**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«КАЛУЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**им. К. Э. ЦИОЛКОВСКОГО»**

Физико-технологический институт

Кафедра информатики и информационных технологий

Направление подготовки 09.03.02 Информационные системы и технологии

*ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА*

по дисциплине «Проектирование информационных систем»

на тему: Разработка генератора электронного документооборота «EDOCript»

студента 4 курса очной формы обучения

**Андреева Павла Олеговича**

научный руководитель

Виноградская М.Ю.

**г. Калуга, 2020 г.**

Оглавление

[Введение 3](#_Toc40118166)

[Глава 1. Основы проектирования и разработки приложения 5](#_Toc40118167)

[1.1 Основные понятия 5](#_Toc40118168)

[1.2 Назначения и область применения разработки 8](#_Toc40118169)

[1.3 Анализ существующих разработок 11](#_Toc40118170)

[Глава 2. Проектирование генератора электронного документооборота «EDOCript» 11](#_Toc40118171)

[2.1 Выбор методов и средств проектирования приложения, установление требований к проектируемому приложению 12](#_Toc40118172)

[2.2 Проектирование модели и архитектуры построения приложения 18](#_Toc40118173)

[2.3 Проектирование модели данных и базы данных приложения 21](#_Toc40118174)

[2.4 Проектирование пользовательского интерфейса 26](#_Toc40118175)

[Глава 3. Программная реализация генератора электронного документооборота «EDOCript» 29](#_Toc40118176)

[3.1 Обоснование выбора состава программных средств реализации 29](#_Toc40118177)

[3.2 Описание пользовательского интерфейса и инструкций пользователю 34](#_Toc40118178)

[3.3 Руководство разработчика 46](#_Toc40118179)

[Заключение 50](#_Toc40118180)

[Список использованных источников 51](#_Toc40118181)

[ПРИЛОЖЕНИЕ А 52](#_Toc40118182)

# Введение

При современной организации производства системы электронного документооборота становятся обязательным элементом информационно-технологической инфраструктуры. С их помощью повышается уровень эффективности деятельности коммерческих компаний и промышленных предприятий, а в госучреждениях на базе технологического электронного документооборота происходит решение задач внутреннего управления, межведомственного взаимодействия и взаимодействия с населением.

Изначально системы подобного класса рассматривались лишь как инструмент, предназначенный для автоматизации задач делопроизводства классического характера, но, улучшаясь вместе с ростом технологий, начали охватывать все более широкий спектр задач. На текущий момент разработчики систем электронного документооборота проводят ориентизацию своих продуктов на возможность работы не только с различного типа корреспонденцией и организационно-распорядительными документами, но и с различными внутренними документами (договорами, нормативной, справочной и проектной документацией, документами по кадровой деятельности и др.). Системы электронного документооборота также получили функционал для решения задач прикладного характера, в которых важной составляющей является проведение работ с электронными документами: управление взаимодействием с клиентами организации, обработка обращений и жалоб граждан, организация автоматического оповещения сервисов/работы сервисной службы, организация проектного документооборота и др. Фактически, системой электронного документооборота называют любую информационную систему, обеспечивающую взаимодействие с электронными документами.

Так как современные компании, имеющие в использовании подобные системы, имеют очень большой поток информации, разработка систем необходимого уровня функционала становится все сложнее и сложнее. В связи с этим появилась необходимость для продвинутой системы тестов, используемых для выявления ошибок в работе ИС.

Первые программные системы разрабатывались в рамках исследовательских программ или программ для нужд министерств обороны. Тестирование подобных продуктов проводилось с записью всех тестовых процедур, получаемых данных, полученных результатов. Тестирование выделялось в отдельный процесс работы, который начинался после завершения кодирования программы, но при этом, как правило, выполнялся тем же самым персоналом.

Из года в год объемы данных, проходящих сквозь информационные системы, растут, так же, как и необходимость разработки тестировочных систем для них. В связи с масштабами информационных систем и сложности их реализации пропорционально усложняются системы тестирования.

Для упрощения процесса тестирования и хранения получаемых результатов требуется организация кейсов с данными различных тестов. Порой количество и наполнение подобных кейсов имеет весьма сложную структуру для того, чтобы составлять их руками человека.

Основными сложностями составления кейсов является: строгость формирования документа, который должен соответствовать всем стандартам и ГОСТам, нормализующим оформление документации, а также наличие грамотно сформированной электронной подписи, подделка которой довольно сложно, так как одним из важнейших направлений развития российского законодательства и правоприменительной практики в нынешнее время является регулирование отношений правового характера в области электронного документооборота и придания юридической силы электронным документам.

Принятие ФЗ «Об электронной цифровой подписи» узаконило использование электронной цифровой подписи как аналога собственноручно. Электронная подпись является необходимым атрибутом электронного документа, использующимся в документации в качестве универсального способа подтверждения факта изъявления воли делового лица. Подпись являет собой результат криптографического преобразования информации в виде последовательности цифр и является средством защиты информации, обеспечивающим возможность контроля целостности документации, также подтверждая ее подлинность. Чтобы документ, «хранимый, обрабатываемый и передаваемый с помощью автоматизированных информационных и телекоммуникационных систем», мог приобрести юридическую силу, необходимо удостоверить его электронной подписью.

Поэтому создаются электронные подписи в специальном программном обеспечении, разработанном исключительно для создания подлинных электронных подписей.

Для автоматизации процесса составления таких кейсов можно использовать различные системы генерации данных, тем самым значительно упрощая работу тестировщика систем.

Целью дипломной работы является автоматизация деятельности тестировщика в виде реализации генератора пакетов данных эмулирующий процесс обмена электронными документами, с использованием электронной подписи, в системе электронного документооборота. (задачи не придумал)

# Глава 1. Основы проектирования и разработки приложения (хз, о чем она)

# Основные понятия

Электронная цифровая подпись (ЭЦП) - (англ. digital signature) - последовательность символов, полученная в результате криптографического преобразования исходной информации, которая позволяет подтверждать целостность и неизменность этой информации, а также ее авторство.

В настоящий момент цифровые подписи используются в основном для проведения процесса аутентификации автора документа и для проверки того факта, что оснащенные подписью сообщение или данные не подверглись модификации во время того, как информация передавалась в компьютерных сетях.

Механизм электронной цифровой подписи (ЭЦП) возник как побочный эффект криптографии с открытым ключом. Поэтому, характерное для систем с открытым ключом разделение ключа на 2 части - секретную и несекретную, позволяет реализовать возможность проверки подлинности без возможности подписать другой документ.

XML — расширяемый язык разметки. XML описывает поддерживаемые документы и частично проводит описание поведения XML-процессоров. Данный язык разметки разрабатывался как язык с высоким уровнем синтаксической формальности, удобный для проведения обработки документации программами, одновременно имеющий достаточный уровень удобства для чтения и создания документов пользователями. XML создавался с нацеленностью на использование в Интернете.

Расширение языка является конкретной созданной на базе XML грамматикой, представленной словарём тегов и их атрибутов, а также набором правил определения, какие атрибуты и элементы могут входить в состав других атрибутов и элементов.

XPath— язык запросов к элементам XML-документа. Разработан для организации доступа к частям XML-документа в трансформационных файлах. Является принятым стандартом консорциума W3C. Язык призван реализовать навигацию по дереву тегов XML. В данном языке используется компактный набор символики, отличный от принятого в коренном XML.

XSLT— язык преобразования XML-документов. Спецификация XSLT входит в состав XSL и является рекомендацией Консорциума всемирной паутины.

При применении таблицы стилей XSLT к XML-документу образуется конечное дерево, которое может быть пересобрано в XML-документ, XHTML-документ, HTML-документа или текстовый файл. Правила выбора и преобразования данных из исходного дерева пишутся на языке запросов XPath.

Python — высокоуровневый язык программирования, ориентированный на повышение производительности разработчика и улучшение читаемости кода. Синтаксис ядра Python минималистичен, но в то же время стандартная библиотека включает в себя большой объём полезного функционала.

Поддерживается разбиение программ на рабочие модули, которые, в свою очередь, могут объединяться в пакеты данных.

Flask — фреймворк для создания веб-приложений на Python, облегчающий работу над бэкенд-кодом. Flask минималистичен, прост в использовании и гибок в реализации. Также у него отсутствуют какие-либо ограничения.

Jinja — это текстовый шаблонизатор для языка программирования Python. Данный программный инструмент может быть использован для создания любого вида исходного кода и разметки данных. Также шаблонизатор обеспечивает грамотное исполнение шаблонов в песочнице.

Шаблонизатор Jinja также добавляет настраиваемый функционал тегов, тестов, фильтров и глобал-переменных Также, в отличие от Django, Jinja позволяет конструктору шаблонов объектный вызов функций вместе с их аргументами.

SQLite — компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду Google-O’Reilly Open Source Awards.

JavaScript — мультипарадигменный язык программирования. Сам js — создан для фронтенда и его код обрабатывается непосредственно браузером, т.е это язык сценариев на стороне клиента, но из-за повсеместной популярности использования из него была реализована бекенд - версия. JavaScript обычно используется как встраиваемый язык для доступа к объектам разрабатываемых приложений. Для данного создано большое количество заточенного под различные задачи программного обеспечения, облегчающего процесс создания программ.

jQuery — набор функций JavaScript, функционал которого фокусируется на взаимодействии языков программирования JavaScript и HTML. Библиотека jQuery облегчает получение доступа к любому элементу объектной модели, обращение к атрибутам и содержимому элементов объектной модели, взаимодействие манипуляции с ними. Также библиотека jQuery предоставляет удобный программный интерфейс для работы с веб-технологией динамической прогрузки отдельных элементов сайта.

# Назначения и область применения разработки

В современных реалиях тестировочные системы для крупных приложения, является жизненной необходимостью в процессе разработки и поддержки приложения. Потому грамотное составление кейсов для тестов имеет большую важность в работе тестировщика. Разрабатываемый генератор пакетов предназначается как раз, чтобы снизить элемент человеческого фактора и самое важное ускорить работу тестировщика системы, тем самым положительно влиять на процесс разработки проекта.

Разрабатываемый программный продукт будет использован в целях генерации пакетов, представляющих собой zip-архив, сформированный по стандартам системы документооборота. Сгенерированные пакеты данных будут использованы в качестве электронной подписи, обеспечивая возможность проверки целостности документов во время EDI.

EDI - электронный обмен данными, отправка и получение информации с помощью web-технологий. Благодаря практической пользе и удобству стандарта EDI для использования в бизнес-переписке и передаче информации, данный стандарт прочно занял нишу в различных экономических отраслях и социальном обслуживании. Любая стандартно обработанная деловая документация, практически постоянно отправляемая в ходе ведения деловых отношений между различными компаниями (такая как заказы на поставку продукции, фактурный счёт, план отгрузок, запросы о наличии товара) может быть подвержена передаче при помощи электронного стандартов EDI, при условии, что обе стороны провели необходимую для этого подготовку. Наряду с EDI существуют и другие стандарты, используемые для безопасного обмена данными. К примеру, EDIFACT имеет широкое распространение Европе, а также в автомобильной промышленности. HIPPA разработан специально для соответствия законодательству деятельности здравоохранительных учреждений. Транслятор EDI, который использует компания, должен поддерживать все стандартные способы шифрации для возможности грамотного обмена данными с компаниями другой направленности либо находящимися в другой территориальной зоне.

