# ВВЕДЕНИЕ

Современные информационные технологии стремительно развиваются и играют большую роль в жизни общества и отдельного человека. Современный человек не может представить свою жизнь без компьютера и интернета. Эти изобретения предоставляют удобства для решения тех или иных вопросов и задач. Современный мир развивается в этом направлении и каждый год появляются новые сервисы и приложения, усовершенствуются старые.

Технологии обработки информации непрерывно совершенствуются, уделяется большое внимание безопасности доступа к данным, а вместе с ними меняются и практические методы обеспечения информационной безопасности. Любой сервис, взаимодействующий с данными, должен быть обеспечен должным уровнем безопасности и защиты, чтобы предотвратить различные нарушения. Такими нарушениями могут быть:

* нарушения конфиденциальности информации — получение широкого доступа к информации, предназначенной для узкого круга лиц;
* нарушения целостности информации — несанкционированное изменение, в результате которого информация не отражает реальное положение вещей или является внутренне противоречивой.

Универсальных методов защиты не существует, во многом успех при построении механизмов безопасности для реальной системы будет зависеть от её индивидуальных особенностей, учёт которых плохо подаётся формализации.

Основными угрозами для информационной безопасности являются перехват данных с целью использования сведений закрытого характера, модификация данных с целью фальсификации передаваемой информации, уничтожение данных для нарушения нормальной работы системы. Данные виды информационных угроз представляют особую опасность для информационной среды, так как их обнаружение является сложной технической задачей. Проблемы, вызванные нарушением информационной безопасности, могут повлечь за собой финансовые потери для физических лиц и организаций в целом, а также носить стратегический характер в борьбе между конкурирующими сторонами. В данной ситуации необходимо использовать комплекс мер, направленных на повышение уровня защиты информационных потоков и информационной среды от нежелательных факторов воздействия.

На данном этапе развития человечества различные сервисы по продаже товаров, билетов, перевозке людей, предоставления услуг и прочие достаточно сильно взаимодействуют с людьми. Однако в некоторых областях такое взаимодействие недостаточно налажено. Для этого необходимо внедрять системы электронного документооборота на предприятиях. Документооборот любой организации состоит из входящей и исходящей корреспонденции, отчетов, приказов и распоряжений. Хранение огромного количества документов доставляет много неудобств, поиск необходимого документа занимает много усилий и времени. Выход из сложившейся ситуации есть – и это система электронного документооборота (СЭД).

Система электронного документооборота позволяет организовать работу с электронными документами (создание, изменение, поиск), а также взаимодействие между сотрудниками (передачу документов, выдачу заданий, отправку уведомлений и т.п.).

Проблема внедрения СЭД возникает по нескольким причинам:

* консерватизм персонала, нежелание обучаться и переобучаться;
* боязнь прозрачности собственной деятельности для руководства,

которая возникает после внедрения системы электронного документооборота;

* нежелание непосредственно работать с компьютером, просматривать и редактировать документы;
* необходимость взаимодействовать с внешним «бумажным» миром, если это касается параллельных структур, с которыми идет постоянная работа.

Целью данного дипломного проекта является разработка СЭД, которая доступная и понятная для работников организации, обеспечивающую простоту взаимодействия между организациями, возможность электронной цифровой подписи для документов.

В соответствии с поставленной целью были определены следующие задачи:

* реализовать функционал создания новых пользователей и организаций;
* наделить каждого пользователя определенными правами;
* обеспечить должный уровень безопасности и доступа к ресурсам;
* реализовать функционал взаимодействия пользователей внутри и за пределами своей организации.

**1** ОБЗОР ЛИТЕРАТУРЫ

## 

## Аналитический обзор

## **1.1.1** Трехуровневая архитектура

В настоящее время трехуровневая архитектура является популярным и эффективным решением в разработке web-приложений. Трехуровневая архитектура (рисунок 1.1) является развитием классической клиент-серверной архитектуры. Трехуровневая архитектура имеет ряд достоинств по сравнению с клиент-серверной архитектурой [1]:

* разделение представления данных, логики приложения и хранилища данных позволяет с минимальными затратами вносить изменения в приложение;
* масштабируемость;
* конфигурируемость — архитектура позволяет быстро переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;
* высокая безопасность;
* высокая надёжность;
* низкие требования к скорости канала (сети) между клиентами и сервером приложений;
* низкие требования к производительности и техническим характеристикам клиентских приложений, как следствие снижение их стоимости.

