при этом будет толстым, т.е. все вычисления будут проводиться на стороне клиента, и на сервер будут отправляться только результирующие данные.

На рисунке 7 приведена диаграмма классов соответствующая приложению.

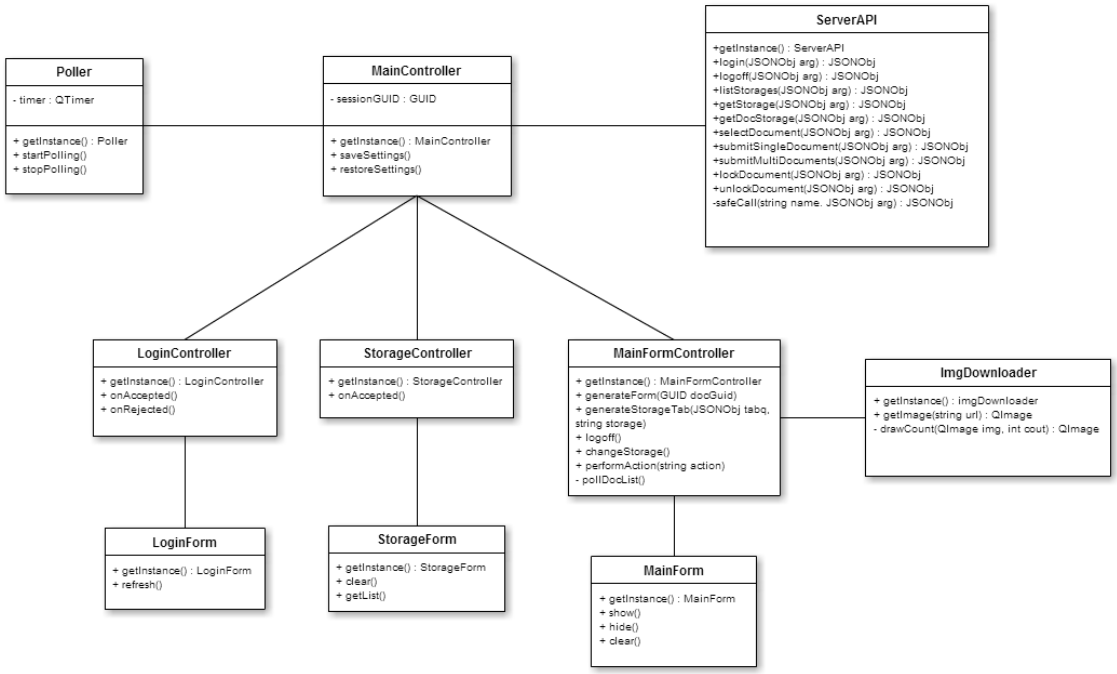

Рисунок 7 — диаграмма классов

Каждый класс является одиночкой. Это необходимо, чтобы точно передать состояние объектов приложения. Язык ECMAScript, и основанный на нём QtScript, не содержат такого понятия, как класс. Но при этом язык является объектным.

3. Реализация приложения Document Viewer

3.1 Виртуальные хранилища документов

Список виртуальных хранилищ можно получить при помощи ProShare API благодаря методу **list_storages**. Метод не принимает аргументов, возвращает JSON массив. Элементами этого массива являются массивы вида:

[Название хранилища, адрес логотипа]

Если у хранилища нет логотипа, то второй элемент будет пустой строкой. Пример ответа приведён в приложении А.

На рисунке 8 приведён скриншот окна выбора виртуального хранилища.

Рисунок 8 — Диалог выбора виртуального хранилища

После выбора виртуального хранилища пользователь попадает в основное окно приложения. По умолчанию виджет с контейнерами и хранилищами документов расположен в верхней части окна. Каждый контейнер представлен вкладкой. Каждая вкладка содержит кнопки. Кнопка представляет хранилище документов. Рисунок на кнопке – логотип хранилища документов и количество документов (если оно не пустое). Подпись к кнопке – название хранилища.

На рисунке 9 приведён скриншот виджета.

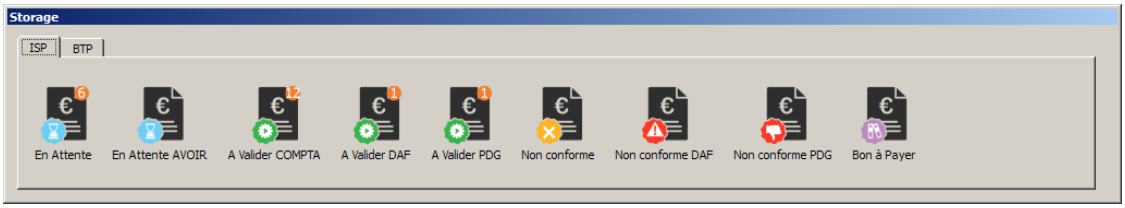

Рисунок 9 — Виджет контейнеров виртуального хранилища

Список контейнеров и хранилищ документов для выбранного виртуального хранилища может быть получен благодаря API методу **get_storage.** В качестве аргумента этот метод принимает название виртуального хранилища. Возвращаемое значение – JSON массив, каждый элемент которого представляет собой информацию о конкретном контейнере. Информация о контейнере – JSON массив следующего вида:

[Название контейнера, права текущего пользователя, JSON массив хранилищ документов]

Права текущего пользователя могут быть “w” и “r”. Пользователь с правами “w” может просматривать документы в хранилище, редактировать их метаданные и осуществлять действия с документами. Пользователи с правами “r” могут только просматривать документы.

JSON массив для хранилища документов имеет следующий вид:

[Название хранилища, адрес логотипа, количество документов]

Логотип для каждого хранилища скачивается с сервера и кэшируется для упрощения процедуры отрисовки кнопок, представляющих хранилище документов.

Пример ответа метода **get_storage** приведён в приложении А.

При нажатии на кнопку хранилища будет загружен список документов указанного хранилища. Список может быть получен благодаря API методу **get_doc_storage**. В качестве аргумента этот метод принимает JSON объект с членами path (путь к хранилищу в формате */Virtual Storage/Container/Document Storage/*) и type (является ли это обращением от пользователя или автоматическим запросом для обновления состояния хранилища). Метод вернёт JSON массив. Каждый элемент массива описывает конкретный документ и имеет вид:

[Название документа, дата добавления, guid, статус документа]

Каждый документ обладает guid. “GUID (Globally Unique Identifier) — статистически уникальный 128-битный идентификатор. Его главная особенность — уникальность, которая позволяет создавать расширяемые сервисы и приложения без опасения конфликтов, вызванных совпадением идентификаторов. Хотя уникальность каждого отдельного GUID не гарантируется, общее количество уникальных ключей настолько велико (2^{128} или $3{,}4028 \times 10^{38}$), что вероятность того, что в мире будут независимо сгенерированы два совпадающих ключа, крайне мала” [14].

Возможные статусы документа в системе:

- Свободен. Такой статус обозначает, что данный документ доступен для блокирования, и ни один пользователь не осуществляет работу с этим документом.
- Заблокирован. Этот статус обозначает, что пользователь осуществляет работу с документом. В таком случае в ответе будет указан guid приложения, в котором был заблокирован этот документ.

Пример ответа метода **get_doc_storage** приведён в приложении А.

Для представления конкретного хранилища документов используется отдельный виджет. По умолчанию этот виджет находится в левой части окна приложения. Он состоит из двух частей:

1. Строка поиска.

2. Представление списка документов.

Строка поиска позволяет осуществлять поиск в текущем хранилище. Параметрами поиска могут быть строка или дата. В случае поиска по строке будут отображены только те документы, название которых содержит строку поиска. Этот вид поиска является регистронезависимым. Для поиска по дате содержимое строки поиска должно иметь следующий вид:

Логическая операция, дата в формате dd/MM/yy (день, месяц, последние две цифры года)

Реализованы следующие логические операции:

- Больше. Обозначается “>”. При использовании этой операции будут выведены документы, добавленные в систему после указанной даты.

- Меньше. Обозначается “<”. При использовании этой операции будут выведены документы, добавленные в систему до указанной даты.

- Больше или равно. Обозначается “>=”. При использовании этой операции будут выведены документы, добавленные в систему начиная с указанной даты.

- Меньше или равно. Обозначается “<=”. При использовании этой операции будут выведены документы, добавленные в систему по указанную дату включительно.

- Равно. Обозначается “=”. При использовании этой операции будут выведены документы, добавленные в систему в указанную дату.

- Не равно. Обозначается “<>”. При использовании этой операции будут выведены все документы, за исключением добавленных в систему в указанную дату.

Комбинирование поиска по дате и названию невозможно. При поиске документы не будут отсортированы.

Примеры строк для поиска:

- >02.01.15. Будут выведены только документы, добавленные в систему после второго января 2015-го года.

- Конт. Будут выведены документы, название которых содержит “конт”.

Перейдём ко второй части виджета – списку документов. В ней отображаются в столбик информация о каждом документе. Информация о документе состоит из превью документа (уменьшенное изображение первой страницы), названия документа и даты добавления в систему.

Пример виджета приведён на рисунке 10.