Также желательно выполнение стандартов CMS для подписанных сообщений. Стандарт синтаксиса производит описание структуры сообщений, включающих в себя защищенные данные, вместе с этим храня в себе сведения, необходимые для их открытия либо использования. К примеру, в сообщении могут находиться защищенные данные, информация об алгоритме хеширования и подписи, времени установки электронной подписи, сертификации открытого ключа, и т.д. Не все названные атрибуты, обязательно должны находиться в подписи, но приложение может само определить, необходимо ли их наличие.

RSA Laboratories в серии своих стандартов криптографии с открытом ключом (PKCS) предложила решение проблемы алгоритмизации передачи множества документов и автоматической организации правильно построенной электронной подписи путем определения синтаксиса для защищенных сообщений в следующих стандартах: PKCS - 7, PKCS - 10.

Развитием и логическим продолжением данных стандартов, сохранившим и логически развившим функционал предыдущих версий, стал стандарт CMS. CMS, кроме определенной заголовком статьи электронной подписи, имеет возможность поддержки дополнительных операций шифрования, процесса вычисления хеша и имитационной вставки, в том числе и по российским алгоритмам шифрации данных, а также множественную инкапсуляцию данных. Последнее означает, что сообщение формата CMS может содержать в себе другие защищенные и отдельно зашифрованные данные.

Всего CMS имеет поддержку шести типов данных:

* просто данные,
* данные с электронной подписью,
* упакованные данные,
* хешированные данные,
* зашифрованные данные,
* данные с проверкой подлинности.

Электронный документооборот, как правило происходит в передаче документов, описанных на расширяемом языке разметки XML. Для формирования шаблонов таких документов используется XSD.

XSD — это язык описания структуры XML документа. Его также называют XML Schema. При использовании XML Schema XML парсер может проверить не только правильность синтаксиса XML документа, но также его структуру, модель содержания и типы данных.

Посмотреть XSD шаблоны для документов участвующих в разработке генератора можно на сайте Ассоциации «РОСЭУ».

Документооборот может осуществляться между тремя видами участников:

* Абонент – налогоплательщик (юридическое лицо или индивидуальный предприниматель) или его уполномоченный представитель
* Налоговый орган – Налоговый орган ФНС России
* Спецоператор – оператор электронного документооборота

У каждого участника документооборота должен быть GUID (Globally Unique Identifier), ИНН (Идентификационный номер налогоплательщика), а также если он является организацией, то ему должен быть присвоен КПП (Код постановки на учет).

# Анализ существующих разработок

Кря-кря, тут был долбоклюй, а текста не было

# Глава 2. Проектирование генератора электронного документооборота «EDOCript»

# 

# 2.1 Выбор методов и средств проектирования приложения, установление требований к проектируемому приложению

Проектирование приложения представляет сложный вид деятельности, разделенный на множество ступеней работы. Научная организация процесса проектирования дает возможность создавать и использовать современные сложные приложения. Для того, чтобы иметь возможность создавать подобные работы, функционал которых может охватывать множество различных видов деятельности, проектировщику приложения нужны как теоретические, так и практические знания в обрабатываемой области.

Проектирование приложения, в отличие от моделирования, может происходить в момент, когда создаваемый объект существует еще только в теории.

Проектирование приложения в основном проводится в трех областях программирования, а именно:

* Проектирование объектов данных, реализуемых в БД;
* Проектирование программ, которые позже будут реализовывать обеспечение выполнения запросов к имеющимся данным;
* Учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой файл-серверной либо клиент-серверной архитектуры, параллельной обработки данных и т.п.

Начальный этап процесса проектирования – определение цели проекта приложения. Цель проекта можно так же представить в виде необходимости решения ряда задач, определяемых направление разработки:

* Требуемого количества необходимых функций, а также уровня адаптационной способности системы;
* Требуемой пропускной способности системы;
* Требуемой реакционной способности системы;
* Безотказной работы системы;
* Необходимого уровня защиты используемых данных;
* Простоты использования и технической поддержки реализуемой системы.

При разработке приложения для удобства и грамотного распределения ресурсов разработки обычно используются различные модели данных, по схеме которых и происходит дальнейшая работа. Существует несколько наиболее распространённых моделей: каскадная, инкрементная и спиральная.

Каскадная модель данных подходит для систем, требования и цели разработки которых на достаточно полном уровне известны на начальном этапе, что позволяет рассчитывать на разработку без изменения плана работы над проектом во время рабочего процесса.

Достоинства модели данных:

* На каждой стадии разработки приложения создается полный необходимый список логов, получаемых данных и документации, необходимого/использованного ПО;
* Четко спланированные заранее, последовательно выполняемые стадии разработки дают возможность заранее спланировать четкие сроки выполнения проекта и необходимые на него различного характера ресурсы.

Недостатки модели данных:

* Реальный процесс разработки нечасто укладывается в четко реализуемую схему. Для возможности использования при разработке данной стратегии система должна быть полностью прогнозируема заранее, что в реальности бывает редко. Также никто не гарантирует, что в процессе работы заказчик не изменит требования к будущему проекту;
* Главным недостатком данной модели является то, что заказчик получает какие-либо результаты только после полной реализации приложения, прошедшего через все этапы разработки.

Если во время процесса разработки, который может занимать достаточно долгое время, требования заказчика изменились или в самом начале были изложены неполно/неточно, на выходе заказчик приложения получает не подходящую для него систему.

Инкрементная стратегия характеризуется линейной разработкой продукта, но при этом в нескольких версиях, каждая из которых будет учитывать предыдущие, все более улучшаясь.

В начале работы над проектом определяются все основные требования к системе, после чего последовательно разрабатываются несколько версий программы. Каждая версия, при этом, может являться законченным, подходящим для реализации и обладающим набором необходимого функционала приложением. Первая версия чаще всего реализует лишь малую часть необходимого функционала, во второй версии количество функций увеличивается, а уже имеющие – шлифуются и дорабатываются, и так – до реализации законченного проекта, обладающего полным функционалом.

Чаще всего инкрементная модель используется в случаях, когда и разработчик, и заказчик четко представляют, что должен являть собой продукт разработки и какими функциями он должен обладать. Может быть несколько причин для использования подобной стратегии:

* Отсутствие у заказчика финансовой возможности сразу обеспечить ресурсами весь проект;
* Отсутствие у разработчика необходимых ресурсов для реализации полного проекта в сжатые сроки;
* Продукт требует поэтапного введения и постепенного освоения своего функционала конечными пользователями.

Внедрение системы сразу, с полным функционалом, может вызвать неприятие конечных пользователей, сильно замедляя процесс адаптации к новым технологическим решениям, а постепенное ее внедрение позволяет привыкнуть к новым функциям, улучшая привыкание посредством обучения за длительное время.

Достоинства и недостатки этой стратегии такие же, как и у классической. Но при этом заказчик, при разработке по инкрементной стратегии, может увидеть результаты разработки гораздо раньше. Уже по результатам, полученным при реализации первой версии продукта, заказчик может изменить список требований, отказаться от реализации дополнительных версий или предложить разработать еще более совершенный функциональный продукт, заключив с разработчиками новый договор.

Спиральная стратегия подразумевает разработку последовательных версий продукта при условии, что в начале работы над проектом определены не все итоговые требования к результату. Требования уточняются в результате разработки версий.

Данная модель жизненного цикла чаще всего используется при разработке новаторских систем. Чаще всего спиральная стратегия используется в случаях, когда ни разработчик, ни заказчик четко не представляют, что должен являть собой продукт разработки и какими функциями он должен обладать, а также при наличии больших рисков при реализации итогового результата. В связи с этим разработка происходит в виде последовательности версий, с возможностью изменений требований в процессе реализации или же полной отмены дальнейшего развития проекта.

Достоинства модели:

* Позволяет как можно быстрее предоставить пользователям начальный продукт, имеющий полную работоспособность, позволяя начать процесс уточнения функционала и добавления требований заказчиком;
* Допускает возможность изменения требований при разработке системы;
* Обеспечивает больший уровень гибкость при разработке;
* Надежность и устойчивость системы увеличиваются, поскольку ошибки и слабые места системы обнаруживаются при каждой итерации данных;
* Позволяет совершенствовать процесс разработки – анализ данных, проводимый во время каждой итерации, позволяет оценивать, что должно быть изменено либо добавлено в разработку, и улучшить процесс на следующей итерации;
* Риски заказчика уменьшаются, поскольку в любой момент разработки можно завершить создание не имеющего перспектив приложения.

Недостатки модели:

* Разработчик не имеет определенности в перспективах развития разрабатываемого проекта, что характеризуется возможностями отмены для заказчика;
* Затруднены операции временного и ресурсного планирования всего проекта в целом.

Для решения проблемы, описанной выше, вводятся ограничения по времени на каждую из стадий цикла разработки. Переход на следущую стадию осуществляется по мере наступления дедлайна, даже если не все запланированные функции были реализованы и не вся работа выполнена. План составляется на основе статистики времени выполнения подобных задач, а также из личного опыта разработчиков.

Из-за того, что наша разрабатываемая информационная система является простой и по своим масштабам довольно небольшой, было решено выбрать урезанную каскадную модель жизненного цикла данной приложения. Из изменений модели – совмещение этапов системного анализа и анализа требований, также совмещение этапов проектирования, кодирования и тестирования. На выходе получаем всего 3 этапа-блока, которые можно назвать как: Оценка требований и подбор инструментов для реализации, реализация и устранение критичных неполадок, развертывание системы и её поддержка.

Основными требованиями для приложения являются:

* Адекватная скорость работы
* Удобный понятный интерфейс
* Расширяемость системы
* Корректная работа с AstralToolBox
* Быстрая работа с исходными данными

Требования к функционалу приложения:

* Загрузка сертификатов пользователя для заполнения полей формы сайта
* Генерация пакетов документов по заданным параметрам в форме сайта
* Подпись сгенерированных документов электронной цифровой подписью полученной из сертификатов
* Получение архива со сгенерированными пакетами данных

Требования к системе пользователя:

* Наличие или возможность установки браузера «Google Chrome»
* Наличие монитора
* Наличие клавиатуры и мыши для навигации
* Минимальный объем ОЗУ 2Гб
* Наличие установленного ПО «Python 3.6.5» (и выше)
* Наличие установленного ПО «ViPNet CSP»
* Наличие установленного ПО «AstralToolBox»
* Наличие сертификатов предполагаемого участника документооборота для электронной цифровой подписи

# 2.2 Проектирование модели и архитектуры построения приложения

При написании любой программы, особенно, если для ее корректной работы требуется большое количество модулей, также стоит заранее проработать построение приложения.