Клиентским приложением может выступать не только компьютер, но и, например, мобильный телефон или планшет.

Тем не менее, можно выделить и следующие недостатки [1]:

* более высокая сложность создания приложений;
* сложнее в разворачивании и администрировании;
* высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;
* высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

## **1.1.2** Электронная цифровая подпись

Стоит отметить, в Республике Беларусь существует закон № 113-3 «Об электронном документе и электронной цифровой подписи», принятый 4 декабря 2009 года Палатой представителей, одобрен Советом Республики 11 декабря 2009 года, а 28 декабря подписан президентом Республики Беларусь А. Г. Лукашенко [2].



Рисунок 1.1 – Трехуровневая архитектура

Существует множество стандартов электронной цифровой подписи (далее - ЭЦП), удовлетворяющих определенным требованиям и условиям. Проектирование протоколов ЭЦП осуществляется двумя способами [3]:

* симметричные криптографические алгоритмы;
* ассиметричные криптографические алгоритмы.

Симметричные схемы электронной цифровой подписи основаны на использовании блочных шифров. Симметричные схемы имеют следующие преимущества [4]:

* стойкость симметричных схем электронных цифровых подписей зависит от стойкости блочных шифров. В мировой практике блочные шифры хорошо изучены, исследованы и протестированы;
* если блочный шифр является недостаточно стойким, его замена осуществляется с минимальными изменениями программного и аппаратного обеспечения, используемого в рамках схемы электронной цифровой подписи.

Симметричные схемы электронной цифровой подписи имеют следующие недостатки [4]:

* при выполнении процедуры подписи необходимо подписывать каждый бит исходного сообщения, что существенно увеличивает размер передаваемого сообщения;
* сгенерированные ключи могут быть использованы лишь один раз, так как после выполнения электронной цифровой подписи раскрывается часть секретного ключа, что значительно снижает уровень криптостойкости.

Наиболее известным протоколом симметричной электронной цифровой подписи является схема Диффи-Хелмана (Diffie-Hellman). Из-за приведенных выше недостатков схемы симметричной электронной цифровой подписи не получили широкого распространения в прикладной сфере.

Альтернативным подходом симметричным схемам является использование ассиметричных протоколов цифровой подписи. Основными составляющими являются следующие компоненты:

* пара ключей. Один из ключей является секретным, другой открытым, доступным широкому кругу пользователей;
* криптографическая хеш-функция;
* требования к генерации криптостойких параметров.

Основные действия, выполняемые в результате электронной цифровой подписи на основе ассиметричных алгоритмов:

* генерация пары ключей. Генерация секретного ключа, вычисление соответствующего ему открытого ключа;
* вычисление значения хеш-функции. Данное значение является уникальным для данного сообщения и обеспечивает контроль целостности передаваемой информации;
* формирование подписи. Использование результата хеш-функции и секретного значения для шифрования и передачи;
* проверка подписи. Данный процесс включает в себя вычисление значения хеш-функции на основе переданного незашифрованного сообщения, получение незашифрованного значения с использованием открытого ключа из переданной зашифрованной подписи, сравнение двух значений.

ЭЦП на базе ассиметричных алгоритмов основана на проблеме вычисления параметров подписи по имеющимся значениям. Также вычислительно сложной задачей является подделка подписи без знания секретного ключа. Выделяют основные вычислительные подходы, на которые опирается ассиметричная цифровая подпись [5]:

* проблема вычисления дискретного логарифма. Как правило, данная проблема рассматривается с использованием конечных полей;
* проблема факторизации. В основе данной проблемы лежит разложение чисел на простые множители.

В настоящее время не существует алгоритмов и методов, способных решать данные задачи за полиномиальное время. Следовательно, криптостойкость данного вида цифровой подписи основывается лишь на трудности вычисления значений в условиях существующего технического прогресса.