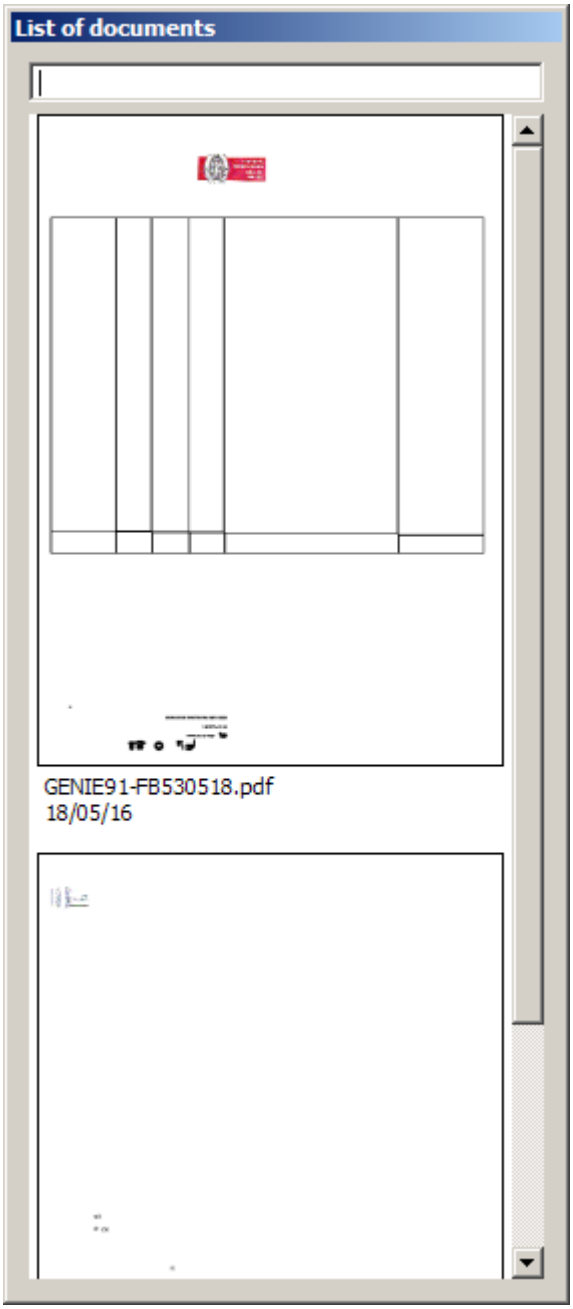

Рисунок 10 — Пример виджета списка документов

3.2. Механизм обновления и синхронизации состояния хранилищ

Вследствие того, что приложение является многопользовательским, возникает проблема актуальности состояния хранилища. Предположим, что некоторый пользователь закончил работу с документом, в результате чего, документ был перемещён в другое хранилище документов. В таком случае должно измениться состояние хранилища, из которого был убран документ, и хранилища, в которое документ был помещён.

Для решения этой проблемы был разработан механизм обновления и синхронизации состояния хранилища. Этот механизм основан на поллинге (англ. polling). Каждые 10 секунд по таймеру приложение отправляет на сервер запрос к API методу **get_doc_storage**. Член

type JSON объекта аргумента метода принимает значение “true”. Полученный список документов сравнивается с текущим списком. Если документ изменил статус, то иконка статуса у текущего документа изменится. В случае перемещения документа из текущего хранилища, документ будет удалён из списка. При этом если пользователь просматривал документ, который был удалён, то просмотр документа для этого пользователя прекратится. При появлении нового документа он будет добавлен в текущий список и информация о документе будет выведена в соответствующий виджет.

3.3. Работа с документами

Работа может осуществляться с одним документом или с несколькими.

При работе с одним документом пользователь может редактировать метаданные объекта и осуществлять действия с этим документом, заложенные в системе. Для осуществления работы с одним документом пользователь должен выбрать конкретный документ из списка в советующим виджете. При этом будет показан документ, сгенерирована форма документа и список возможных действий. При изменении значения какого-либо из компонентов формы или осуществлении действия документ будет заблокирован. При выборе другого документа в режиме работы с одним документом или при успешном завершении действия документ будет разблокирован.

При работе с несколькими документами возможно только осуществлять действие с ними. При этом каждый выбранный документ будет заблокирован. Просмотр конкретного документа или редактирование объектов его формы невозможно. Документ будет разблокирован при повторном нажатии на него в режиме работы с несколькими документами. При успешном осуществлении действия с документами все они будут разблокированы.

3.4. Блокирование документа

Как уже было сказано ранее документ в системе может иметь два статуса: заблокирован или свободен. В процессе работы документ может менять свой статус. Для этого используются API методы lock_document и unlock_document.

Метод **lock_document** позволяет проверить статус документа, при этом, если статус документа – свободен, то документ будет заблокирован пользователем, который вызвал этот метод. В качестве аргумента метод принимает guid документа. Возвращаемое значение – JSON массив вида:

[статус операции, guid владельца]

Статус операции может быть “owned” (в результате выполнения документ был заблокирован) и “locked” (документ уже кем-то заблокирован).

Метод **unlock_document** позволяет разблокировать документ. Метод принимает guid документа. Возвращает статус выполненной операции. Статус “ok” означает, что документ с указанным guid был разблокирован. Статус “error” означает, что документ с указанным guid не был заблокирован пользователем, который запрашивает разблокирование.

Вследствие того что lock_document имеет два возможных статуса, то для пользователя получается больше возможных вариантов состояния документа. Если в системе преду-

смотрено только два состояния: свободен и заблокирован, то для конкретного пользователя статус документа “заблокирован” подразделяется на два: заблокирован этим пользователем и заблокирован другим пользователем.

3.5 Форма документа

Для этого для каждого документа генерируется виджет формы документа. Пример виджета документа приведён на рисунке 11.

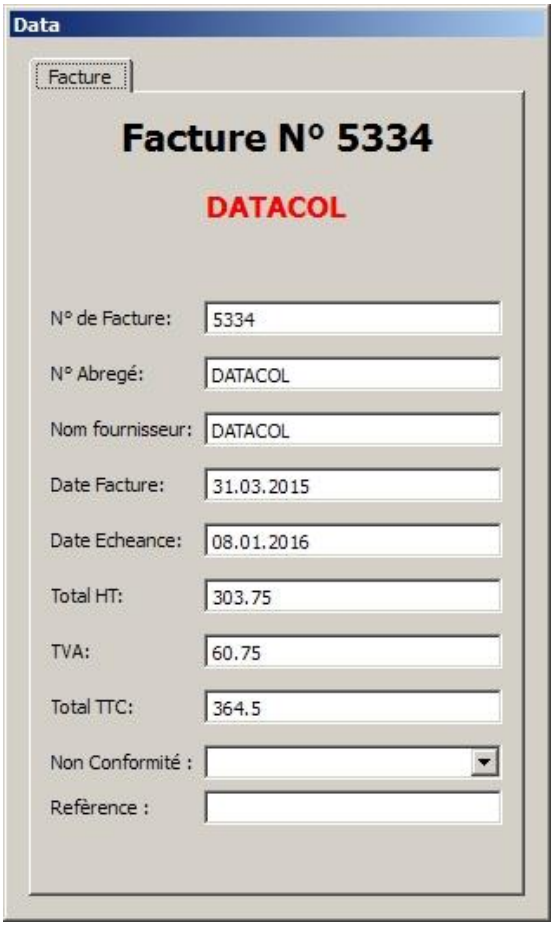

Рисунок 11 — Пример виджета формы документа

Для получения формы документа используется API метод **select_document**. В качестве параметрf метод принимает JSON объект с членами path – путь к документу в формате virtual storage/container/document storage и source – guid документа в системе. Метод возвращает JSON массив с тремя элементами. Первый элемент – статус документа в системе. Второй элемент – JSON массив из двух элементов: JSON массив для формы и JSON массив для виджета действий с документом. Третий элемент – JSON объект для значений компонентов формы по умолчанию. Пример ответа метода **select_document** приведён в приложении А.

Рассмотрим JSON массив для формы. Каждый нечётный элемент массив содержит название вкладки на форме. Каждый чётный элемент – JSON массив JSON объектов компонентов расположенных на вкладке. Каждый такой объект обязательно содержит члены type – тип элемента, name – название компонента, label – название поля. Остальные члены зависят от типа.

Типы компонентов формы:

1. Текстовое поле (англ. TextField).
2. Многострочное текстовое поле (англ. MultilineTextField).
3. Поле выбора (Combobox).
4. Список выбора (List of checkboxes).
5. Гипертекст (англ. HyperText).

Для каждого компонента, кроме гипертекста, существует набор условий проверки значения поля. В общем виде условие можно записать как:

$$\alpha \text{ операция } \beta$$

α и β могу быть условиями проверки, константами и встроенными функциями. Встроенные функции: type – возвращает тип поля, принимает имя поля, action – возвращает название выбранного действия, не требует аргументов. Операцией являются операции сравнения: больше, меньше, равно, не равно, больше или равно, меньше или равно и логические операции конъюнкция и дизъюнкция для условий проверки. В объекте компонента формы условия проверки представлены в члене checkRule – массив с элементами вида:

[условие проверки, текст ошибки]

Объект с типом текстовое поле содержит дополнительные члены: mandatory со значение true (поле обязательно должно быть заполнено) или false (поле может быть не заполнено) и mask – маска ввода. Маски ввода бывают трёх типов: string – строка, numeric - вещественное число, data – дата в формате dd/MM/yyyy (день, месяц, год). Соответствие вводимого значения маске проверяется в момент ввода очередного символа поля. В случае несоответствия поле будет подсвечено красным и выведена ошибка. Пример поля приведён на рисунке 12.

