**Оглавление**

[**1.** **Определения и сокращения, принятые в программе** 1](#_Toc442548913)

[**2.** **Процесс получения первичного ключа и установка лицензионного ключа** 4](#_Toc442548914)

[**3.** **Проверка лицензионного ключа при старте приложения** 12](#_Toc442548915)

[**4.** **Использование лицензиатора** 14](#_Toc442548916)

[**5.** **Модуль версий ФАНЗ** 15](#_Toc442548917)

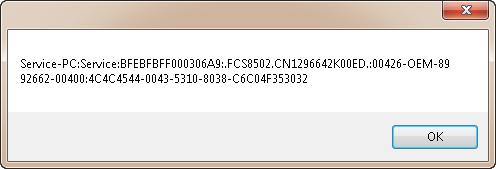
[**6.** **Дополнительно** 21](#_Toc442548918)

[**7.** **Схема взаимосвязей объектов** 22](#_Toc442548919)

# **Определения и сокращения, принятые в программе**

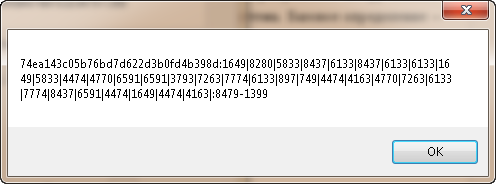
* **ФАНЗ** – Финансовый Анализ. Новые знания. Рабочее и зарегистрированное название программы.
* **Первичный ключ** – зашифрованная алгоритмом *MD5* текстовая строка, заключающая в себе информацию о текущей конфигурации клиентского компьютера, на котором установлена *ФАНЗ*. Собирает следующую информацию о системе: имя компьютера: имя пользователя: серийный номер HD (id), если несколько дисков, то несколько серийников: id процессора: id материнской платы: установленный код активации windows: *UUID.*

Выглядит в исходном виде следующим образом. Т.е., сюда попадают все сведения, которые алгоритм смог собрать.



Соответственно, после шифрования получаем хэш строки в таком виде (просто пример): ***dea5a7e6eea52f21a1de1d091e521196***. Это есть контрольная сумма, полученная в результате шифрования.

* **UUID –** уникальный идентификатор раздела, на котором установлена операционная система. Базовое определение – стандарт идентификации, используемый в создании программного обеспечения.
* **Лицензиатор –** утилита для формирования лицензионного ключа, на основе полученного первичного ключа и выбранной версии программы.
* **Лицензиар –** одна из сторон лицензионного соглашения, предоставляющая другой стороне — лицензиату — право на использование объекта лицензии.
* **Лицензиат –** [юридическое лицо](https://ru.wikipedia.org/wiki/%D0%AE%D1%80%D0%B8%D0%B4%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE) или [индивидуальный предприниматель](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8%D0%B2%D0%B8%D0%B4%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%B5%D0%B4%D0%BF%D1%80%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D1%82%D0%B5%D0%BB%D1%8C), имеющие лицензию на осуществление конкретного вида деятельности.
* **Лицензионный ключ** – формируется лицензиатором, устанавливается клиентом и дает право на использовании объекта лицензии. В нашем случае состоит из трех частей. Конечный вид лицензии следующий:

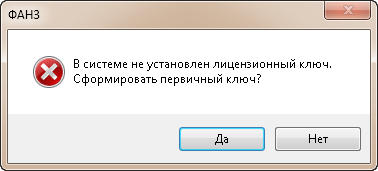


* Первая часть ключа, это зашифрованная *MD5* строка первичного ключа, которая тоже является хэш суммой.
* Вторая часть ключа, это id выбранной версии, которую оплатил клиент. Id сначала шифруется *MD5*, а после хэш кодируется через алгоритм *RSA*.
* Третья часть, это пара открытых ключей, сформированных при кодировании id. С помощью этой пары расшифровывается id и получаем хэш версии. По этому хэшу программа «*включает»* необходимый функционал в программе.
* Все части разделены между собой **“:”**. Чтобы потом было удобно по частям считывать данные и проводить проверку.
* **RSA —** [криптографический алгоритм с открытым ключом](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), основывающийся на [вычислительной сложности](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [задачи факторизации](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB) больших целых чисел.
* **Хэш –** то же, что хэш-функция; число фиксированной длины, которое ставится в соответствие данным произвольной длины таким образом, чтобы вероятность появления различных данных с одинаковым хэшем стремилась к нулю, а восстановить данные по их хэшу было как можно труднее.
* **Валидация** – процедура, дающая высокую степень уверенности в том, что конкретный процесс, метод или система будет последовательно приводить к результатам, отвечающим заранее установленным критериям приемлемости.