В качестве основы асимметричной криптографии используют следующие математические компоненты [5]:

* поля Галуа;
* эллиптические кривые.

Криптография с использованием полей Галуа основана на использовании конечного числа элементов. Так примером простейшего поля является кольцо вычетов по модулю простого числа.

Эллиптическая криптография основана на использовании различных классов и видов эллиптических кривых. Возможно использование эллиптических кривых над конечными полями. Основным преимуществом эллиптической криптографии является отсутствие субэкспоненциальных алгоритмов для решения задачи дискретного логарифма в группах точек эллиптических кривых, что ведет к значительному сокращению длин параметров. Сравнение длин параметров приведено в таблице 1.1 [6].

Алгоритмы ассиметричной электронной цифровой подписи подразделяются на обычные цифровые подписи, и цифровые подписи с восстановлением документа. При использовании цифровой подписи с восстановлением из содержимого подписи возможно восстановить исходный документ, что исключает необходимость его передачи. При использовании хеш-функций для подписи необходимо кроме подписанного значения передавать сам документ.

Таблица 1.1 – Сравнение параметров

|  |  |  |
| --- | --- | --- |
| Симметричный ключ, бит | Размер простого числа p, бит | Размер эллиптической группы, бит |
| 80 | 1024 | 160 |
| 128 | 3072 | 256 |
| 256 | 15360 | 512 |

Алгоритмы асимметричной электронной цифровой подписи могут быть вероятностными и детерминированными. При использовании вероятностной подписи результатом подписи для одинаковых сообщений будут различные результаты. При использовании детерминированных схем результат для одинаковых сообщений после выполнения подписи будет одинаковым.

Существует ряд стандартов, описывающих ассиметричные цифровые подписи:

* **стандарты открытого ключа (PKCS - public key cryptographic standards) – набор стандартов, разработанных RSA Laboratories;**
* **схема Эль-Гамаля (ElGamal). Является одним из вариантов алгоритма Деффи-Хеллмана. Данный алгоритм не был запатентован, в результате чего является дешевой альтернативой, так как нет необходимости платить за лицензию;**
* **вероятностная схема подписи Рабина (Rabin). Использование вероятностных алгоритмов подписи;**
* **схема БЛС** (BLS - Boneh-Lynn-Shacham)**. Является схемой быстрой электронной цифровой подписи;**
* российские стандарты ЭЦП: ГОСТ Р 34.10-94, ГОСТ Р 34.10-2001;
* белорусский стандарт ЭЦП: СТБ 1176.2-99;
* **американский стандарт ЭЦП: DSA (Digital Signature Algorithm);**
* **американский стандарт ЭЦП: ECDSA (Elliptic curve digital signature algorithm). Данный стандарт является модификацией DSA с использованием эллиптических кривых.**

**Основными преимуществами ЭЦП на базе ассиметричных алгоритмов являются:**

* **в отличие от симметричной криптографии нет проблемы передачи ключа по открытому каналу;**
* **в ассиметричной криптографии секретным является лишь один ключ, в то время как в симметричной несколько;**
* **использование одного и того же ключа несколько раз значительно не снижает криптостойкости цифровой подписи. В симметричных алгоритмах необходимо каждый раз использовать новый ключ;**
* **в ассиметричных системах распределения ключей задействовано меньше ключей.**

**Основными недостатками электронной цифровой подписи на базе ассиметричных алгоритмов являются:**

* **изменение алгоритма вносит серьезные коррективы в использование криптосистемы;**
* **использование больших ключей в сравнении с симметричными алгоритмами;**
* **процесс шифрования, дешифрования занимает больше времени, чем аналогичные процессы в симметричных алгоритмах.**

## **1.2** Обзор существующих аналогов

Документооборот является важной составной частью делопроизводства и информационного обеспечения управления. Электронный документооборот – движение документов в организации с момента их создания, получения до завершения исполнения или отправления в электронно-цифровой форме [7].

Одной из популярных СЭД является система «Directum» (рисунок 1.2). Первая версия данного продукта появилась в декабре 1999 года. Система разработана в России в городе Ижевске и предназначена для использования под операционной системой Windows [8].