Рисунок 12 — Пример текстового поля

Пример поля с ошибкой приведён на рисунке 13.

Рисунок 13 — Пример текстового поля с ошибкой

Пример JSON объекта с типом текстовое поле приведён в листинге 1.

Листинг 1. Пример JSON объекта с типом текстовое поле

```
{
    "mandatory":true,
    "name":"tva",
    "DefaultValue":"",
    "mask":"numeric",
```

```
        "label":"TVA:",
        "checkRule":[
                [
                        "type(me)='numeric'",
                        "La valeur doit être une valeur numérique"
                ]
        ],
        "type":"TextField"
}
```

Объект с типом многострочное текстовое поле аналогичен объекту с типом текстовое поле за исключением маски. Пример JSON объекта для многострочного текстового поля приведён в листинге 2.

Листинг 2. Пример JSON объекта для многострочного текстового поля

```
{
        "mandatory":true,
        "name":"comment",
        "DefaultValue":"",
        "label":"Commentaires:",
        "checkRule":[
                [
                        "type(me)='string'",
                        "La valeur doit être une chaîne"
                ]
        ],
        "type":"MultilineTextField"
}
```

Объект с типом поле выбора содержит дополнительный член defaultValue – список возможных значений поля. На рисунках 14 и 15 приведён пример поля выбора

Рисунок 14 — Поле выбора

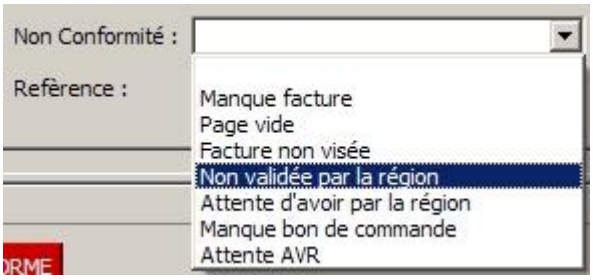

Рисунок 15 — Поле выбора с выпадающим списком

Пример JSON объекта с типом поле выбора приведён в листинге 3.

Листинг 3. Пример JSON объекта с типом поле выбора

```
{
    "DefaultValue":{
    "":"RIEN",
    "Manque facture":"NON CONFORME",
    "Page vide":"NON CONFORME",
    "Facture non visée":"NON CONFORME",
    "Non validée par la région":"NON CONFORME",
    "Attente d'avoir par la région":"AVOIR",
    "Manque bon de commande":"NON CONFORME",
    "Attente AVR":"AVOIR"
    },
    "checkRule":[
        [
            "(action()='si_non_conforme_compta' and me<>'') or action()<>'si_non_conforme_compta'",
            "Vous avez cliqué sur non conformité, vous devez en définir une."
        ],
        [
            "(action()='si_non_conforme_daf' and me<>'') or action()<>'si_non_conforme_daf'",
            "Vous avez cliqué sur non conformité, vous devez en définir une."
        ],
        [
            "(action()='si_non_conforme_pdg' and me<>'') or action()<>'si_non_conforme_pdg'",
```

```
                "Vous avez cliqué sur non conformité, vous devez en définir une."
            ]
        ],
        "type":"ListField",
        "name":"nonConformity",
        "label":"Non Conformité :"
    }
```

Объект с типом список выбора содержит дополнительный член defaultValue – список опций, каждое из которых может быть отмечено. На рисунке 16 приведен пример списка выбор без отмеченных значений.

Рисунок 16 — Пример списка выбора

Пример JSON объекта с типом список выбора приведён в листинге 4.

Листинг 4. Пример JSON объекта с типом список выбора

```
{
    "name":"nonConformity:",
    "DefaultValue": [
            "Choix1",
            "choix1",
            "Choix2",
            "choix2",
            "Choix3",
            "choix3",
            "Choix4",
            "choix4"
    ]
    "type":"ListOfChecboxes"
}
```

Объект с типом гипертекст содержит дополнительный член align – выравнивание текста. Выравнивание текста может быть по правому краю (значение right), левому краю (значение left) и по центру (значение center). Текст этого компонента указан в члене content. Также у такого объекта отсутствует член label. Пример компонента гипертекст с выравниванием по центру приведён на рисунке 17.

Рисунок 17 — Пример гипертекста

Пример JSON объекта с типом гипертекст приведён в листинге 5.

Листинг 5. Пример JSON объекта с типом гипертекст

```
{
    "content":"<H1>Facture N° FB530518</H1><H2><FONT Color="#FF0000">Genie Flexion</FONT></H2><BR>",
    "align":"center",
    "type":"HyperText",
    "name":"title"
}
```

Для генерации виджета формы документа был разработан алгоритм на основе детерминированного конечного автомата. Дадим определение детерминированному и недетерминированному конечным автоматам.

“Недетерминированный конечный автомат – это пятёрка $M = (Q,\ \Sigma, \delta,\ q_0,\ F)$, где

(1) Q – конечное множество состояний;
(2) Σ – конечное множество допустимых входных символов;
(3) δ – отображение множества $Q \times \Sigma$ в множество $\mathrm{P}(Q)$, определяющее поведение управляющего устройства; функцию δ иногда называют функцией переходов;
(4) $q_0 \in Q$ – начальное состояние управляющего устройства;
(5) $F \subseteq Q$ – множество заключительных состояний;

Пусть $M = (Q,\ \Sigma, \delta,\ q_0,\ F)$ – недетерминированный конечный автомат. Назовём автомат M детерминированным, если множество $\delta(q,\ \alpha)$ содержит не более одного состояния для любых $q \in Q$ и $\alpha \in \Sigma$”. [15]

В нашем случае множество состояний Q будет состоять из состояний:

- Отсутствие входа.
- Распознано текстовое поле (TF).
- Распознано многострочное текстовое поле (MTF).
- Распознано поле выбора (Cb).
- Распознан список выбора (LoC).

- Распознан гипертекст (HT).

Входом для алгоритма является массив JSON объектов компонентов формы. Начальное состояние q_0 определено как отсутствие входа. На основе значения члена type определяется, какой компонент должен быть сгенерирован. Следовательно, множество Σ состоит только из возможных значений type: $\Sigma = \{\text{textfield}, \text{multilinetextfield}, \text{combobox}, \text{listofcheckboxes}, \text{hypertext}\}$. Тип очередного компонента не зависит от предыдущего, следовательно, значение функции переходов δ зависит только от входного символа. Соответствие входного символа и следующего состояние автомата приведено в таблице 1.

Таблица 1. Соответствие входного символа и следующего состояния автомата

Входной символ	Состояние автомата
textfield	Распознано текстовое поле
multilinetextfield	Распознано многострочное текстовое поле
combobox	Распознано поле выбора
listofcheckboxes	Распознан список выбора
hypertext	Распознан гипертекст

Алгоритм завершает работу, когда просмотрены все элементы входящего массива. Следовательно, множество заключительных состояний $F = Q$.

Схема конечного автомата приведена на рисунке 18.

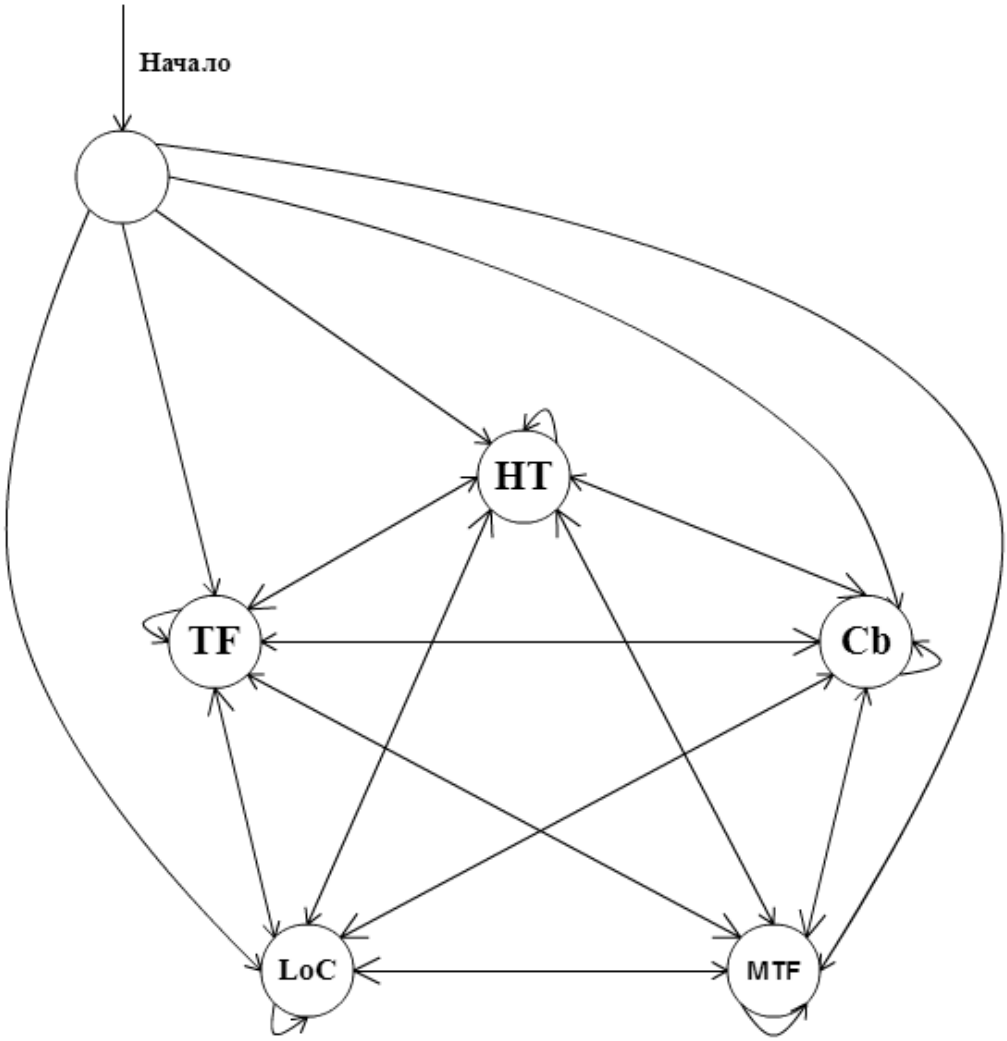

Рисунок 18 — Схема конечного автомата

Разработанный алгоритм можно описать так:

Дан массив из N JSON объектов, каждый из которых описывает один компонент формы.