В процессе разработки нередко приходится переделывать некоторые модули программы, и если модель приложение предусматривает такой расклад событий, то остальные модули сохранят свою работоспобность.

Чтобы это работало желательно придерживаться семи основных принципов:

1. Эффективность системы – программы должна решать поставленные задачи с максимально корректно в минимальные сроки, при этом не забывая об надежности, производительности, безопасностью и растущей со временем нагрузкой.
2. Гибкость системы – со временем приходится менять некоторые требования или добавлять новые. Задача разработчика получить такую модель приложения, чтобы вносить эти изменения с минимальными затратами по переписыванию кода. Для этого архитектура построения программы должна предполагать возможное дальнейшее развитие проекта.
3. Расширяемость системы – в начальном этапе разработки имеет смысл использовать принцип добавлять только самый необходимый функционал, но при этом не забывать про возможность наращивания, того самого, функционала не самой первой необходимости. То есть это предполагает расширение программы преимущественно написанием дополнительного кода, а не переписыванием старого.
4. Масштабируемость процесса разработки – желательно иметь наличие возможности сокращения процесса разработки проекта, посредством добавления новых людей. Модель должна предполагать распараллеливание задач программы по модулям, которые в свою очередь можно распределить между сотрудниками.
5. Тестируемость – адекватная возможность тестирования проекта, позволяет вовремя отловить наличие ошибок, тем самым позволяя конкретно выполнить принципы, описанные в пунктах выше.  
   Для этого в процессе работы обычно используют GIT (Распределенная система управления версиями). В структуре проекта Git используются несколько репозиториев или веток, один из которых является истинно верным. Обычно главная ветка называется origin или master. А также множество веток для каждой поставленной задачи. Тем самым тестировщик может удобно и быстро загрузить нужную ветку с задачей и составить по ней отчет, которой также можно отобразить в Git.
6. Возможность повторного использования – предполагает, что есть вероятность миграции людей от одного проекта к другому и пришедшему новому человеку будет довольно сложно вникнуть в проект. Потому желательно использовать наименее экзотические методы и средства реализации, естественного не в ущерб адекватности.

Принципы ООП - это четыре главные особенности, которые вместе образуют парадигму объектно-ориентированного программирования. Понимание и грамотная реализация данных принципов позволит как избавиться от ошибок и неточностей в коде, так и реализовать практически все вышеперечисленные задачи.

1. Наследование.

Наследование — механизм кода, который дает возможность описания нового класса на основе уже существующего в коде. Свойства и функционал родительского кода после этого наследуются новым классом. Главный плюс использования наследования – возможность использовать код повторно, поскольку поля и методы, описанные в родительских классах, можно спокойно использовать в классах, реализованных на их основе.

Механизм наследования имеет высокую гибкость, и недостающие в наследующих классах функции можно добавить отдельно.

2. Абстракция

Абстракция - выделение главных, наиболее значимых характеристик предмета и наоборот — отбрасывание второстепенных, незначительных. В более широком смысле - это придание объекту уникальных характеристик, отличающих его от остальных объектов, таким образом, четко определив его концептуальные границы. В результате использования данного принципа способы использования отделяются от способов реализации, становясь более простыми объектами. Данный подход является одной из основ ООП. Применение абстракционного принципа позволяет работать с объектами, не вдаваясь в особенности их реализации.

3. Инкапсуляция

Инкапсуляция - ограничение доступа к данным и возможностям их изменения. В более широком смысле - это свойство, позволяющее пользователю не задумываться о сложности реализации модуля, а взаимодействовать с ним посредством интерфейса, что также объединяет и защищает жизненно важные для компонента данные. При этом пользователю предоставляется только спецификация объекта. Объект доступен для пользователя только в качестве предоставленного интерфейса. Реализуется с помощью ключа public. Пользователю недоступно использование закрытых данных и методов. Реализуется с помощью ключей private, protected, internal.

4. Полиморфизм

Полиморфизм — это возможность работы с несколькими различными типами данных так, будто это один и тот же тип. При этом поведение объектов будет различаться в зависимости от того типа данных, к которому они на самом деле принадлежат. Язык поддерживает возможность полиморфизма в случае, если классы, имеющие одинаковые требования, особенности, могут быть по-разному реализованы — например, реализация класса может быть подвержена изменению в процессе наследования.

В данном приложении была выбрана модульная архитектура. Ввиду ее соответствия всем нормам современных требований к написанию программ, имеющих сложную конструкцию.

Главным принципом модульной архитектуры является глобальная декомпозиция, которая исполняет важнейшую роль снижения сложности работы проекта.

Иерархическая декомпозиция предполагает собой детализацию высокоуровневых процессов проекта на более конкретные точные процессы, обеспечивая возможность представления работы процессов высокого уровня с помощью нижележащих процессов.

# 2.3 Проектирование модели данных и базы данных приложения

Эволюция модели данных

1. Начальный этап

На начальном этапе проектирования модели данных первоначальные задачи могут выполняться вместе с процессом проведения разработки опытных прототипов. В проектах с уже существующей БД, проектировщик может использовать структуру существующей базы для реализации начальной физической модели.

1. Этап уточнения

На этапе уточнения создается контрольная версия системной архитектуры. В крупных системах одной из основных проблем, влекущих за собой низкую производительность приложения, может являться неудачное проектирование модели данных на начальных этапах развития проекта. По этой причине как разработка прототипа, позволяющего произвести оценку будущей производительности и необходимых системных требований заранее, так и моделинг данных являются существенными этапами разработки модели.

К моменту окончания этапа уточнения основные структуры БД должны находиться на своем месте, для возможности поддержки выполнения определенных сценариев, необходимых для работы приложения.

1. Этап построения

Во время этапа построения модели не должна проводиться основная реструктуризация, однако во время различных итераций этапа построения могут быть добавлены как дополнительные таблицы, так и элементы хранения данных - на основе проектирования набора вариаций использования, которые созданы для текущей итерации проекта. При проектировании модели данных на этом этапе основное внимание проектировщика должно уделяться уровню производительности системы и исправлению появляющихся ошибок с помощью описанных выше ООП методов программирования

Именно на этапе построения модели проектировщик пользуется таким инструментом, как создание физической модели данных, которая обслуживается с помощью либо внесения изменений непосредственно в модель, либо путем инструментального считывания внесенных таким образом обновлений.

1. Этап внедрения

Следующий этап, часто гораздо более протяженный во времени, чем предыдущие – это внедрение, когда нужно произвести интеграцию полученной системы, постройку дополнительных систем и последнее обслуживание уже реализованной модели. Модель данных, как и модель проектирования, обслуживается на этапе внедрения в ответ на поступающие запросы заказчика на изменение. Проектировщик должен реализовывать синхронизацию модели и базы данных, в то время, пока приложение проходит окончательное тестирование и пока не пройдет развертка в рабочую версию.

Можно представить модели для рассматриваемой предметной области с использованием функционального и объектно-ориентированного подходов к проектированию. На основании этих моделей осуществляется построение соответствующих диаграмм с описанием бизнес-функций, входящих в состав моделируемых бизнес-процессов.

На рисунке 1 ниже предcтавлена диаграмма IDEF3, отражающая представление рабочих процессов проектируемой информационной системы со стороны пользователя.

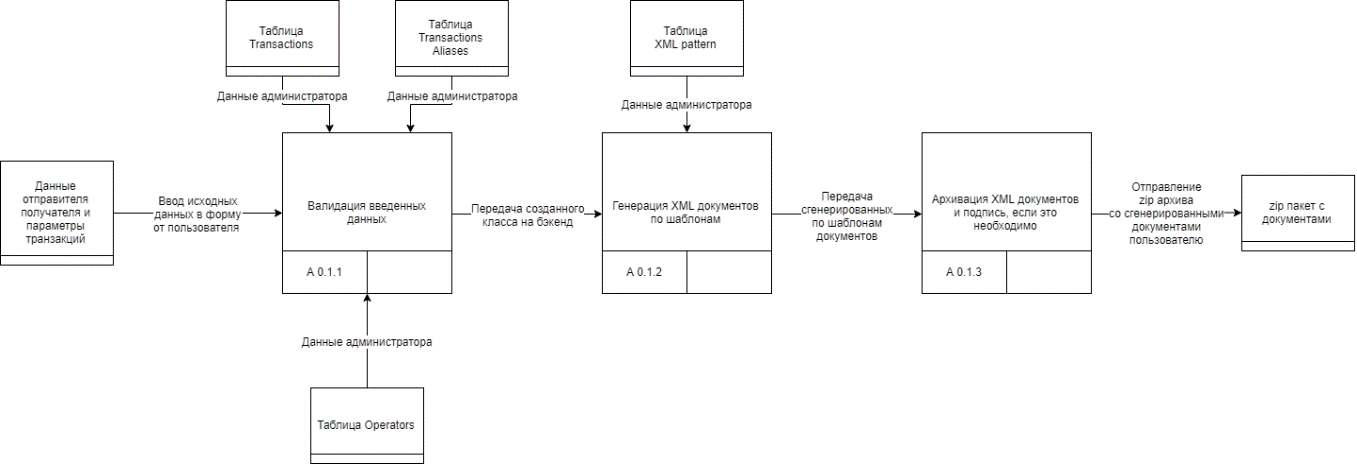


Рисунок 1. Диаграмма IDEF3

На данной диаграмме:

A 0.1.1. Валидация введенных данных: запрос к базе данных администратора – корректировки введенной информации на уровне фронтенда.

A 0.1.2. Генерация XML документов по шаблонам: обращение к базе данных администратора – выгрузка нужного шаблона из базы – корректировка шаблона с учетом введенных пользователем данных.

A 0.1.3. Архивация XML документов и подпись, если это необходимо: сбор пакетов документов по указанному пользователем методу – шифрование документа (при необходимости) – подпись документа (при необходимости).

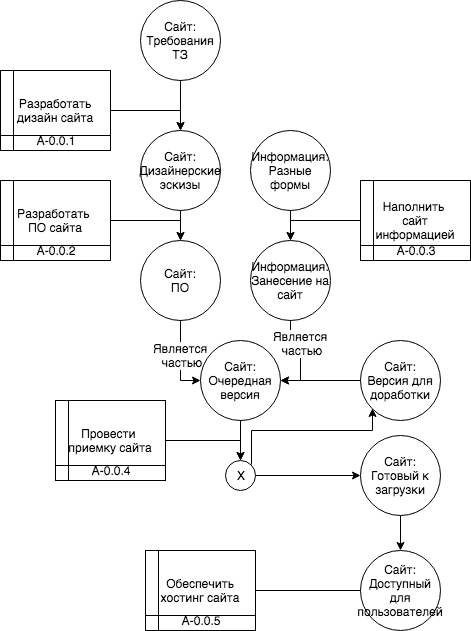
Диаграмма переходов состояний системы представлена на рисунке 2.  


Рисунок 2. Сеть переходов состояний системы

База данных в текущей версии сайта основана на SQL. В тестовом сайте база данных может быть реализована на yaml файлах, где в каждом отдельном файле храниться некоторая информация. Этот способ может быть оправдан при небольшой нагрузке на сайт.