Directum — система управления корпоративным контентом (Enterprise Content Management), на базе возможностей которой строится полноценная система электронного документооборота и инфраструктура эффективного взаимодействия сотрудников предприятия от уровня топ-менеджмента до конечных исполнителей [9].

Из преимуществ системы «Directum» можно выделить:

* удобные напоминания;
* наличие предпросмотра вложенных файлов, файловых хранилищ;
* большой выбор заполненных справочников в базовой конфигурации, интерактивные мастера создания документов.
* наличие полнотекстового поиска
* удобные функции настройки шаблонов поисков.
* возможность создавать поисковые папки.

К основным недостаткам системы относятся:

* отсутствие постраничного вывода объектов;
* нельзя распределить права доступа к настроенным поискам;
* нет настроенных поисков в веб-клиенте;
* нет возможности создавать сложные поисковые запросы;
* перегруженный интерфейс.

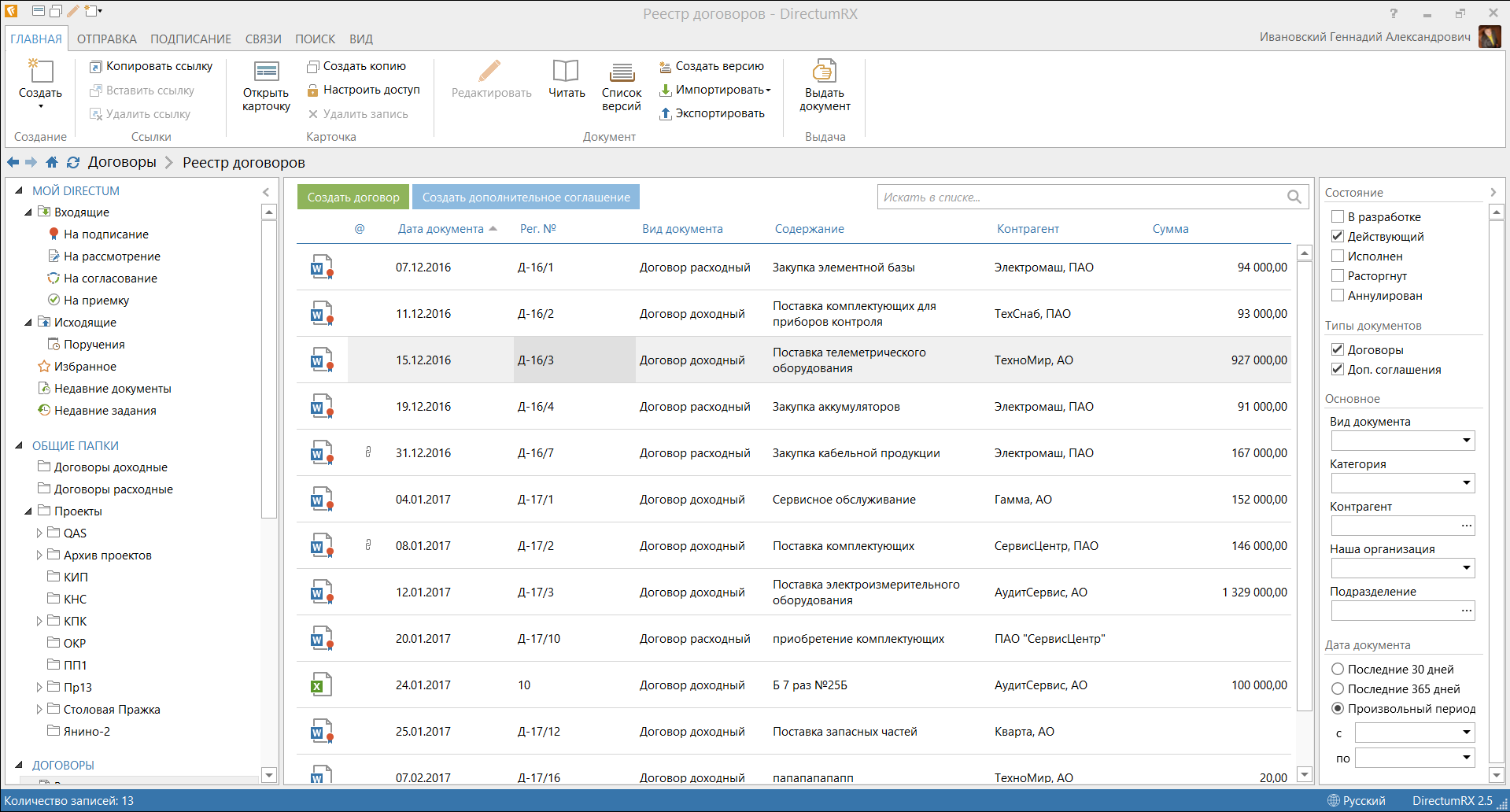


Рисунок 1.2 – Приложение для рабочего стола системы «Directum»

Рассмотрим еще один аналог СЭД «Optima WorkFlow» (рисунок 1.3). Optima WorkFlow — программная платформа для создания систем управления документами (электронного документооборота) в государственных и коммерческих организациях любого масштаба [10].

Данный продукт разработала российская компания Optima software (ООО «Документ Менеджмент»). Первая версия появилась в 1997 году. Optima WorkFlow обеспечивает комплексную автоматизацию процессов обработки документов и позволяет перейти к безбумажной технологии работы с электронными документами.

Платформа Optima WorkFlow — это открытая, web-ориентированная архитектура, функциональность, применение самых современных технологий и промышленных стандартов, интеграция с лидирующими ИТ-решениями, визуальные средства настройки и адаптации системы.

К преимуществам системы относятся:

* возможность применения дополнительных стилей в описи документов;
* вывод изображений в опись документов и область просмотра;
* возможность фильтрации загруженной описи документов;
* настройка области просмотра выбранной записи в описи документов;
* понятный интерфейс;
* удобные поиски в полях-списках регистрационных карточек;
* удобно работать с прикрепленными файлами.

Из недостатков можно выделить:

* отсутствует возможность создавать пользовательские папки.
* пользователи не могут создавать новые журналы, подборки документов, настройки не сохраняются в базу данных;
* непрозрачность процесса при создании задания, для исполнителей нельзя настроить напоминания.