Шаг 1. В цикле для каждого элемента выполнить шаги 2 – 4.

Шаг 2. Определить тип очередного компонента.

Шаг 3. Если очередной компонент имеет тип гипертекст, то создать объект класса label, тип текста в созданном объекте установить как гипертекст, применить выравнивание согласно значению члена align, установить текст. Иначе, поместить в созданный объект условия проверки значения, создать объект класса label, текст в созданном объекте установить согласно названию поля. Проверить есть ли для этого компонента значение по умолчанию. Для компонента типа текстовое поле связать изменение содержимого поля с функцией проверки маски этого поля. Для компонента типа поле выбора заполнить список возможных значений.

Шаг 4. Созданный объект разместить компонента на форме.

3.6 Действия с документом

Действия с документом – сценарий (скрипт), исполняемый на стороне сервера ProShare. Сценарии создаются владельцем документа. Каждый сценарий имеет своё название. Один документ может как иметь множество сценариев, так и не иметь ни одного.

Для осуществления действий с документом пользователю предоставляется виджет возможных действий с документом. Виджет генерируется в соответствии со списком действий. Список действий с документом доступен через API метод **select_document.** Фрагмент ответа метода – JSON массив с JSON объектов. Каждый объект описывает одно действия. Основные члены: цвет, название кнопки, название действия, которая представляет кнопка. Действие нажатие кнопки связывается с отправкой данных на сервер. Пример виджета с действиями приведён на рисунке 19.

Рисунок 19 — Пример виджета действий с документом

Алгоритм осуществления действий с одним документом состоит из двух основных частей: проверка условий на значение компонентов форм, создание JSON объекта и отправка данных на сервер. Рассмотрим алгоритм проверки условий. Этот алгоритм можно описать следующим образом:

Шаг 1. Для каждого компонента формы создать переменную с названием компонента формы. В созданную переменную положить значение компонента.

Шаг 2. В цикле для каждого компонента формы проверить корректность условий на значение. Каждое условие переводится на язык исполнителя. Далее вычисляется его значение в булевом формате (истина или ложь). Если результатом является истина, то значение поля соответствует условию. Иначе ошибка, соответствующие этому условию, записывается в список ошибок. Поле, в котором была допущена ошибка, будет подсвечено красным (аналогично несоответствие значения поля маски) и под ним будет выведена последняя ошибка, произошедшая в этом поле.

Если хотя бы в одном из условий не выполнено, то пользователю будет выведено сообщение со списком ошибок.

Если ошибок не было, то будет создан JSON объект с двумя обязательными членами: action (название действия), guid (guid документа, над которым совершается действия). Остальные члены будут соответствовать компонентам формы: название соответствует названию поля, значение – значению. Для текстового поля и многострочного текстового поля значением является текст, записанный в эти поля, для поля выбора – текущий элемент, для списка выбора – список выделенных элементов. Если текстовое поле содержит дату, то она будет переведена в формат yyyy/MM/dd (год, месяц, день)

Отправка данных на сервер для одного документа осуществляется API методом **submit_single_document**. Этот метод принимает в качестве аргумента JSON объект описанный выше. Возвращаемое значение – статус операции: успешно выполнена операция или нет.

Рассмотрим пример проведения действия. На рисунке 20 приведена правильно заполненная форма документа.

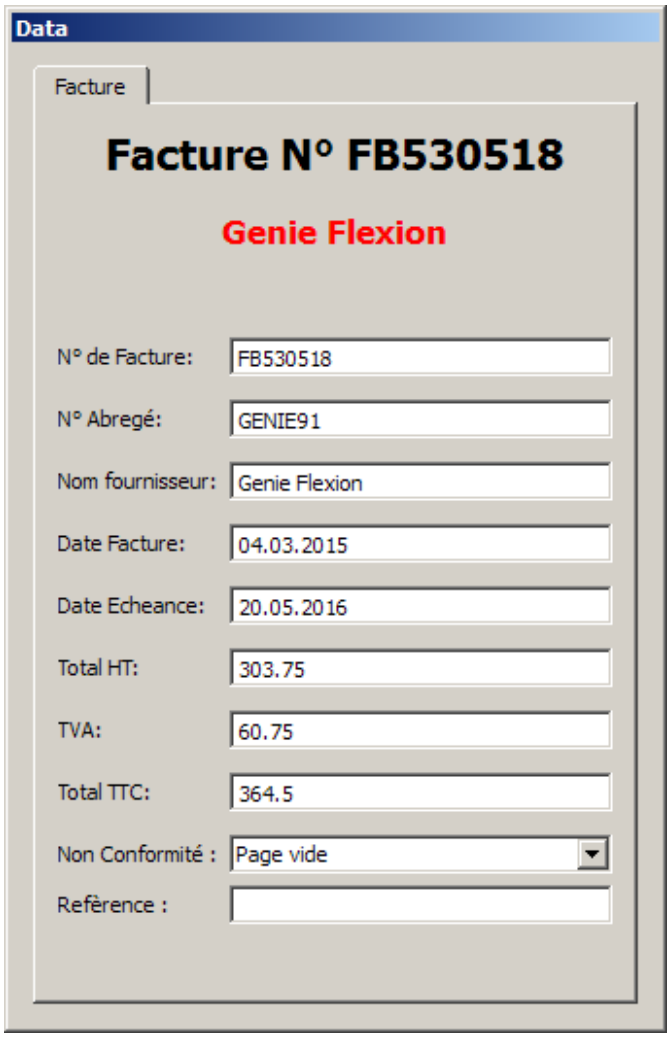

Рисунок 20 — Пример формы документа

В таблице 2 приведено соответствие названия компонента названию поля для формы, приведённой на рисунке 20.

Таблица 2. Соответствие названия компонента названию поля для формы

Название компонента	Название поля
numFacture	N° de Facture
numAbrege	N° Abregé
nomFournisseur	Nom fournisseur
InvoiceDate	Date Facture
InvoiceEnd	Date Echeance
ht	Total HT
tva	TVA

ttc	Total TTC:
nonConformity	Non Conformité
reference	Refèrence

Возможные действия с выбранным документом приведены на рисунке 21.

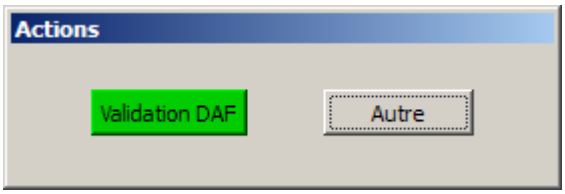

Рисунок 21 — Пример виджета действий

Пусть выбрано действие “Validation DAF”. Название действия - si_ok_DAF. Выбранный документ имеет guid aa7968f1-1a94-4eca-8aef-f24d871cb9e3. Созданный JSON объект приведён в листинге 6.

Листинг 6. Пример JSON объекта для одного документа

```
{
    “event”: “PDVSingleFilePost”,
    “document”: “aa7968f1-1a94-4eca-8aef-f24d871cb9e3”,
    “action”: “si_ok_DAF”,
    “form-data”: {
        “numFacture”: “FB530518”,
        “numAbrege”: “GENIE91”,
        “nomFournisseur”: “Genie Flexion”,
        “InvoiceDate”: “2015.03.04”,
        “InvoiceEnd”: “2015.05.20”,
        “ht”: “303.75”,
        “tva”: “60.75”,
        “ttc”: “364.5”,
        “nonConformity”: “Page vide”,
        “reference”: “”
    }
}
```

Для действий с несколькими документами алгоритм выглядит значительно проще. Так как для нескольких выбранных документов нет формы, следовательно, не требуется проверка условий формы. Также изменится и формат JSON объекта, который будет создан для

отправки на сервер. Объект будет иметь два члена: первый – название действия, второй – массив выбранных документов. Каждый элемент этого массива – guid документа.

Отправка данных на сервер для нескольких документов осуществляется API методом **submit_multi_documents**. Этот метод принимает в качестве аргумента JSON объект описанный выше. Возвращаемое значение – статус операции: успешно выполнена операция или нет.