SQLite

SQLite — это компактная встраиваемая система управления базами данных. Исходный код является открытым и все что касается этой СУБД очень хорошо документировано.

MySQL

Основным достоинством является огромная популярность и довольно большое количество документации. Используя MySQL, можно создать огромный проект, требует некоторых знаний для реализации. К минусам можно отнести не слишком высокую скорость выполнения запросов.

Redis

Это довольно молодая технология, позволяющая существенно сократить время на обработку запроса за счет хранения элементов БД в ОЗУ. Идеально для небольших проектов. Технология довольно легка для освоения, есть достаточное количество документации. Данная технология в конечном итоге идеальна для кеширования и совместного использования с PostgreSQL.

Исходя из всего вышеперечисленного было решено использовать SQLite т.к. это технология очень проста для освоения и работы и идеальна для небольших проектов. Все это вписывается в требования для разрабатываемой информационной системы: Ориентирована на достижение максимальной производительности на атомарных операциях, написана на Си, интерфейсы доступа созданы для большинства основных языков программирования, Хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения. Также поддерживает транзакции и пакетную обработку команд.

Даталогическая модель БД приведена на Рисунке 5.



Рисунок 5. Даталогическая модель БД

На рисунке 6 представлена инфологическая модель БД.



Рисунок 6. Инфологическая модель БД

На рисунке 7 представлена диаграмма потоков данных

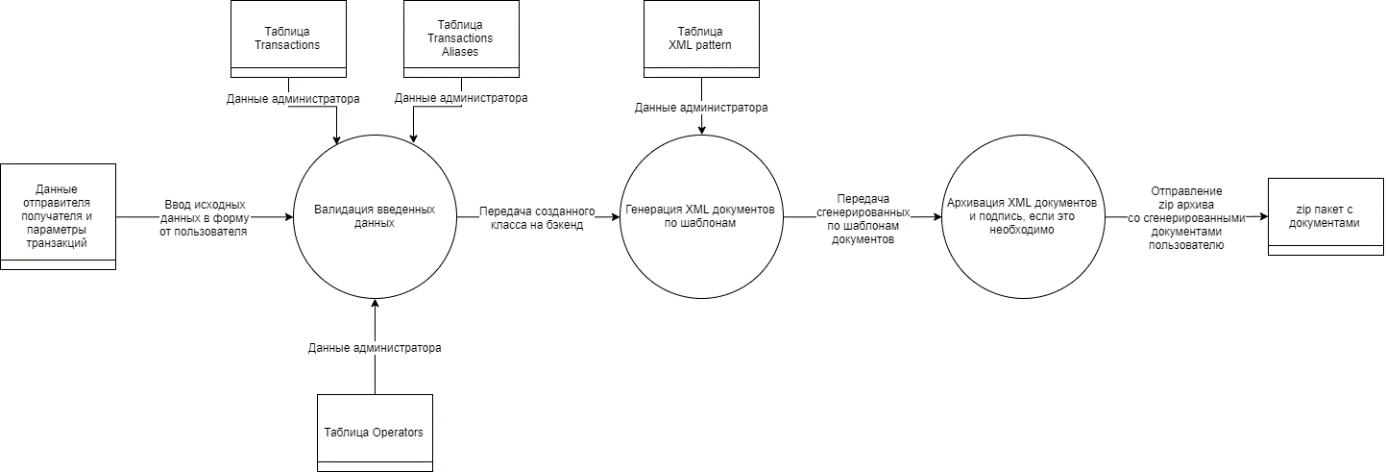


Рисунок 7. Диаграмма потоков данных

В этой модели:

Поток 1 Валидация введенных данных: запрос к базе данных администратора – корректировки введенной информации на уровне фронтенда.

Поток 2 Генерация XML документов по шаблонам: обращение к базе данных администратора – выгрузка нужного шаблона из базы – корректировка шаблона с учетом введенных пользователем данных.

Поток 3 Архивация XML документов и подпись, если это необходимо: сбор пакетов документов по указанному пользователем методу – шифрование документа (при необходимости) – подпись документа (при необходимости).

# 2.4 Проектирование пользовательского интерфейса

Интерфейс пользователя – это одна из важных частей программы, постоянно находящаяся на виду у конечного пользователя, призванная обеспечивать управление предоставленными пользователю его уровнем допуска к системе функциями. При разработке ПО программисту не стоит выделять все время разработке исключительно кода, откладывая на потом реализацию интерфейса. Для удобства пользователей и реализации грамотного, отвечающего требованиям команды тестировщиков, при наличии оной, эргономичного интерфейса должно быть затрачено достаточное количество времени для полной бесшовной реализации всего функционала. Данное время следует учесть при озвучивании итоговых сроков разработки программы.

Удобство пользовательского интерфейса входит в понятие эргономичности разрабатываемого приложения, которое включается в процессы разработки и тестирования продукта как часть системы качества. Разработка интерфейса должна вестись совместно с дизайном программного продукта в целом.

Программа разрабатывается для обеспечения продуктивной работы пользователя, т.е. для того, чтобы он, с помощью реализуемой разработчиком компьютерной программы, быстрее, надёжнее и качественнее решал свои производственные задачи.

С точки зрения удобства пользовательского интерфейса в программе важно создать такой набор функций, реализованных для пользователя, который увеличит производительность работы в сравнении с программными продуктами подобного типа, ускорит обучение пользователей новым для них программным компонентам, а также обеспечит высокий уровень удовлетворенности пользователем от итоговой работы с приложением.

Разработчику необходимо тщательно продумать и осознать сценарий взаимодействия программы с пользователем, приведя его к оптимальной системе выполнения задач, и реализовать ПИ в соответствии с этой системой. Разработчиком во время процесса разработки программного продукта придумывается и реализуется сценарий различных типов взаимодействия продукта и пользователя, а так же реализация пользовательского интерфейса согласно данному сценарию.

Для того, чтобы разобраться в технологии решения задач пользователя, разработчику необходимо выяснить следующие моменты:

* Какая информация необходима пользователю для решения поставленных им задач;
* Какую информацию пользователь может не учитывать;
* Какие решения принимаются пользователем в процессе взаимодействия с приложением;
* Может ли пользователь решать несколько задач одновременно;
* Какие типовые операции, либо какие из функций, реализованных в приложении, использует пользователь при решении задачи;
* Что произойдет при обхождении пользователем придуманного Вами алгоритма действий, пропуская либо обходя часть реализованных для решения задач шагов.

Дизайн интерфейса должен обеспечивать максимальную производительность пользователя при решении задач и приводить к: сокращению длительности операций чтения, редактирования и поиска информации, что достигается грамотной эргономикой интерфейса, уменьшению времени навигации и выбора команды, увеличению длительности устойчивой работы пользователя.

# Глава 3. Программная реализация генератора электронного документооборота «EDOCript»

# 

# 3.1 Обоснование выбора состава программных средств реализации

Основным языком программирования проекта был выбран Python, ввиду его огромной мощности и распространенности как в решении бытовых задач, так и написании сложных вещей наподобие нейросетей.

Python представляет собой универсальный мультипарадигменный скриптовый язык.

Универсальность основывается на том, что у данного языка не имеется определенной привязки к какой-либо области разработок. Он отлично подходит для реализации веб-приложений, десктопных решений и множества других вещей.

Мультипарадигменность заключается в минимальном ограничении разработчика во многих аспектах. Любая концепция, будь это функциональных подход или объектно-ориентированный, имеет возможности реализации, тем самым позволяет реализовать самые сложные проекты разной направленности.

Скриптовость выражается в выполнении системой различных кратких описаний действии, именуемых сценариями, то есть по сути является гибридом системного и командного языка.

Главные преимущества Python заключаются в:

* Максимально низкий порог вхождения, тем самым позволяет человеку с минимальными знаниями теории начать простые разработки за очень короткий срок.
* Язык очень грамотно спроектирован. Он хорошо отлажен, в процессе разработки принимает в себя новые приемы программирования различных типов и конфессий.
* Предельно читабельный синтаксис позволяет легко вникать в уже написанный код.
* Легкость работы привела к рождению огромного количества сторонних библиотек, максимально разной направленности.
* Невероятная легкость в процессе развертывания на почти любой платформе начиная Windows заканчивая Arduino.

Но лучше всего этот язык характеризует его философия «The Zen of Python»:

* Красивое лучше, чем уродливое.
* Явное лучше, чем неявное.
* Простое лучше, чем сложное.
* Сложное лучше, чем запутанное.
* Плоское лучше, чем вложенное.
* Разреженное лучше, чем плотное.
* Читаемость имеет значение.
* Особые случаи не настолько особые, чтобы нарушать правила.
* При этом практичность важнее безупречности.
* Ошибки никогда не должны замалчиваться.
* Если они не замалчиваются явно.
* Встретив двусмысленность, отбрось искушение угадать.
* Должен существовать один и, желательно, только один очевидный способ сделать это.
* Хотя он поначалу может быть и не очевиден, если вы не голландец.
* Сейчас лучше, чем никогда.
* Хотя никогда зачастую лучше, чем прямо сейчас.
* Если реализацию сложно объяснить — идея плоха.
* Если реализацию легко объяснить — идея, возможно, хороша.
* Пространства имён — отличная штука! Будем делать их больше!

Для разработки генератора пакетов электронного документооборота необходимо использование одного из нескольких инструментов, позволяющих с помощью языка Python создать сайт, поскольку этот язык является наиболее освоенным у исполнителей.

Рассмотрим несколько таких инструментов:

Django.

Django является очень распространенным веб-фреймворком. Основными достоинствами данной платформы являются:

* Очень подробная документация, позволяющая достаточно быстро разобраться в своей работе.
* ORM (Object-relational mapper), или система проецирования реляционных данных в объекты. Практически во всех случаях не требуется использование SQL запросов, тем самым часто обеспечивает защиту от SQL-injection уязвимости.
* Автоматически создаваемая панель администратора сайта. Сильно ускоряет процесс разработки в плане отсутствия надобности прописывания прав администратора, при сохранении всех норм безопасности.
* MTV (Model-Template-View), предполагает разделение моделей баз данных, шаблонов веб-странницы и логику этих странниц на разные модули, что очень помогает организовать работу.

Flask;

Flask – это минималистичный инструмент, предоставляемый по принципу «Только нужное». Идеален для небольших проектов, отлично документирован, организован по принципу MTV, однако для любого более крупного проекта потребует большого объема самостоятельных доработок. Этот инструмент идеален для проектирования систем подобных нашей.

Bootstrap;

Bootstrap – это одна из популярных frontend библиотек. Её разработчиком является компания Twitter. Поскольку работать с css довольно сложно, а без css сделать красивый сайт не удастся, то необходимо использовать Bootstrap в нашем проекте. Данный фреймворк характеризуется рядом прекрасных достоинств:

* Адаптивность – позволяет корректно отображать сайты на экранах различных устройств и диагоналей, при этом оставаясь читабельным и адекватным в плане использования.
* Кросс-браузерность – в связи с различием работы некоторых движков популярных веб-браузеров, не всегда одинаково описанный элемент в коде будет отрисован одинаково во всех браузерах. Этот нюанс полностью учитывался при разработке данного фреймворка.
* Легкость освоения – выражена в огромном количестве статей по поводу работы этой системы стилей. А также не перегруженностью и понятностью написанных классов, используемых в работе.