**С основными определениями все.**

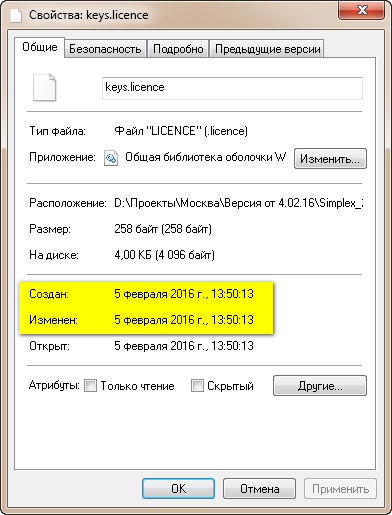
# **Процесс получения первичного ключа и установка лицензионного ключа**

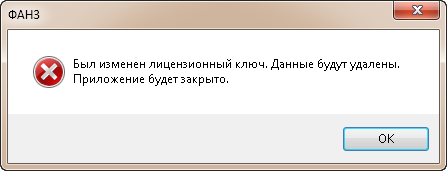
1. **Запускаем программу, получаем такое предупреждение**

****

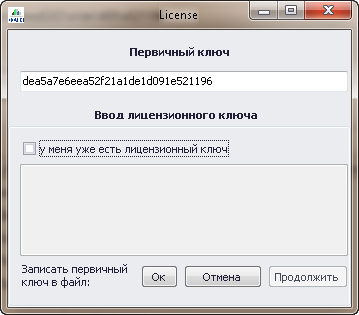
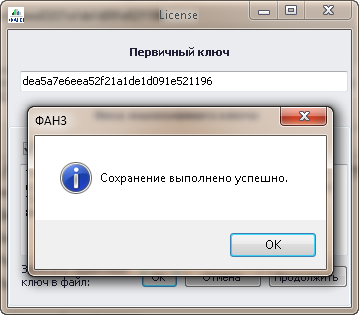
Как бы это не парадоксально звучало, но нужно нажать да, чтобы продолжить, нет, отказаться и выйти из программы. На этом небольшая пауза. Немного о процессе установки ключа.

* 1. При запуске *ФАНЗ*, проверяется наличие самого лицензионного ключа в корне программы ***keys.licence*** и записи в системном реестре Windows, точнее наличии необходимого раздела и записанного в нем ключа …\LocalMachine\Software\Fanz\RegDate.
  2. При отсутствии этих данных запускается форма для формирования ключа, которая сначала выводит сообщение, указанное выше. Т.е., «клиент чист» и никогда себе не ставил *ФАНЗ*.
     1. Если по каким-то причинам в системе остались данные в реестре или в корне ПО уже лежит какой-то лицензионный ключ, то эти данные автоматически удаляются и тут же запускается предложение сформировать первичный ключ. Речь не о жуликах, просто клиент к примеру захотел сменить тип лицензии и оплатил новую. Ну или, да, жулики.
     2. При этом, если и ключ и запись в реестре уже есть, то проверяется значение в ключе ***RegDate*** на предмет равенства дате последнего изменения ***keys.licence***. При установке ключа, автоматически создается раздел и ключ в реестре, в него записывается дата создания ***keys.licence***. Поэтому если файл лицензии пытались изменить, открывали и ковырялись, то windows отражает дату изменения файла. Выглядит это вот так:

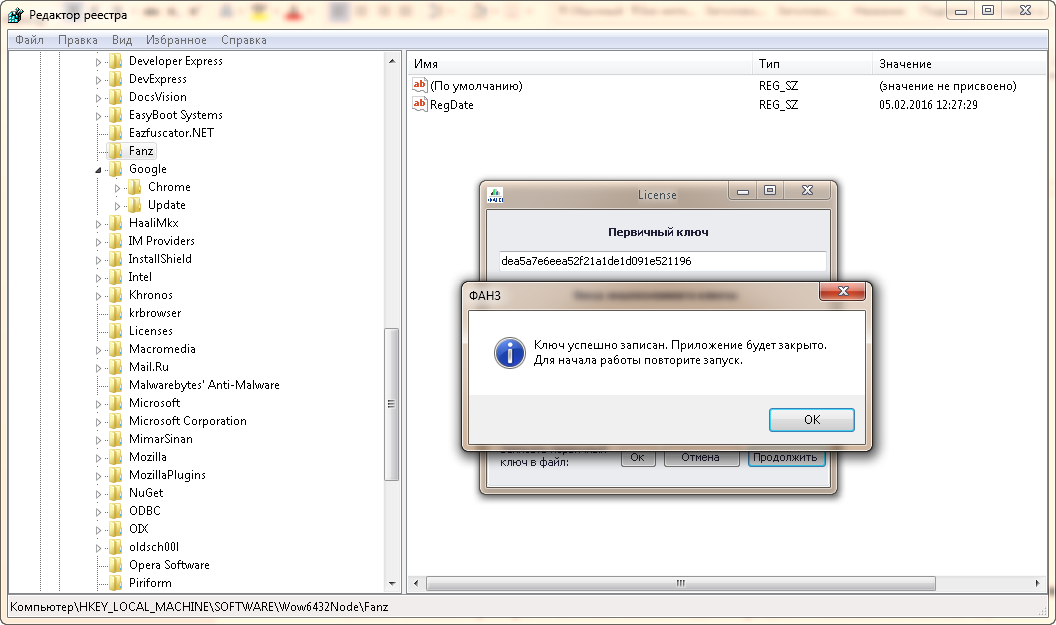
То, что пишется в создан и в измен, без каких-либо изменений, должно совпадать. В реестр попадает первое значение, сравнивается потом со вторым. Если проверка не проходит, то снова удаляются и запись в реестре и файл лицензии, после чего программа закрывается, выдав предупреждение. Но при последующем открытии т.к., лишние данные были удалены, снова стартует предложение сформировать первичный ключ.



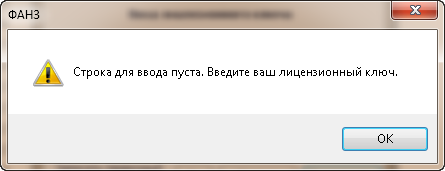
* 1. После предложения сформировать *первичный ключ* переходим на форму *License*. Тут автоматически записывается наш первичный ключ, который мы можем скопировать в буфер обмена или же по кнопке сохранить в текстовый файл ***primary key.txt*** в корне папки. А после этой файл выслать на почту, кому как удобней.

****

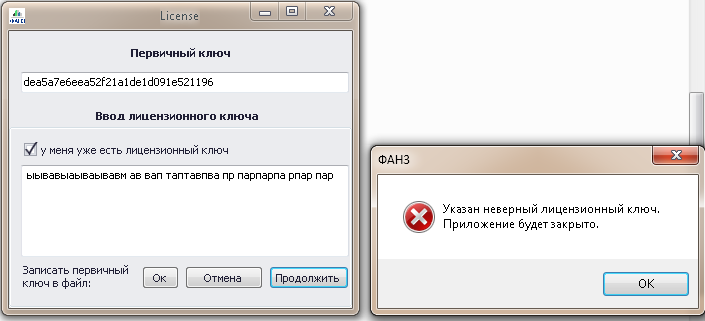
* 1. После того, как выслали первичный ключ, можно закрыть приложение через крестик или кнопкой отмена. Обращаю внимание, что для удобства кнопка продолжить для записи ключа и само поле для ввода ключа неактивны. Чтобы их задействовать нужно выбрать у меня уже есть лицензионный ключ.
  2. После получения ключа, заказчик либо повторно запускает ПО и переходит в это окно, либо вообще не закрывает и ждем ответа по почте, опять же как удобно. После получения *лицензионного ключа,* ставим отметку в чекбоксе, чтобы активировать поле ввода и кнопку. Вводим ключ и нажимаем продолжить. Все просто. Создается ключ лицензии, создается раздел и ключ в реестре. При последующем запуске, после проверки ключа запускается приложении со с необходимым набором функционала. Но, снова но. Рассмотрим, что происходит при нажатии на кнопку продолжить.
     1. Если все проходит без проблем, то выдается сообщение следующего вида:

Тут я правда, сразу открыл редактор реестра, чтобы показать, как это все выглядит.