Рассмотрим пример осуществления действия для нескольких документов. Пусть возможные действия с документами совпадают с действиями с одним документом. Документы имеют guid'ы: d10537ba-8929-484a-bcbd-9c4ffe0f3269, a9b3a12f-fab9-4f57-b047-e52cd971fb59, 65a620bc-ee40-4147-a971-d55fbcc8d976, b9b2dc65-3569-4443-a73b-3c2e421c613f", 956fe2bb-fbab-4b14-8d88-62bee163fb2c. Созданный JSON объект приведён в листинге 7.

Листинг 7. Пример JSON объекта для нескольких документов

```
{
    "event": "PDVMultiplyFilePost",
    "action": "NON-CONFORME",
    "filelist": [
        "d10537ba-8929-484a-bcbd-9c4ffe0f3269",
        "a9b3a12f-fab9-4f57-b047-e52cd971fb59",
        "65a620bc-ee40-4147-a971-d55fbcc8d976",
        "b9b2dc65-3569-4443-a73b-3c2e421c613f",
        "956fe2bb-fbab-4b14-8d88-62bee163fb2c"
    ]
}
```

3.7. Пользовательский интерфейс

Перед началом работы с основным окном приложения пользователю необходимо авторизоваться в системе и выбрать виртуально хранилище, в котором он будет работать. Для этого были созданы дополнительные окна: окно авторизации и окно выбора хранилища. Оба окна основаны на компоненте QDialog. Он предоставляет виджет диалогового окна и две основных сигнала: принять и отклонить. Сигнал "принять" для формы авторизации означает, что авторизация прошла успешно, а для формы выбора хранилища – осуществлён вход в хранилище. Для второй формы сигнал "отклонить" не используется. В форме авторизации сигнал "отклонить" означает, что пользователь с таким логином или паролем не может быть авторизован в системе.

Виджеты основного окна приложения должны быть перемещаемые. Для этого виджеты контейнера, списка документов, формы документа и действия с документом сделаны на основе компонента QDockWidget. Это позволит пользователю подстроить интерфейс

наиболее удобным для себя образом. Например, переместить какой-либо виджет на другую позицию внутри окна или вынести его за приделы окна. На рисунке 22 приведён пример интерфейса по умолчанию.

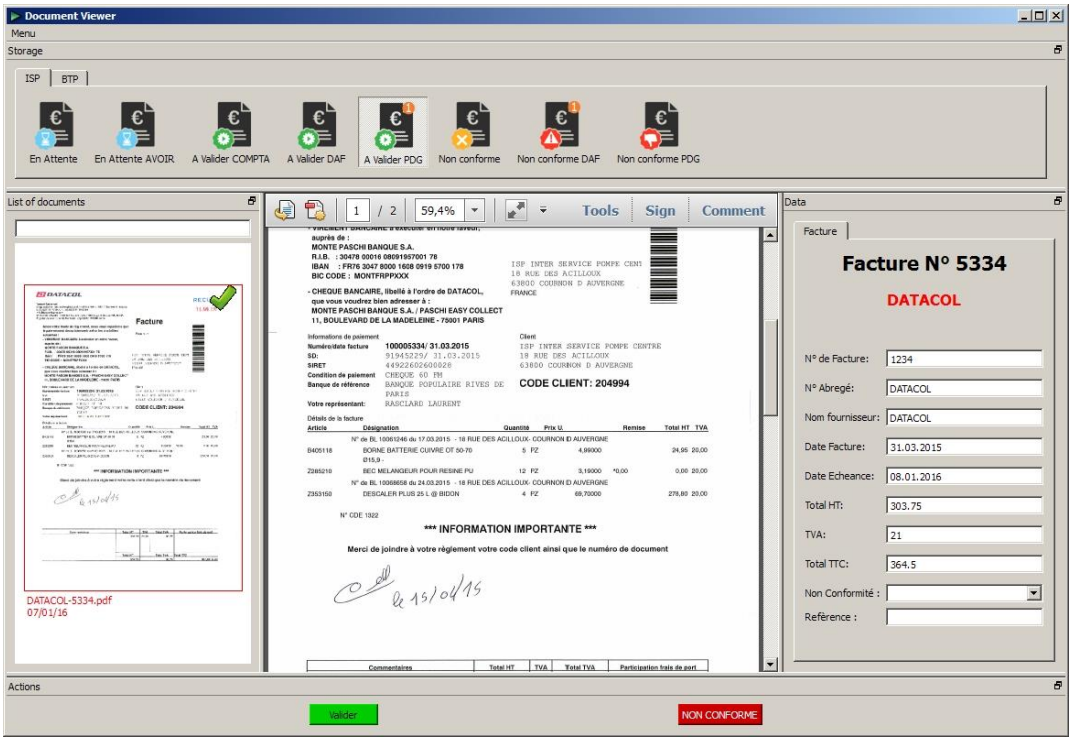

Рисунок 22 — Пример интерфейса по умолчанию

На рисунке 23 приведён пример интерфейса с изменённым расположением виджетов.

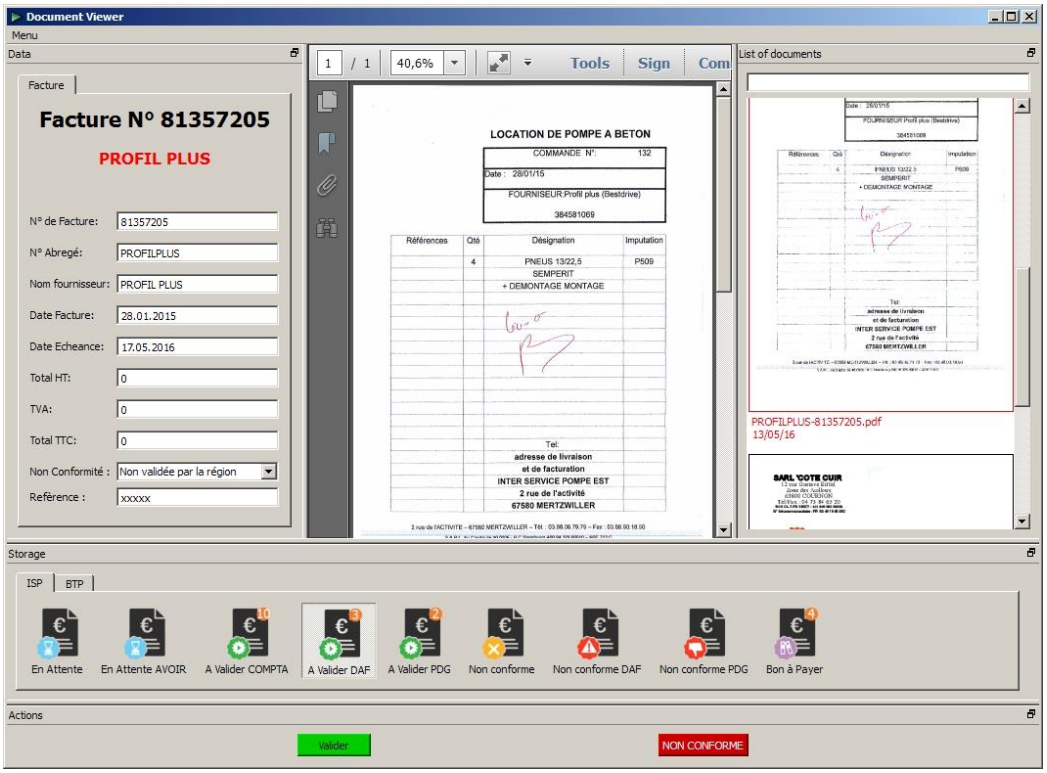

Рисунок 23 — Пример интерфейса с изменённым расположением виджетов

На рисунке 24 приведён пример интерфейса с виджетами, вынесенными за пределы главного окна.

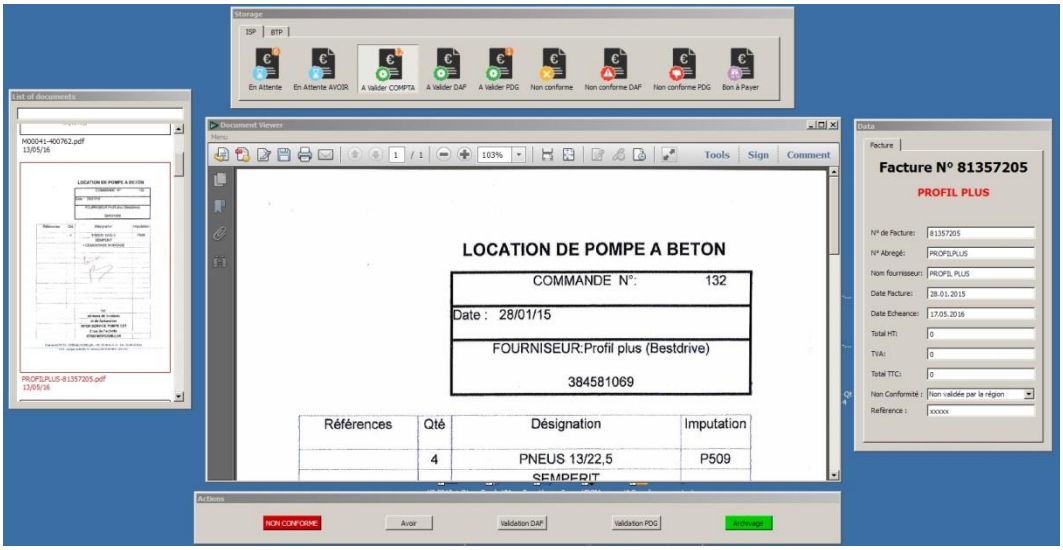

Рисунок 24 — Пример интерфейса с виджетами за пределами главного окна

Для отображения документа используется компонент QWebView с плагином Adobe PDF. Компонент QWebView – виджет, который может быть использован для просмотра и редактирования web-страниц. Виджет просмотра документа по умолчанию находится в центре главного окна приложения. Пример виджета приведён на рисунке 25.