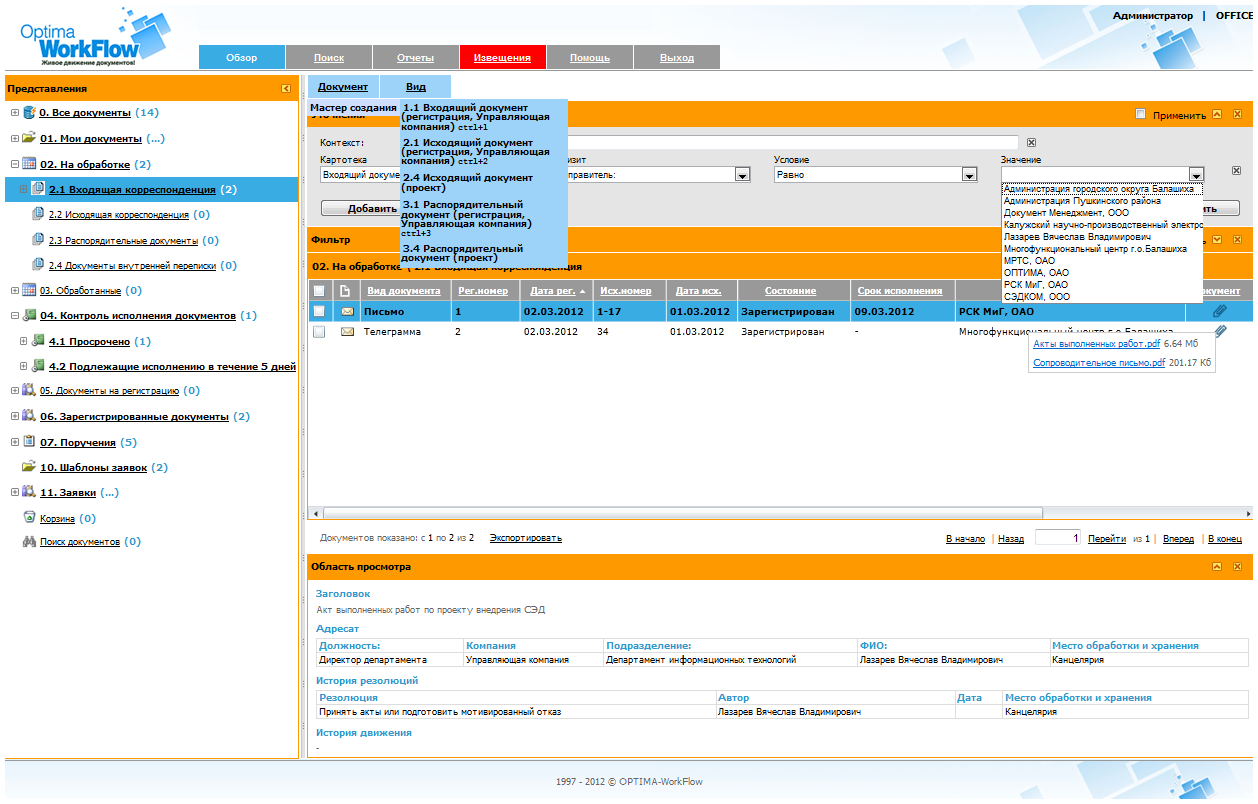


Рисунок 1.3 – Веб-приложение «Optima Workflow»

В 1998 году в компании Digital Design группа разработчиков под руководством Владимира Андреева создала OfficeClerk — первый прототип системы, построенный на основе Microsoft Exchange. Через год принято решение о создании собственной системы Docsvision (рисунок 1.4) на базе SQL Server и платформы Microsoft .Net [11]. Заключен договор ISV-партнерства с Microsoft, подтверждающий высокое качество системы Docsvision.

Компания «ДоксВижн», создатель и разработчик системы управления документами и бизнес-процессами предприятий и организаций Docsvision, является одним из лидеров рынка СЭД в России, СНГ и странах Балтии.