Естественно, Bootstrap не лишен недостатков.

Например:

* Шаблонность – очень часто сайты использующие макеты классов Bootstrap выглядят довольно похоже друг на друга. Решением этой проблемы может быть отказ от готовых решений, что повлечет собой весьма большие затраты по времени освоения понятий UX (User experience), и применить их в проекте. Или банальном переписывании кусков кода с импортированными классами Bootstrap.
* Старые браузеры – могут не очень корректно отображать наполнение странницы, ввиду банального устаревания методов работы с css.

И поскольку для нашего приложения не требуется наличие уникального внешнего вида, а тестировщики обязаны работать с самым современным оборудованием, то это решение будет идеальным для наших задач.

Так как основной задачей систем электронного документооборота является обеспечение валидности информации, содержащейся в передаваемых документах, автоматически приходит в голову вопрос обеспечения безопасности документов. Для таких целей как правило обязательно использование электронной цифровой подписи.

В связи с отсутствием профессиональных знаний в области шифрования было решено использовать стороннее программное обеспечение, позволяющее проводить операции с сертификатами условных пользователей смоделированного документооборота.

ViPNet CSP 4.

Является российским криптопровайдером, сертифицированным ФСБ России как средство защиты информации и электронной подписи.

Позволяет создавать ключи электронной подписи, формировать и проверять валидность подписи по ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012.

Хэширует данные по ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012.

Шифрует и обеспечивает имитозащиту данных по ГОСТ 28147-89

Базовая комплектация программы предоставляет класс защищенности КС1, утвержденный приказом ФСБ РФ от 27 декабря 2011г. № 796 «Об утверждении Требований к средствам электронной подписи и Требований к средствам удостоверяющего центра»

Существует бесплатная версия данного ПО, тем самым делая данный криптопровайдер идеальным для решения поставленных задач при реализации нашего программного продукта.

Для подписи электронных документов с соблюдением всех современных нормативов было также решено использовать сторонний продукт с уже реализованной требуемой функцией.

Выбор пал на AstralToolBox. Программное обеспечение, предназначенное для того, чтобы дать пользователям интернет обозревателей выполнять функции операционной системы, недоступные самим браузерам. Например, MS Crypto Api.

Приложение реализовано по принципу локального веб-сервера, работающий по протоколу WebSocket, принимающий запросы пользователя из браузера. Сервер реализован для платформы Net Framework.

Приложение предоставляет интерфейс для выполнение основных криптографических функций:

* Генерация ключевой пары
* Генерация запроса на сертификат
* Установка сертификата в контейнер
* Проверка подписи
* Подписание данных
* Шифрование данных
* Расшифрование данных
* Получение списка сертификатов с хранилища пользователя раздела «Личные»

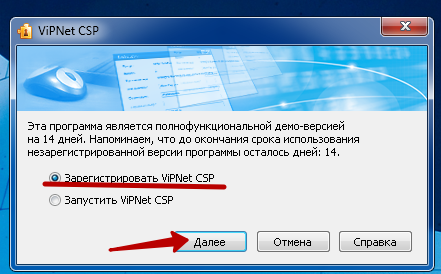
Для всего этого реализовано весьма удобное для работы API.

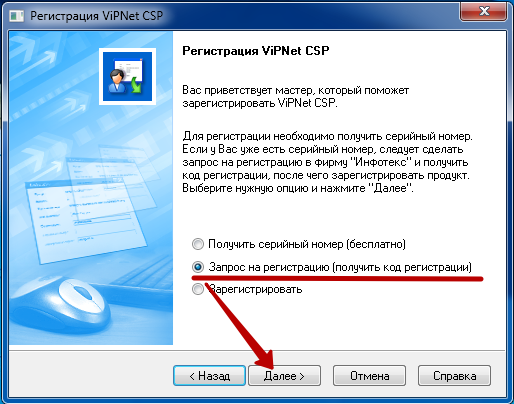
Но самое важное для нас то, что AstralToolBox полностью совместим с работой ViPNet CSP 4, что сильно упрощает процесс работы с документами и интеграцию автоматизации в код проекта.

# 3.2 Описание пользовательского интерфейса и инструкций пользователю

Для начала работы пользователя требуется предварительная подготовка. Во-первых, нужно установить требуемое программное обеспечение.

Регистрация программы. Открываем VipNet CSP.



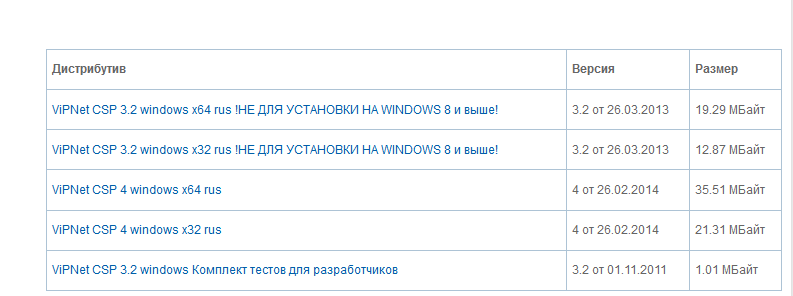


Способ запроса на регистрацию – Через Интернет (online).

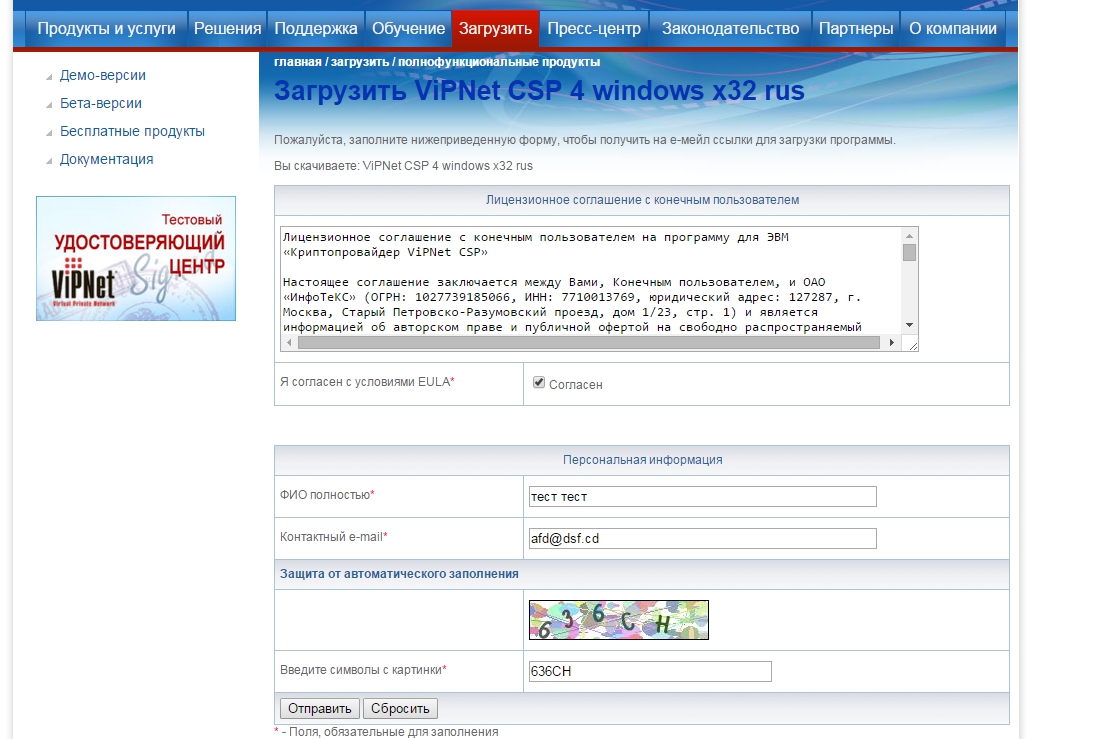
Далее необходимо зайти на сайт Infotecs

http://www.infotecs.ru/downloads/product\_full.php?id\_product=2096

Выбираем нужный дистрибутив.

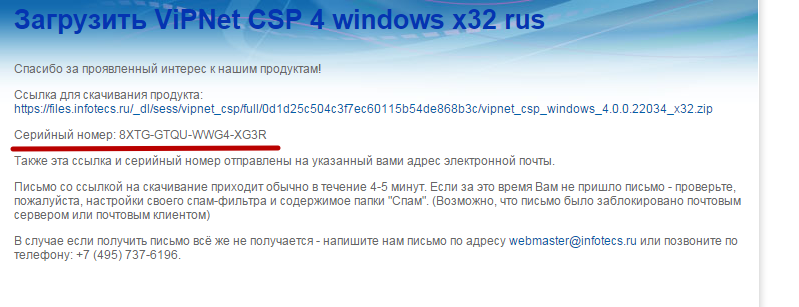


Далее:

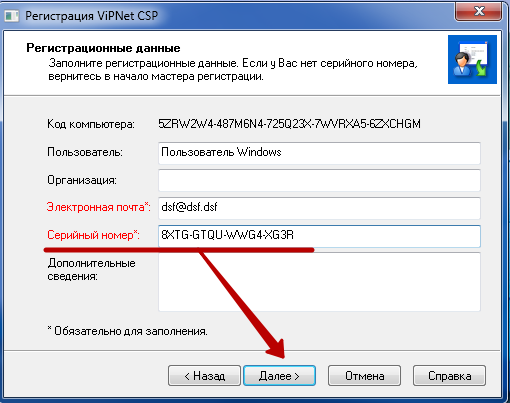


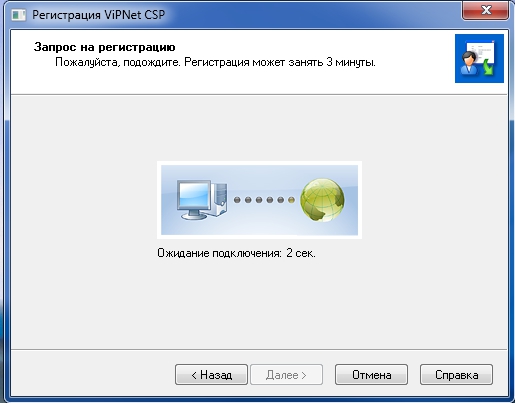
В персональной информации заполняем.

Получаем нужный нам серийный номер.