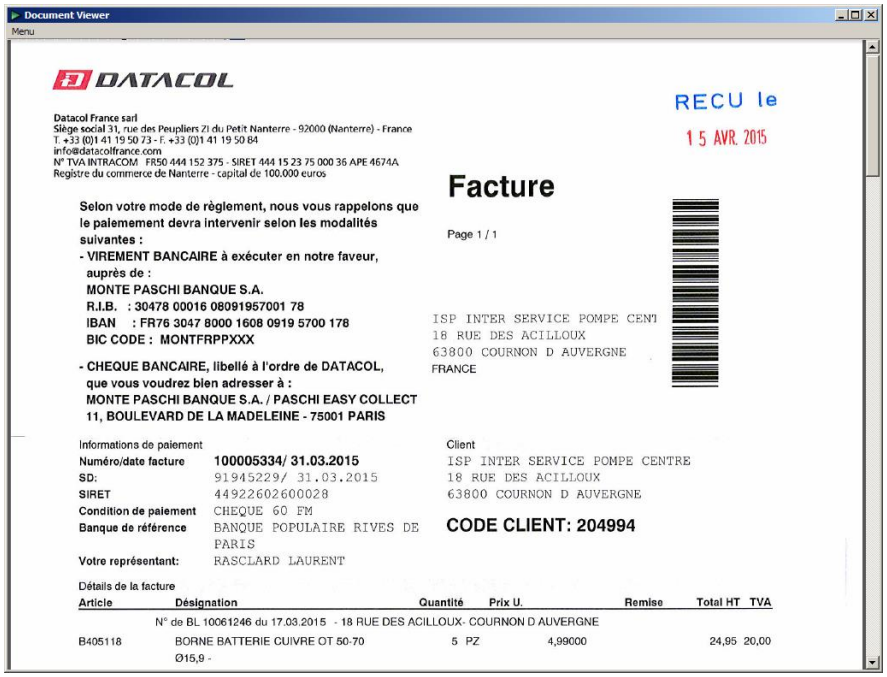

Рисунок 25 — Виджет просмотра документа

Виджет списка документов состоит из компонентов QLineEdit для строки поиска и QGraphicsView для списка документов. Компонент QLineEdit – виджет для редактирования текста, записанного в одну строку. Компонент QGraphicsView – виджет отображения сцен. Сцена генерируется при ходе в хранилище документов. При работе механизма синхронизации состояния хранилища сцена может измениться. Если в результате синхронизации количество документов в хранилище изменилось, то сцена полностью перерисовывается. Иначе

заменяются только те документы, которые были изменены. Превью документов скачиваются объектом imgDownloader и кэшируются для того чтобы в случае полной перерисовки сцены повторно их не скачивать.

Виджет хранилища основан на компоненте QTabWidget. Каждая кнопка хранилища документов - объект с прототипом QPushButton. Для загрузки иконок хранилища документов используется объект imgDownloader. Иконки так же кэшируются. Но в отличие от превью документов иконки скачиваются в асинхронном режиме. Это означает, что виджет хранилища уже может быть сгенерирован и готов к использованию, а иконки хранилищ будут дорисовываться по мере загрузки с сервера. Виджет списка действий использует аналогичные кнопки для действий.

Виджет формы документа так же основан на компоненте QTabWidget. В таблице 3 приведено соответствие компонентов формы и компонентов Qt, используемые для создания формы.

Таблица 3. Соответствие компонентов формы и компонентов Qt

Компонент формы	Компонент Qt
Гипертекст	QLabel
Текстовое поле	QLineEdit
Многострочное текстовое поле	QTextEdit
Поле выбора	QComboBox
Список выбора	QCheckBox

Для названий полей также используется компонент QLabel. Этот компонент используется для отображения текста или изображения. Компонент QTextEdit – является виджетом для отображения и редактирования текста. Компонент QComboBox – виджет, позволяющий выбирать значение из выпадающего списка. Компонент QCheckBox – виджет для кнопки, которая может быть или включена или выключена.

Заключение

Целью работы была разработка приложения для работы с документами в потоке работ. Платформой разработки была выбрана VDOM Desktop Applications.

В результате работы:

- Проведён обзор существующих систем электронного документооборота.
- Разработан программный продукт согласно требованиям заказчика.
- Разработанный программный продукт был протестирован.

Приложение получило название ProShare Document Viewer и было. Акт о внедрении приведён в приложении.

ЛИТЕРАТУРА

1. Янко Д.В. Электронный архив для систем электронного документооборота и систем управления информационными ресурсами предприятия // Проблемы информатики. 2012. № 4 (16). С. 65-88.

2. Управление корпоративным информационным контентом [Электронный ресурс]: офиц. сайт. 2016. URL: https://msdn.microsoft.com/ru-ru/library/ee554868.aspx (дата обращения: 03.03.2016).

3. Автоматизация процесса – Workflow [Электронный ресурс]: Консалтинговая компания “Взгляд Вашего Потребителя” 2016. URL: http://www.regcons.ru/5-step-1-6.htm (дата обращения: 03.03.2016)

4. SharePoint [Электронный ресурс]: Википедия: свободная энцикл. 2016. URL: https://en.wikipedia.org/wiki/SharePoint (дата обращения: 12.03.2016).

5. The History of Notes and Domino [Электронный ресурс]: офиц. сайт. 2016. URL: http://www.ibm.com/developerworks/lotus/library/ls-NDHistory (дата обращения: 15.03.2016).

6. DIRECTUM 5.0 – новое поколение [Электронный ресурс]: офиц. сайт. 2016. URL: http://www.directum.ru/5000079.aspx (дата обращения: 20.03.2016).

7. Андреев А.Д. Разработка web-приложения EYE ON FILE ONLINE НА БАЗЕ VDOM технологий для организации архивов документов- Томск: Томск. гос. ун-т. Факультет информатики, 2012.- 54 с.

8. Qt [Электронный ресурс]: Википедия: свободная энцикл. 2016. URL: https://ru.wikipedia.org/wiki/Qt (дата обращения: 04.10.2015).

9. API [Электронный ресурс]: Википедия: свободная энцикл. 2016. URL: https://ru.wikipedia.org/wiki/API (дата обращения: 05.10.2015).

10. JSON [Электронный ресурс]: JSON офиц. сайт. 2016. URL: http://www.json.org/json-ru.html (дата обращения: 05.10.2015)

11. Portable Document Format [Электронный ресурс]: Википедия: свободная энцикл. 2016. URL: https://ru.wikipedia.org/wiki/Portable_Document_Format (дата обращения: 13.10.2015).

12. Frequently Asked Questions [Электронный ресурс]: Github. 2016. URL: https://github.com/mozilla/pdf.js/wiki/Frequently-Asked-Questions (дата обращения: 03.11.2015).

13. Enable cross-origin resource sharing [Электронный ресурс]: офиц. сайт. 2016. URL: http://enable-cors.org/ (дата обращения 03.11.2015).

14. GUID [Электронный ресурс]: Википедия: свободная энцикл. 2016. URL: https://ru.wikipedia.org/wiki/GUID (дата обращения: 09.10.2015).

15. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Т. 1. Синтаксический анализ. / А. Ахо, Дж. Ульман; Под. ред. В. М. Курочкина; Пер. с англ. В. Н. Агафонова // М.: Мир, 1978.

Приложение А

Примеры ответов API методов

Пример ответа API метода **list_storages**.

[

["Factures", ""]

]

Пример ответа API метода **get_storage**.

[

["ISP", "w", [["En Attente", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=En_Attente.png&type=res", 0],

["En Attente AVOIR", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=En_Attente.png&type=res", 0],

["A Valider COMPTA", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 5],

["A Valider DAF", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 7],

["A Valider PDG", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 5],

["Non conforme", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme.png&type=res", 0],

["Non conforme DAF", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme_DA.png&type=res", 0],

["Non conforme PDG", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme_PDG.png&type=res", 0],

["Bon \u00e0 Payer", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Archive.png&type=res", 6]]],

["BTP", "w", [["En Attente", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=En_Attente.png&type=res", 0],

["En Attente AVOIR", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=En_Attente.png&type=res", 0],

["A Valider COMPTA", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 0],

["A Valider DAF", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 0],

["A Valider PDG", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=a_valider.png&type=res", 0],

["Non conforme", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme.png&type=res", 0],

["Non conforme DAF", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme_DA.png&type=res", 0],

["Non conforme PDG", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Non_conforme_PDG.png&type=res", 0],

["Bon \u00e0 Payer", "/get_plugin_resource?guid=f41b003f-3832-455b-a6cc-a1bac7db0651&name=Archive.png&type=res", 0]]]

]

Пример ответа API метода **get_doc_storage**.