К преимуществам данной системы можно отнести:

* возможен полнотекстовый поиск;
* присутствуют настраиваемые поисковые шаблоны;
* реализован механизм drag&drop;
* можно применять преднастроенные темы;
* проектировка карточек задания;
* поддержка создания сложных поисковых запросов с помощью XML;
* поддержка распределения прав доступа к настроенным поискам.

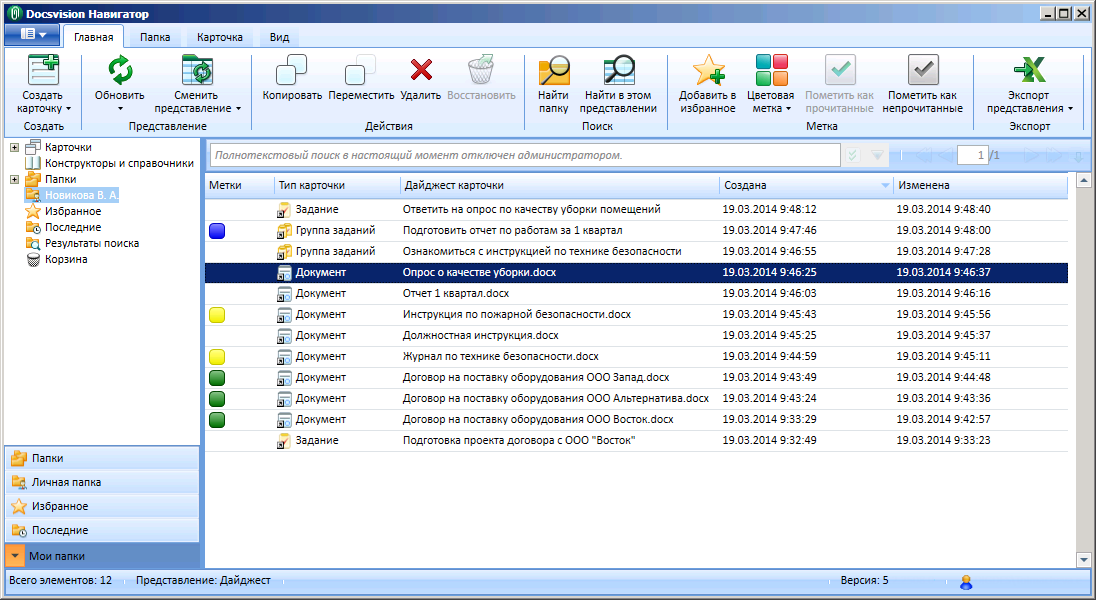


Рисунок 1.4 – Приложение для рабочего стола системы «DocsVision»

При этом можно выделить следующие недостатки:

* напоминания функционируют только при включенном MS Outlook;
* медленная работа панели просмотра.