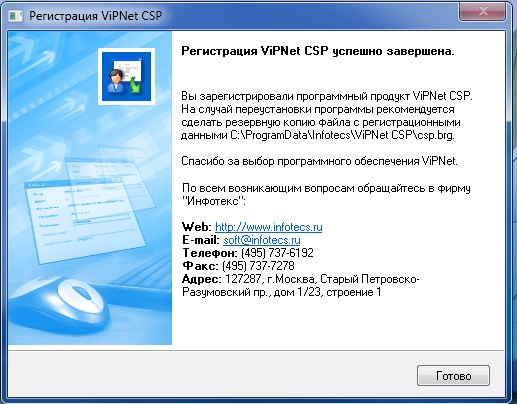


Вписываем его в регистрационные данные.

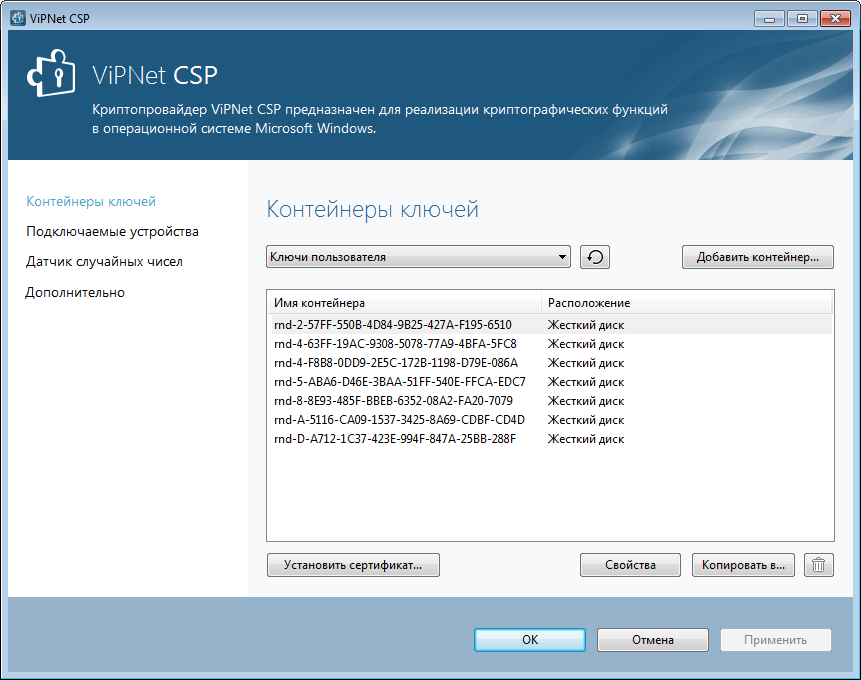




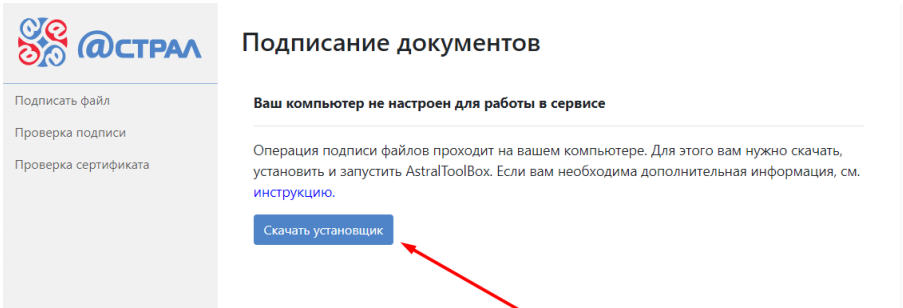
Обычно не больше 10 секунд. Если дольше значит что-то пошло не так.



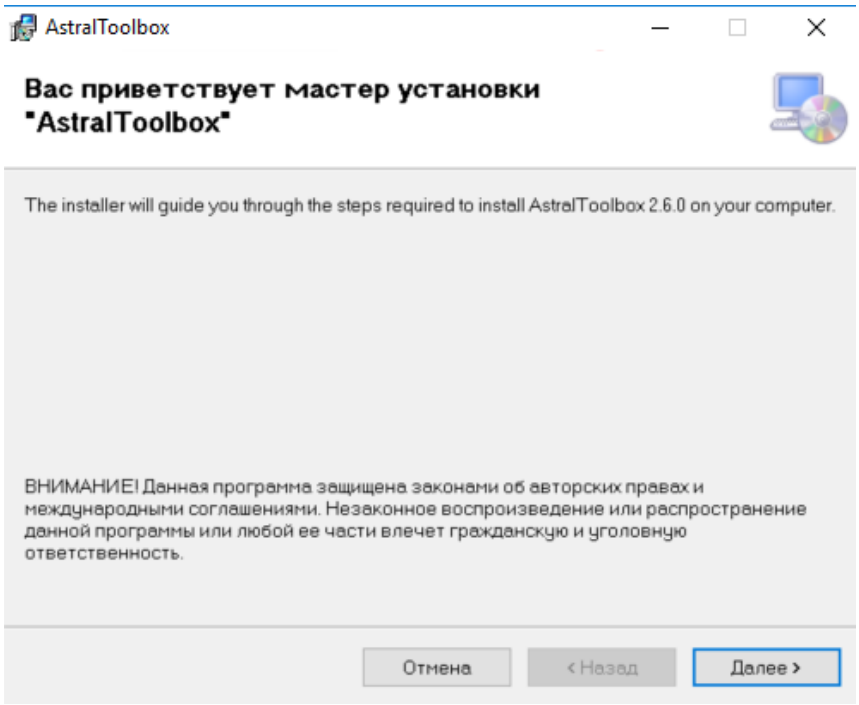
После завершению установки программного продукта ViPNet CSP 4, следует добавить в контейнер сертификаты предполагаемых пользователей моделируемого процесса электронного документооборота.



Теперь требуется установить AstralToolBox. Перейдите на странницу «http://crypto.keydisk.ru/». После загрузки установщика.

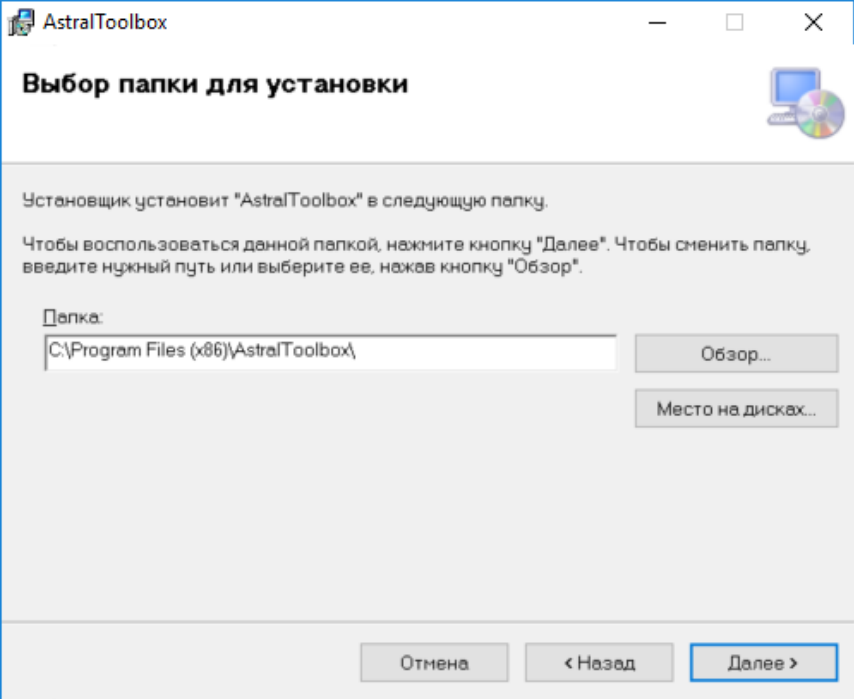


Запустите установочный файл. В открывшемся окне нажмите Далее.

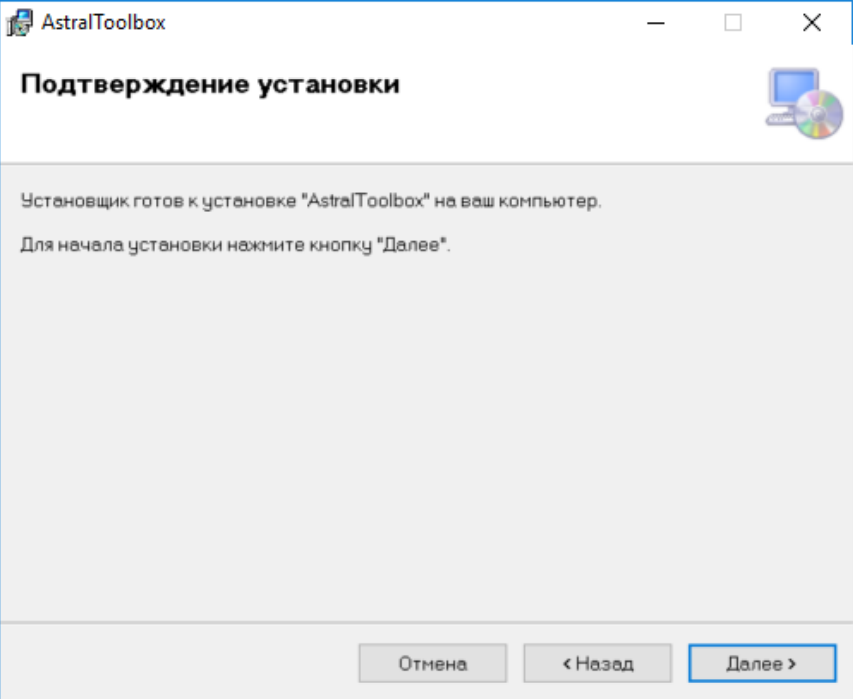


Перед вами экран выбора папки, в которую будет установлена программа. По умолчанию это «C:\Program Files(x86)\AstralToolbox». Если необходимо, укажите другую папку или выберите ее, нажав кнопку Обзор.

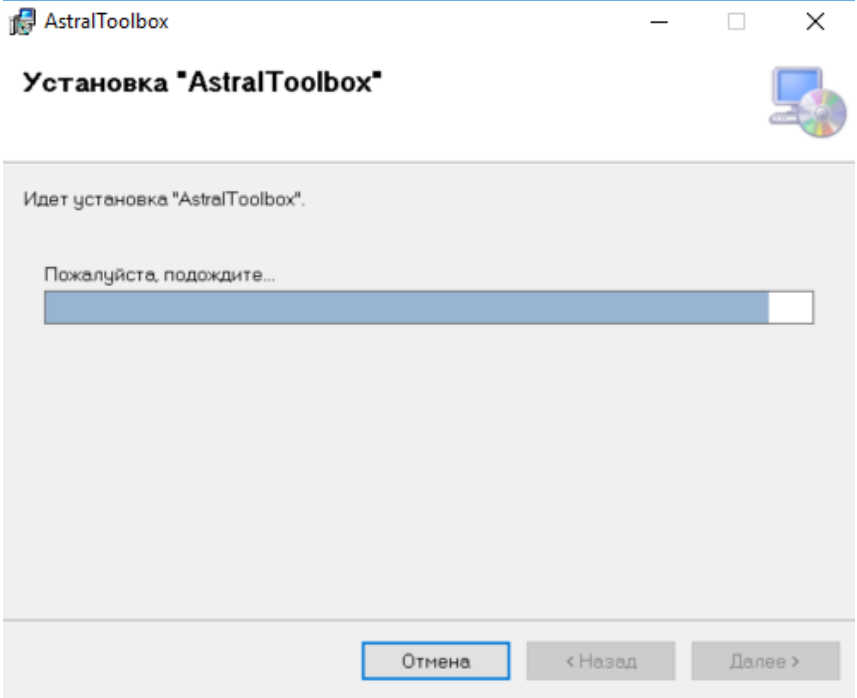
Нажмите Далее для прохождения.