[

["M00041-400762.pdf", "13/05/16", "1175ceb9-431d-4cd2-8005-ae6b3e4c0421", "free", []],

["PROFILPLUS-81357205.pdf", "13/05/16", "90eb3bbf-264a-47b1-9652-fb1f3d8cb982", "free", []],

["ISP_COTEC-45.pdf", "19/05/16", "7e2c4688-bbc1-4db0-b7c7-8db3ed89701c", "free", []],

["FIR-00238197-ocr.pdf", "19/05/16", "990b9592-dff1-43f4-9d72-a0cb5f538def", "free", []],

["M00041-400762-Ocr.pdf", "19/05/16", "8228f4d4-b6f0-4653-847b-bb9149933b7f", "free", []],

["PROFILPLUS-81357205-ocr.pdf", "19/05/16", "728a4f5d-8474-452b-b5f2-f04421b8d85c", "free", []]

["WURTH-203425-ocr.pdf", "19/05/16", "4ba73861-0343-44c8-8c60-2e9bd055a72f", "free", []]

]

Приложение Б

Руководство пользователя

Для запуска приложения после соединения VDOM Application Launcher с сервером необходимо в системном трее кликнуть по значку и выбрать ProShare Document Viewer. Пример приведён на рисунке П1.

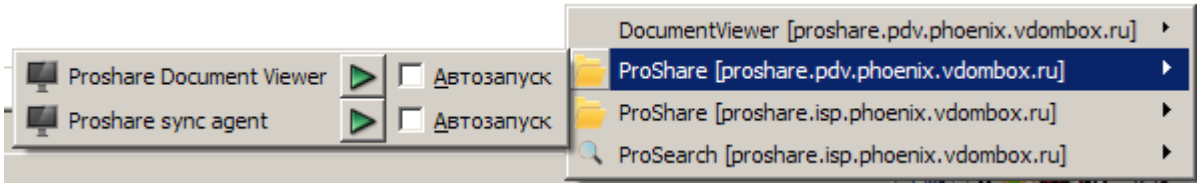

Рисунок П1. Запуск приложения

После запуска приложения появится окно авторизации (рисунок П2). Для того чтобы запомнить данные для входа, следует отметить галочку “save credentials?”. В таком случае при следующем входе в систему, пользователь будет автоматически авторизован по сохранённым данным.

Рисунок П2. Окно авторизации

Далее следует выбрать одно из доступных виртуальных (рисунок П3). Для этого нужно щелчком мыши выбрать одно из хранилищ и кнопку “sing in”. Поле “keep this choice” аналогично сохранению данных для входа.

Рисунок П3. Окно выбора виртуального хранилища

После этого на экране появится основной интерфейс приложения (рисунок П4). Для начала работы следует выбрать один из контейнеров. Они представлены закладками на виджете хранилища (рисунок П5). Далее следует выбрать одно из хранилищ документов. Каждому хранилищу соответствует кнопка, содержащаяся во вкладке контейнера.

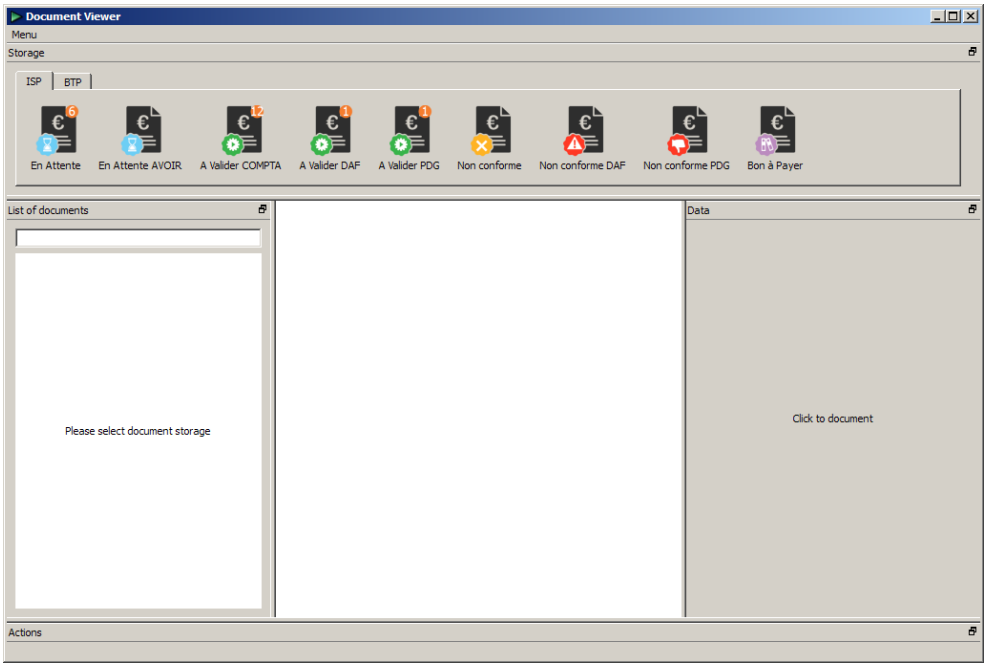

Рисунок П4. Главное окно приложения

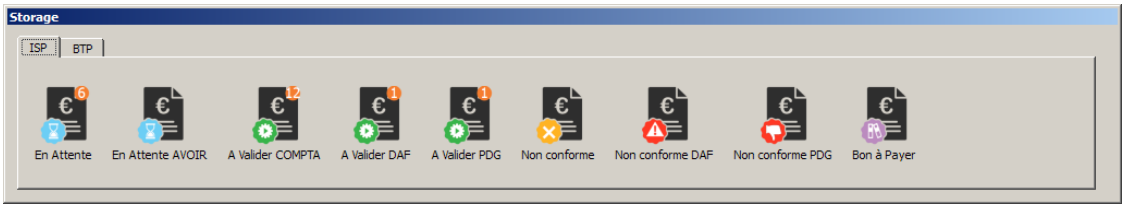

Рисунок П5. Виджет хранилища

После того, как завершится получение данных с сервера, в виджете списка документов будут отображены документы, содержащиеся в выбранном хранилище (рисунок П6).

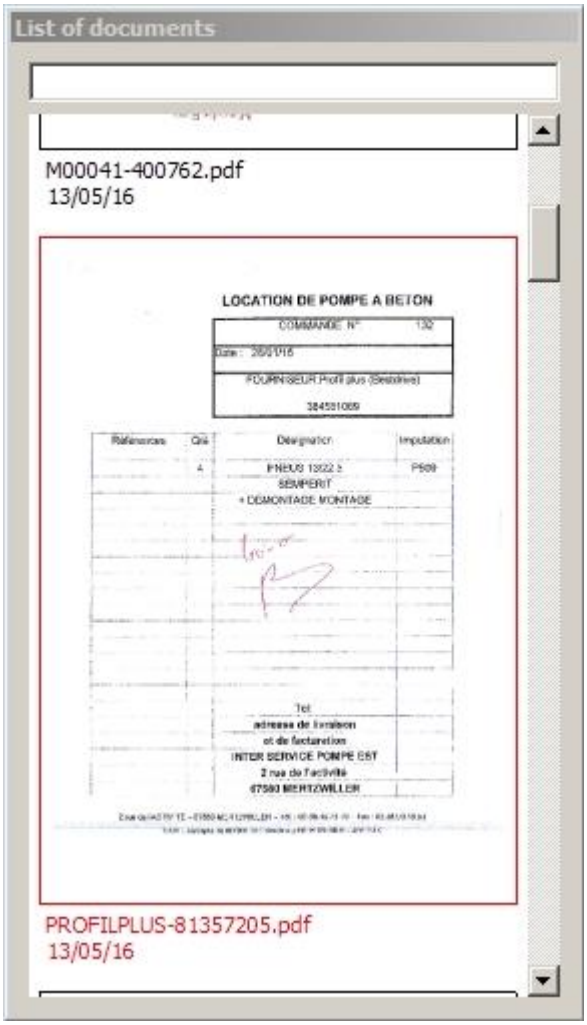

Рисунок П6. Виджет списка документов

Чтобы приступить к работе с документом в режиме работы с одним документом правой кнопкой мыши нужно кликнуть по документу. После этого в виджете просмотра документа будет отображен весь документ, в виджет формы документа будет загружена форма выбранного документа и в виджет действий загружены возможные действия с документом. Главное окно приложения при работе с одним документом приведено на рисунке П7. Если требуется завершить работу с документом, не совершая никаких действий, то необходимо повторно кликнуть на документ.