* + 1. В момент нажатия продолжить срабатывают сразу 2 проверки. Первая, что само поле в целом не пустое, в противном случае предупреждение.

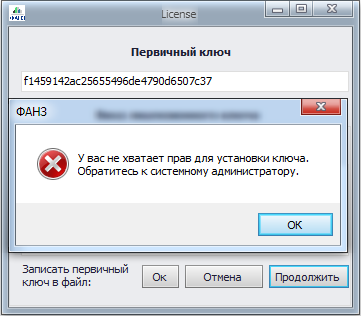


Второй момент, если введенный ключ не соответствует структуре ключа, то выходит другое сообщение.

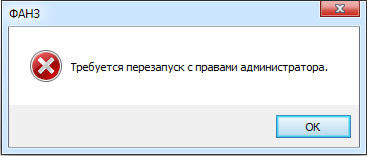
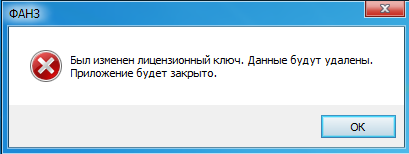


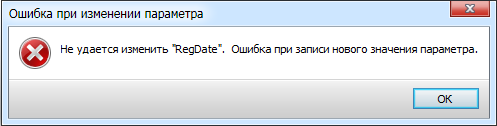
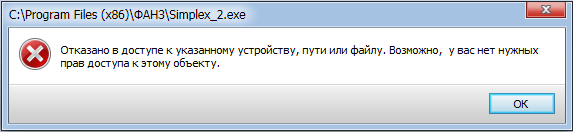
Структура проверяется по такому правилу. Ключ состоит по сути из 3-х частей, проверка смотрит, чтобы в указанном ключе, присутствуют все 3 части, соблюдена последовательность символов, длина первой части, наличие второй и третьей. Если сама структура не нарушена, то происходит запись ключа.

* + 1. Очень важное замечание. Все действия по установке ключа, а также шаги при получении описанных или других ошибок, необходимо выполнять с правами администратора. Объяснение простое, логика программы не может перекрыть правила для пользователей в самой системе windows. Неизвестно кто и где будет ставить программу. У нас 3 операции, которые требуют повышенных прав: это непосредственно сама по себе установка ПО. Так как не всегда разрешено пользователю устанавливать программы в Program Files. Если это корпоративная сеть, то как правило пользователям запрещают устанавливать, изменять программы, вносить какие-то изменения в систему и т.д. Второе, это да, удаление. И третье, что простым пользователям запрещается – это работа с системным реестром. Т.е. создание, запись ключей и тем более удаление считаются опасными операциями и для не администраторов, это по умолчанию закрыто. Для некоторых запрещено даже просто его открывать на чтение, поскольку изменять реестр может быть катастрофично. Я когда тестировал постарался обработать 2 случая, когда бы хотим установить ключ и пробуем без прав администратора, то получаем сообщение:



И второй случай, когда необходимо удалить записи в реестре или удалить ключ лицензии. Т.е., это когда срабатывают проверки, описанные в самом начале.





В целом, по описанию процесса получения первичного ключа и установки лицензионного все.

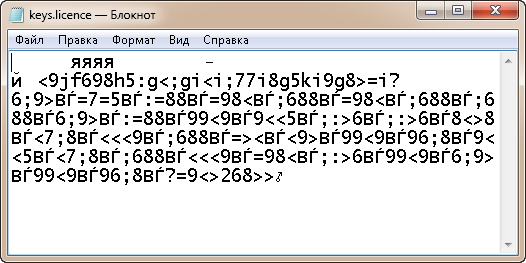
Основные тезисы:

1. Все административные операции по установке, удалению и т.д. выполнять с правами администратора или с расширенными правами для простого пользователя, которые позволяют вносить изменения в реестр и работать с папками и файлам в служебных папках Program Files / Program Files (x86).
2. С момента запуска приложения проводятся многочисленные проверки, начиная от проверки системы, до непосредственно самой установки ключа, операций с реестром и файлом ключа.
3. Обработаны пользовательские способы обхода проверок и максимально продуманы возможные варианты событий на клиентском компьютере.

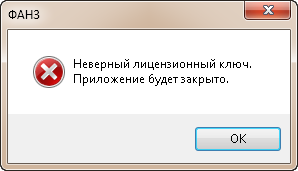
# **Проверка лицензионного ключа при старте приложения**

Если выполнены базовые проверки на предмет изменения данных в реестре и в самом ключе, то при запуске проверяется уже непосредственно сам ключ, который записывается в файл keys.