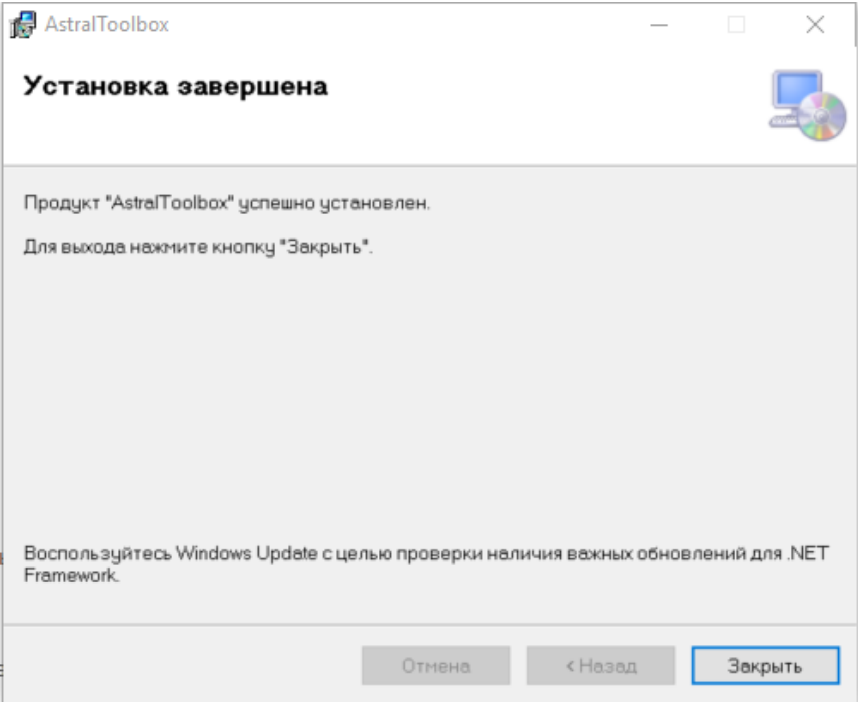
Установщик готов к установке. Нажмите Далее для установки.



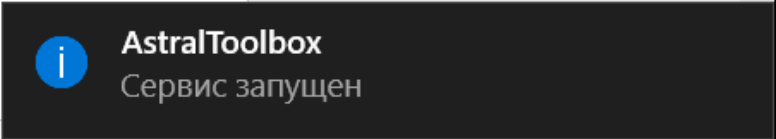
Дождитесь установки программы.



Программа успешно установлена. Нажмите Закрыть.



После установки AstralToolbox должен был автоматически запуститься на вашем компьютере.

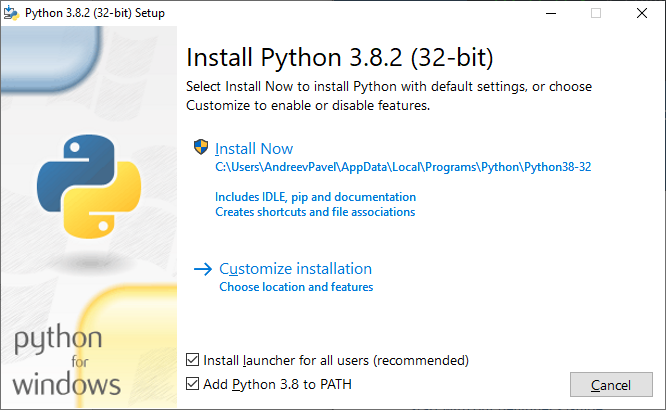


Если этого не произошло, заходим в папку, которую вы выбрали при установке программы. И запускаем AstralToolbox.exe

Далее устанавливаем Python. Переходим на сайт «https://www.python.org/» и сохраняем инсталлятор версии 3.6.5 и выше.

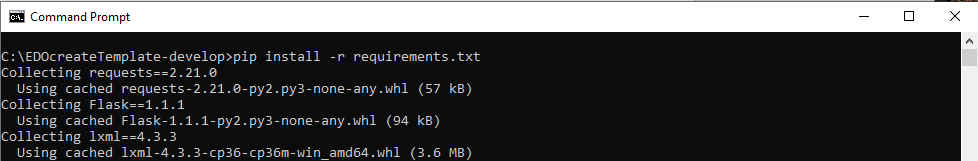


Во время установки обязательно ставим галочку на пункте «Add Python to PATH» или «Добавить путь в глобальные переменные».

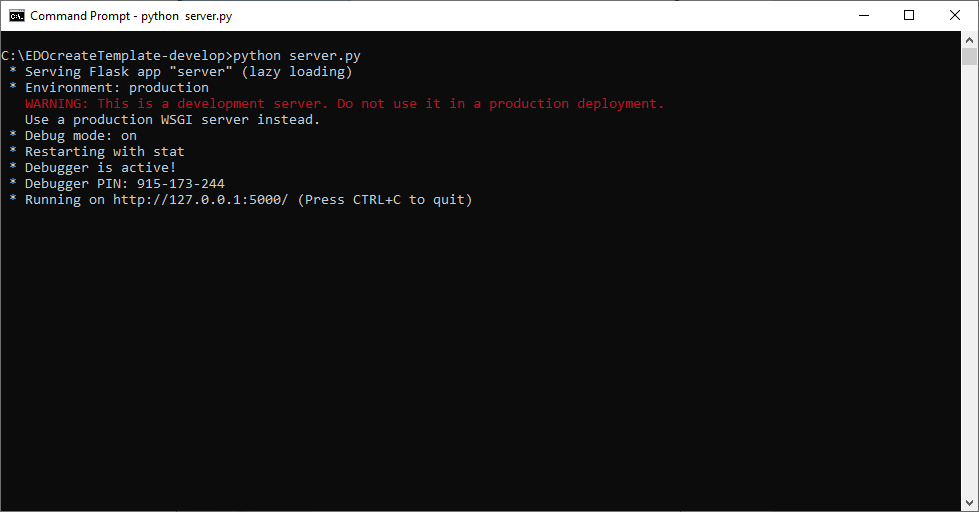


По окончанию установки можно приступать к установке генератора.

Перейдем в директорию проекта в командной строке Windows. Далее установим нужные библиотеки для Python посредством команды «pip install -r requirements.txt».



После всех приготовлений можно запускать сервер приложения «python server.py».



Если все прошло успешно, то сайт приложения будет доступен на странице, по адресу «http://127.0.0.1:5000/».

Общий алгоритм работы генератора для пользователей такой системы реализован и показан на рисунке 8.



Рисунок 8. Алгоритм сайта

При открытии страницы пользователю открывается форма для заполнения параметров генерации документов (рисунок 9). После этой процедуры, если данные введены достаточно корректно, пользователь получает zip пакет со сгенерированными документами.

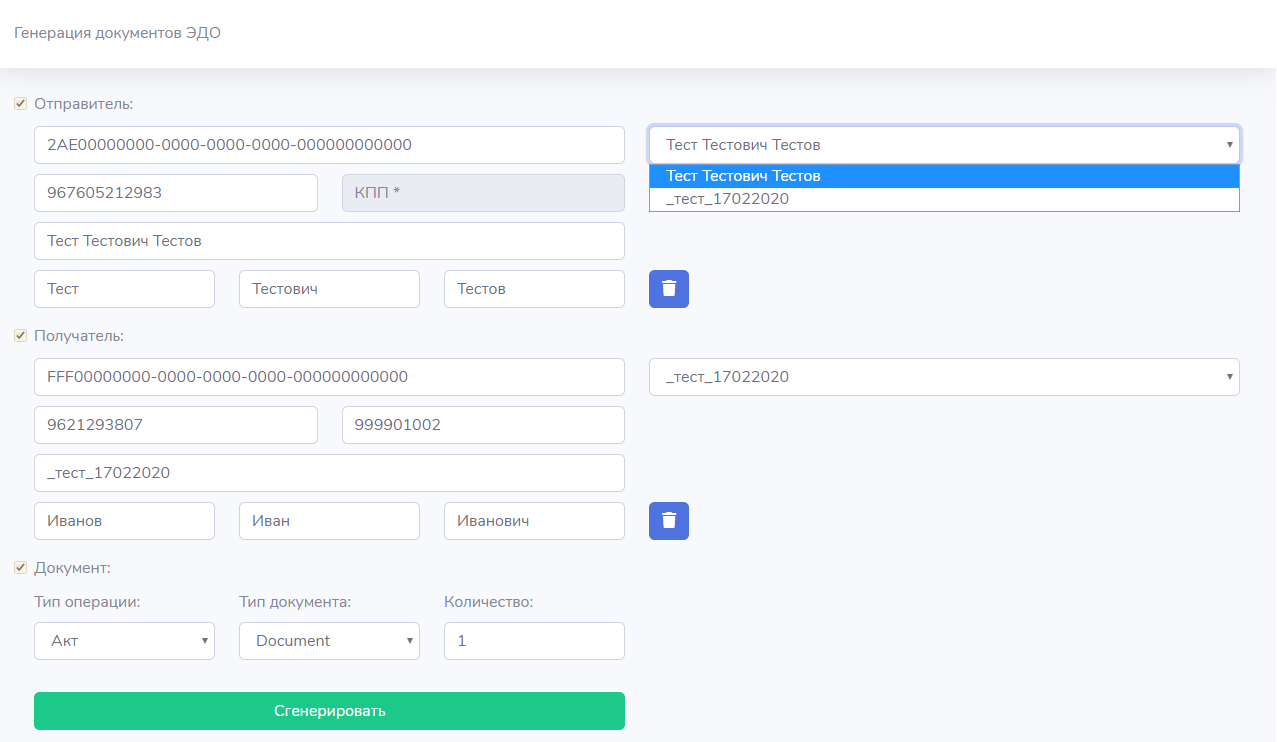
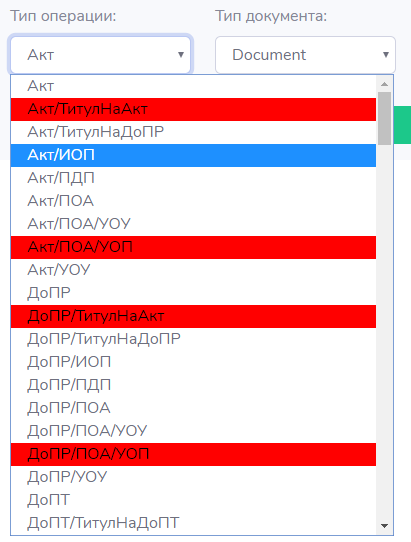


Рисунок 9. Форма заполнения параметров генерации

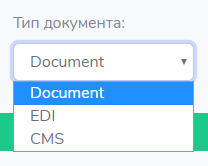
На страннице приложения присутствует форма для заполнения ключевых данных о двух абонентах, между которыми должен быть смоделирован электронный документооборот. Справа от основных полей находятся выпадающие списки с уже установленными сертификатами, для электронной цифровой подписи генерируемых документов. При выборе сертификата остальные поля автоматически заполняются в соответствии с данными из сертификата. Также поля можно редактировать по своему усмотрению, в угоду тестировщика.