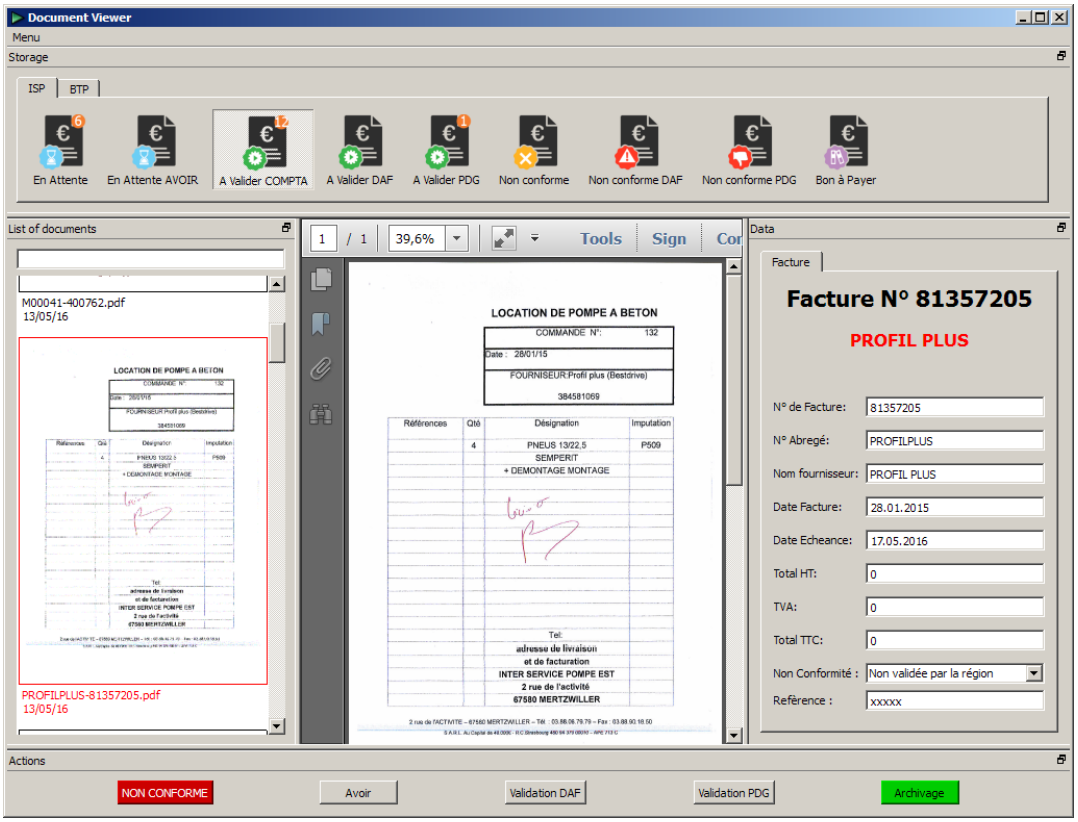

Рисунок П7. Окно приложения при работе с одним документом

Для начала работы с документами в режиме нескольких документов необходимо с зажатой клавишей контрол кликнуть по документу. Аналогично для добавления и удаления документа из списка работы. Главное окно приложения при работе с несколькими документами приведено на рисунке П8.

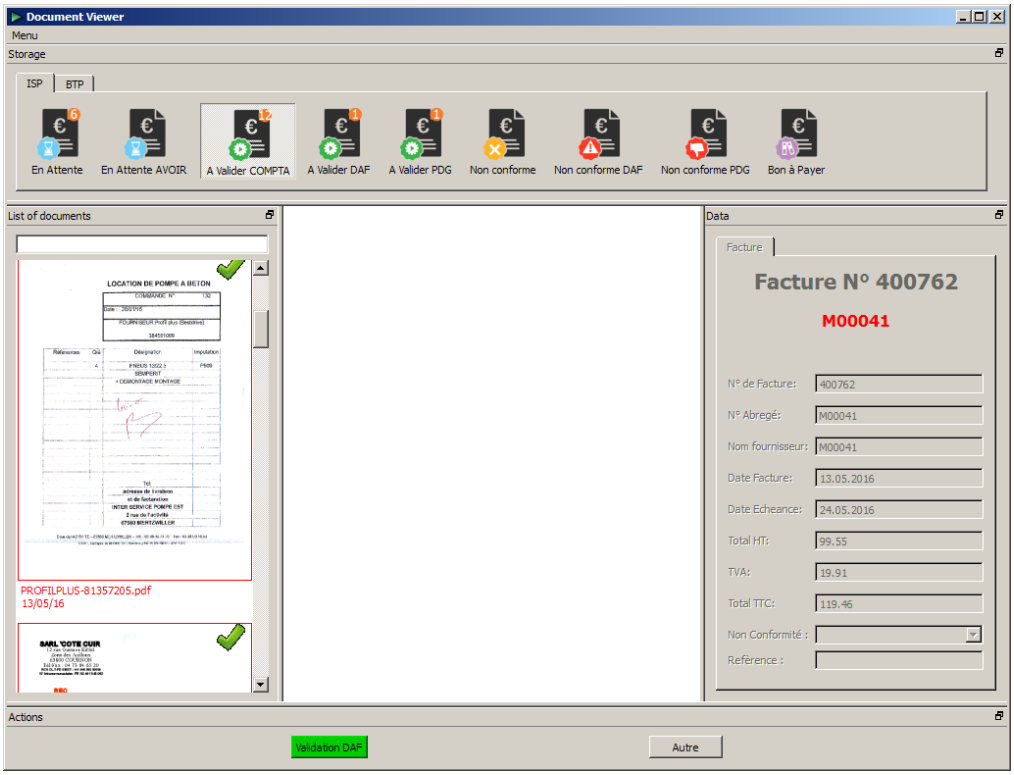

Рисунок П8. Окно приложения при работе с несколькими документами

Чтобы совершить действие с документом требуется нажать одну из кнопок действия в виджете действий (Рисунок П9).

Рисунок П9. Виджет действий с документои

Если проверка условий прошла успешно, то будет выведено окно с сообщением об успешной отправке данных на сервер ProShare (рисунок П10).

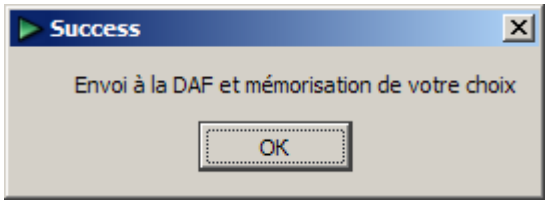

Рисунок П10. Пример сообщения при успешной отправке данных

При нарушения условий проверки будет выведено окошко со списком ошибок и комментариями к ним (рисунок П11).

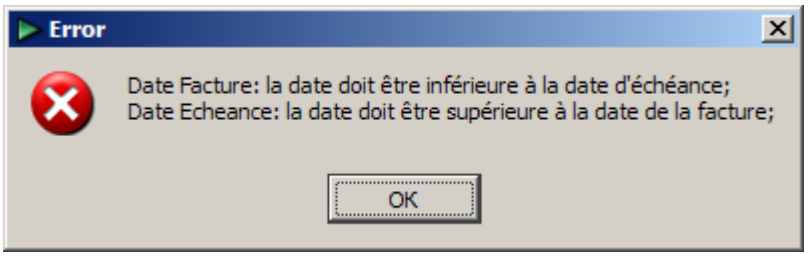

Рисунок П11. Пример сообщения со списком ошибок

Смена виртуального хранилища (пункт “Change storage”) или пользователя (пункт “LogOff”) возможна выбором соответствующего пункта в меню (рисунок П12).

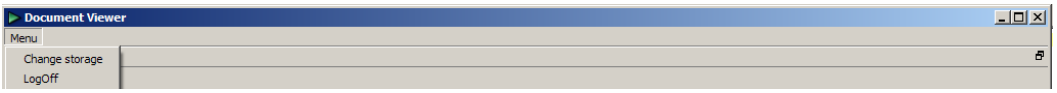

Рисунок П12. Меню приложения

Отчет о проверке № 1

дата выгрузки: 07.06.2016 12:05:37
пользователь: igzhukov963@yandex.ru / ID: 3254902
отчет предоставлен сервисом «Антиплагиат»
на сайте http://www.antiplagiat.ru

Информация о документе

№ документа: 2
Имя исходного файла: ot4etRework.docx
Размер текста: 2886 кБ
Тип документа: Прочее
Символов в тексте: 55924
Слов в тексте: 7167
Число предложений: 455

Информация об отчете

Дата: Отчет от 07.06.2016 12:05:37 - Последний готовый отчет
Комментарии: не указано
Оценка оригинальности: 93.63%
Заимствования: 6.37%
Цитирование: 0%

Оригинальность: 93.63%
Заимствования: 6.37%
Цитирование: 0%

Источники

Доля в тексте	Источник	Ссылка	Дата	Найдено в
1.77%	[1] Qt	http://ru.wikipedia.org	раньше 2011 года	Модуль поиска Интернет
1.77%	[2] Qt	http://dic.academic.ru	раньше 2011 года	Модуль поиска Интернет
1.77%	[3] Qt - кросс-платформенный инструментарий разработки ПО на языке программирования C++ от компании Qt Development Frameworks (ранее известна как Qt Software, Trolltech).	http://studopedia.net	14.11.2015	Модуль поиска Интернет

ООО «Вдом Ресерч»

"УТВЕРЖДАЮ"
Директор ООО «Вдом Ресерч»
Ерохин А. Е.
“ 8 ” июня 2016 г.

А К Т

о внедрении результатов
выпускной квалификационной работы
Жукова Игоря Андреевича

Комиссия в составе: председатель Ерохин А.Е, члены комиссии: Овсянников М.С., Козин С.С., Захаров Д.В. составили настоящий акт о том, что результаты дипломной работы “Разработка приложения для работы с документами в потоке работ для платформы Vdom desktop applications” использованы в проектах Proshare и Document Viewer.

Председатель комиссии

_____ Ерохин А. Е.

Члены комиссии:

_____ Овсянников М.С.

_____ Козин С.С.

_____ Захаров Д.В.