## 

## **2** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

**2.1** Постановка задачи

Проанализировав достоинства и недостатки аналогов, следует выделить следующие задачи для реализации в данной работе, которые позволят конкурировать с существующими аналогами:

* реализовать возможность добавления, редактирования и удаления организаций, пользователей;
* реализовать функционал взаимодействия пользователей внутри и за пределами своей организации;
* создать простой и понятный пользовательский интерфейс;
* реализовать функционал разграничения пользователей, чтобы разделить их права и возможности;
* реализовать возможность загрузки документов;
* реализовать постраничный вывод документов;
* предоставить пользователю возможность запрашивать документ на подпись;
* предоставить пользователю возможность подписывать документы;
* реализовать полнотекстовый поиск в документах;
* финансовые затраты на разработку и сопровождения должны давать возможность предоставить более дешевую альтернативу существующим аналогам.

Разрабатываемое программное средство должно иметь ряд функциональных возможностей, являющихся преимуществами перед существующими аналогами:

* система должна поддерживать механизм версионирования документов;
* система должна использовать собственный алгоритм создания электронной цифровой подписи.

**2.2** Описание структурной схемы

Поставив задачу, разбиваем систему на функциональные блоки(модули). Это необходимо для обеспечения гибкой архитектуры. Такой подход позволяет изменять или заменять модули без изменения всей системы в целом и обеспечивает слабую связанность.

В разрабатываемом веб-приложении можно выделить следующие блоки:

* блок пользовательского интерфейса;
* блок представления данных;
* блок реляционной базы данных;
* блок работы с базой данных;
* блок обработки данных;
* блок аудита событий;
* блок поиска данных;
* блок цифровой подписи документа;
* блок отправки электронных писем.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.064 C1.

Каждый модуль имеет определенную ответственность. Каждый модуль взаимодействует с другими должным образом для обеспечения корректной работы и достижения эффективной работы системы.

Рассмотрим функциональные блоки веб-приложения.

*Блок пользовательского интерфейса* является клиентской частью веб-приложения. Данный блок необходим для визуализации данных на странице веб-браузера, полученных от блока представления данных, а также обработки всех действий пользователя с последующей отправкой обработанных данных блоку представления данных. Блок пользовательского интерфейса должен содержать набор определенных html страниц, которые предоставляют возможность вводить и отображать определенную информацию.

*Блок представления данных* является фактически верхним уровнем в трехуровневой архитектуре. Он необходим для получения данных от блока пользовательского интерфейса, валидации этих данных и последующей передаче блоку обработки данных, а также для передачи данных пользователю для отображения в веб-браузере. Данный блок должен содержать какие-то сущности, которые могут принимать необходимые http-запросы типа GET, POST, PUT, DELETE и отдавать данные по http протоколу, а также определять насколько то или иное действие допустимо для каждого пользователя в каждом конкретном случае.

*Блок реляционной базы данных* необходим непосредственно для хранения данных, используемых системой и пользователями. Чтобы хранить данные необходимые данной системе, целесообразно использовать MSSQL базу данных и Microsoft SQL Server. В SQL Server 2005 встроена поддержка .NET Framework. Благодаря этому, хранимые процедуры БД могут быть написаны на любом языке платформы .NET, используя полный набор библиотек, доступных для .NET Framework, включая Common Type System (система обращения с типами данных в Microsoft .NET Framework). Однако, в отличие от других процессов, .NET Framework, будучи базисной системой для SQL Server 2005, выделяет дополнительную память и выстраивает средства управления SQL Server вместо того, чтобы использовать встроенные средства Windows. Это повышает производительность в сравнении с общими алгоритмами Windows, так как алгоритмы распределения ресурсов специально настроены для использования в структурах SQL Server.