В пункте формы «Документ» в подпункте «Тип операции» можно выбрать тип операции или цепочку генерируемых документов.рис?



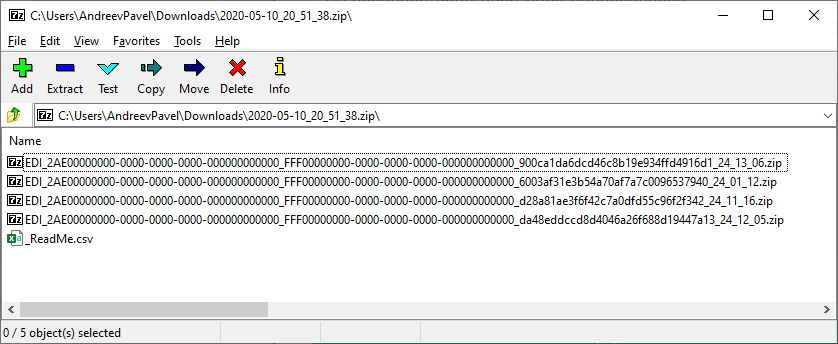
Красным выделены транзакции, которые устарели, и выбрать их нельзя.

В пункте «Тип документа» можно выбрать тип пакета генерируемого пакета для соответствующей системы.рис?



После указания количества желаемых документов и нажатия на кнопку «Сгенерировать» сервер отправит пользователю архив со сгенерированными пакетами

Структура полученного архива состоит из zip пакетов, содержащих в себе сам сгенерированный документ и прилагающуюся подпись к нему. Также в архиве лежит csv таблица с основными данными о полученных пакетах документов.???



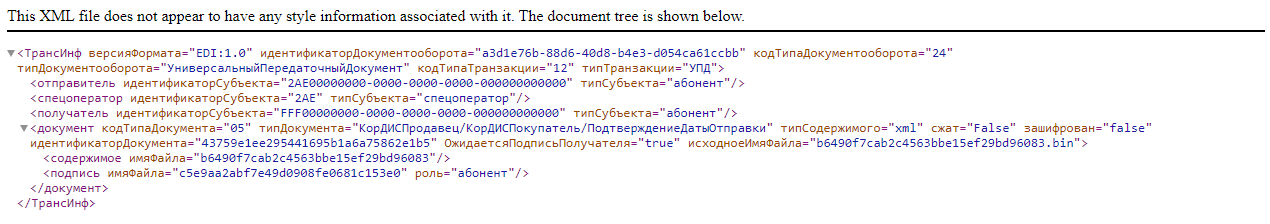
На этом функционал приложения заканчивается.

# 3.3 Руководство разработчика

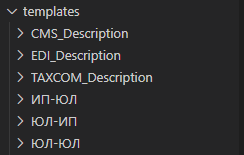
Для дальнейшего развития генератора пакетов стоит разъяснить, какие знания и действия пригодятся человеку, который продолжит развитие данного проекта или руководство разработчика.

Начнем с основ. Для начала разработчику требуются знания Python и реализация парадигм ООП (Объектно-ориентированного программирования), чтобы вникнуть в архитектуру приложения.

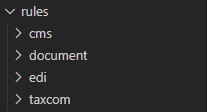
Также разработчику требуется иметь понимание электронного документооборота. Ведь взять любой шаблон для транзакции не получится. В данном генераторе учитываются три варианта построения шаблона, в зависимости от введенных данных из формы в интерфейсе сайта. Связано это с тем, что отправители и получатели могут быть как физическими лицами, так и юридическими, из-за чего у юридических лиц появляется пункт «КПП» его нужно подставлять в генерируемый документ, с учетом всех вытекающих из этого дополнительных тегов в самом документе, какие-то надо убрать, какие-то добавить. Шаблоны документов разделены на три вида, с учетом всего вышеперечисленного, на «ИП-ЮЛ», «ЮЛ-ИП» и «ЮЛ-ЮЛ». Однако этого недостаточно для создания пакета документа. Например, для создания пакета по стандарту EDI в пакете требуется иметь файл описание содержимого пакета, ведь некоторые файлы могут быть зашифрованы и лишены отличительных особенностей.



Для таких небольших xml файлов тоже выделены шаблоны, они рассортированы по папкам «cms», «edi», «document», «taxcom».

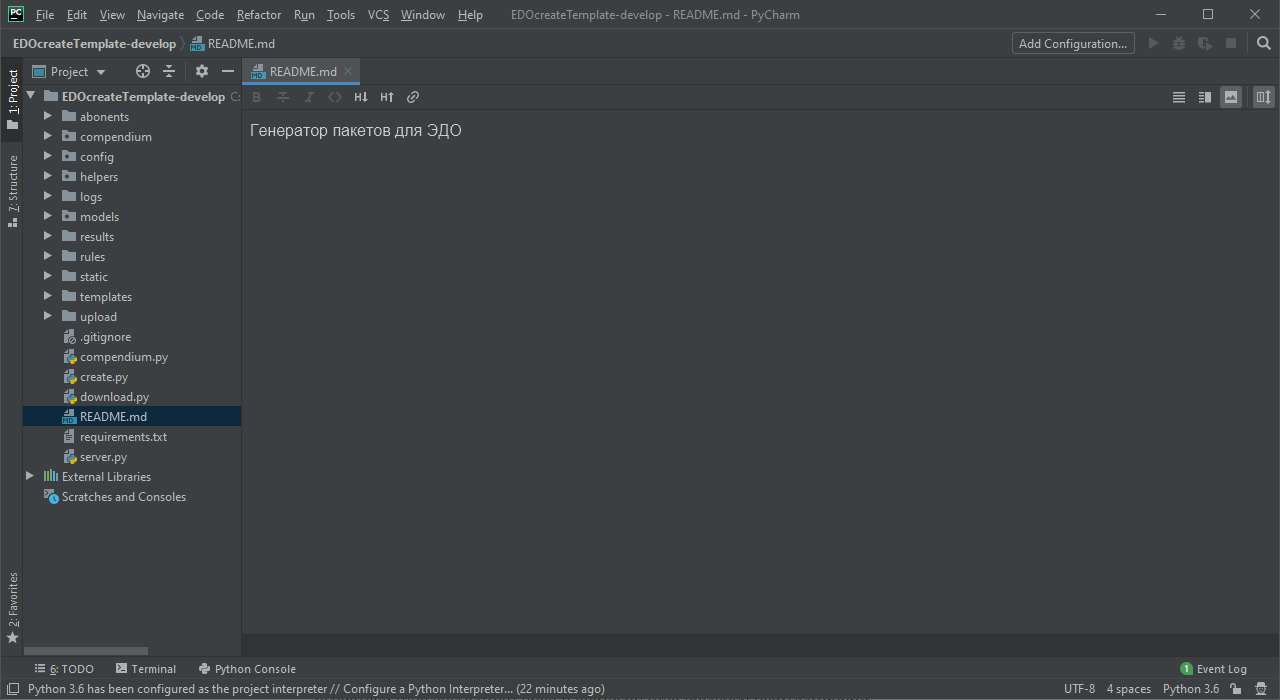


Чтобы заполнять полученные шаблоны, требуется прописать правила. Правила описаны на языке запросов к элементам XML-документа XPath. Хранятся все они в папке «rules» и отсортированы по моделям оформления пакетов документооборота. Все это разбиение сделано с учетом дальнейшего развития приложения и усложнения системы.

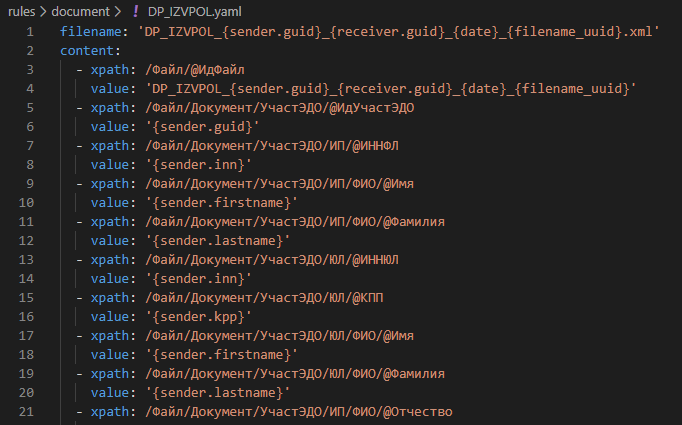


Заниматься разработкой стоит в IDE (Integrated Development Environment). В пример хочу привести разработку от компании JetBrains для разработки на языке программирования Python – «PyCharm».

PyCharm имеет очень весомое достоинство в виде мощной интеграции с современными фреймворками для работы в сфере разработок программного обеспечения. Также очень удобный редактор позволяет крайне быстро проводить навигацию по коду проекта, что очень чувствуется в больших, многоуровневых проектах. Для систем подобных нашей крайне необходим частое тестирование, в этом помогает встроенный отладчик среды разработки. Утилита поддерживает все свежие версии Flask, Django, IronPython, Jython, Cython, PyPy wxPython, PyQt, PyGTK и многие другие инструменты. В PyCharm можно, а главное удобно проводить Unit тестирование. Чтобы не засорять компьютер лишними данными библиотек, и исключить фактор влияния операционной системы на работы разрабатываемого приложения, есть возможность использовать виртуальные среды для функционирования приложения. А также очень важна полностью реализованная работа с Git системами.



Для добавления или изменения генерируемых шаблонов следует пройти в директорию «templates», далее добавить или изменить нужный шаблон документа. Хранятся они в формате XML. После чего проверить, есть ли нужный набор правил заполнения данного документа в директории «rules» и добавить, если таковой отсутствует.



# Заключение

В рамках выполнения выпускной квалификационной работы, достигнута поставленная цель и выполнены все предполагаемые требования задачи. А именно, разработано приложение генератор «EDOCript» для генерации пакетов с эмулированного электронного документооборота между двумя абонентами с учетом всех нормативов и стандартов, для упрощения работы тестировщика работающего в системах ЭДО.

Выполнены следующие задачи:

* Проанализирована предметная область
* Принято решение по выбору программной среды
* Разработаны технические требования к системе
* Разработан интерфейс пользователя
* Успешно разработано приложение генератор для генерации документооборота для систем тестирования ЭДО.

# Список использованных источников

# ПРИЛОЖЕНИЕ А