*Блок работы с базой данных* необходим для взаимодействия с базой данных. Он должен уметь работать с базой данных таким образом, чтобы получать и записывать в таблицы базы необходимые данные для должного функционирования системы. Данный блок взаимодействует с блоком обработки данных и блоком поиска данных. Он передает им необходимые данные в ответ на обращение к нему.

*Блок обработки данных* является основным блоком в разрабатываемой системе и отвечает за обработку данных пользователей, компаний и документов. Он принимает данные от блока представления данных, преобразует их при необходимости и передает другим блокам. Данный блок взаимодействует с блоком работы с базой данных и может использовать функционал модулей поиска, цифровой подписи документа, отправки электронных писем, аудита событий при необходимости в зависимости от действия пользователя.

*Блок аудита событий* служит для записи в отдельную таблицу в базе данных важной информации о событиях, которые происходят в системе. Такими событиями могут быть:

* возникновение исключительной ситуации в системе, когда код программы не выполняет или не может выполнить какое-либо действие по определенным причинам;
* изменение состояния сущности системы, например, редактирование неких сущностей или изменение уровня доступа к ним.

*Блок поиска данных* служит для поиска необходимых доступных данных в ответ на запрос пользователя. Данный блок непосредственно взаимодействует с базой данных для нахождения данных в таблицах и далее передает данные блоку обработке данных.

*Блок цифровой подписи документа* необходим для генерации и проверки цифровой подписи.

*Блок отправки электронных писем* служит для отправки писем определенным пользователям с целью их уведомления. Необходимые данные приходят от модуля обработки данных, который в свою очередь получает информацию от модуля аудита событий.

## **2.3** Обоснование выбора языка программирования

Любое разрабатываемое программное средство должно удовлетворять ряду обязательных требований, таких как переносимость, расширяемость, надежность. Для обеспечения таких характеристик немаловажным является выбор языка и платформы программирования, на котором оно реализуется.

Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов [12].

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз.

Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на GitHub.

ASP.NET Core построен на основе кросс-платформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS X, Linux. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой, то есть мы можем запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS, а для развертывания веб-приложения можно использовать традиционный IIS, либо кросс-платформенный веб-сервер Kestrel.

Хотя ASP.NET Core преимущественно нацелено на использование .NET Core, но фреймворк также может работать и с полной версией фреймворка .NET.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC. А Web Forms полностью ушли в прошлое.

Кроме объединения вышеупомянутых технологий в одну модель в MVC добавлен ряд дополнительных функций.

Одной из таких функций являются тэг-хелперы (tag helper), которые позволяют более органично соединять синтаксис html с кодом С#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, такие как Autofac, Ninject. Хотя при желании их также можно продолжать использовать.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, мы можем создавать приложения в среде Visual Studio Code, которая является кросс-платформенной и может работать как на Windows, так и на Mac OS X и Linux.

При разработке в Visual Studio 2015 проекты приложений имеют встроенную поддержку с такими популярными инструментами, как Bower, Grunt, Gulp, который позволяют управлять скриптами JavaScript и стилями CSS, автоматизировать и оптимизировать процесс веб-разработки.

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты.

Если суммировать, то можно выделить следующие ключевые отличия ASP.NET Core от предыдущих версий ASP.NET:

* новый легковесный и модульный конвейер http-запросов;
* возможность развертывать приложение как на IIS, так и в рамках своего собственного процесса;
* использование платформы .NET Core и ее функциональности;
* распространение пакетов платформы через NuGet;
* интегрированная поддержка для создания и использования пакетов NuGet;
* eдиный стек веб-разработки, сочетающий Web UI и Web API;
* конфигурация для упрощенного использования в облаке;
* встроенная поддержка для внедрения зависимостей;
* расширяемость;
* кроссплатформенность: возможность разработки и развертывания приложений ASP.NET на Windows, Mac, Linux;
* развитие как open source, открытость к изменениям.

Однако в заключение стоит отметить, что та прежняя модель, которая использовалась в ASP.NET 4.5 - MVC 5, Web API 2 - все это остается и даже продолжает развиваться в рамках .NET 4.6.