1. Опустил во второй части следующий момент. При создании ***keys.licence*** в программе видоизменяется введенный код, через мой небольшой алгоритм шифрования, а после эта закодированная текстовая информация сериализуется. Это значит, что данные записываются в бинарный формат. Если ключ именно попробовать открыть через любую программа windows, самое частое это блокнот, поскольку, когда внутри всякое безобразие, он не ругается, то выйдет что-то подобное этому:

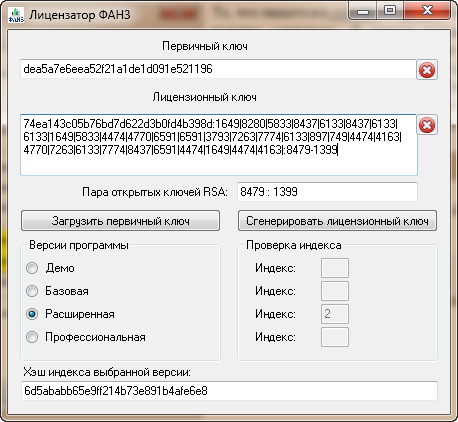


1. Для чего это сделано. Опять-таки если смотреть действия пользователя, то не надо быть программистом, чтобы при необходимости изменить даты создания, изменения и открытия файлов. Есть для этого софт. Поэтому если явно записать данные и оставить простую возможность изменить ключ, то при изменении содержимого и временных отметок, а также редактирование реестра, даже простой пользователь сможет обойти проверки.
2. Поэтому есть такой барьер. И вот когда запускается программа, проходят базовые проверки, содержимое десериализуется, полученные данные дешифруются к исходному виду, то полученный ключ также условно разбивается на три части и каждая начинает проверяться.
   1. Первая часть, как мы помним, это хэшированный хэш полученного *первичного ключа*. Для проверки в этот момент программа считывает данные для *первичного ключа*, шифрует и сравнивает полученный хэш с тем, который указан в ключе. Малейшее несовпадение, хэш полностью меняется, а это значит, что данные изменились. Т.к., ключ привязывается к компьютеру, то при ошибке на этом этапе можно сказать, что компьютер скорее всего другой. Или новый пользователь, или новое железо. Да, тут есть очень тонкий момент, что и железо можно поменять и пользователя сменить и вирус словить и т.д., но это можно решать на уровне поддержки и в случае таких обращений принимать решение, на основе нового *первичного ключа* за деньги высылать новый ключ или же бесплатно. Согласен, что не самая эффективная штука, но зато эффективно блокирует самые очевидные махинации, чтобы не покупать лицензии. Вдаваться в подробности содержимого первичного ключа не нужно, чтобы не было желания воспроизвести, но донести до понимания общей схемы о последствиях нужно. Более продвинутые защиты требуют использование других технологий, а это и другие цены. Тут не вызывает особого удивления цена на программу всего в 20-30 тысяч и полмиллиона на лицензии на 50 пользователей к примеру, 100 тысяч на внедрение и по 30 тысяч на ежемесячное обслуживание.
   2. Потом считывается пара открытых ключей и с их помощью расшифровывается вторая часть ключа, которая закодирована через *RSA*. Если все эти операции, проходят, то полученный хэш типа версии передается на запуск программе. Этот хэш проходит валидацию, то есть программа сравнивает этот хэш, с тем, что хранится в списке хэшей всех версий. Если все верно, то запускается окно авторизации, а дальше как обычно, только функционал уже будет соответствующий. Если ключ не проходит проверку на этапе считывания или уже валидации, то выскочит ошибка, приложение закроется.

Тут заканчивается все проверки ключей, единственное, как и ранее в демо-версии, которая сейчас на сайте, если лицензия демо, то на форме splash screen будет выведено сообщение

«Внимание! Используется демо-версия…»

# **Использование лицензиатора**

Так выглядит сам лицензиатор.

1. В первичный ключ попадают полученные данные из буфера обмена или при загрузке текстового файла с данными. Вспоминаем, что клиент сразу может выгрузить ключ в готовый текстовый файл.
2. Выбираем версию программы и нажимаем сгенерировать. Все просто. Запишется лицензионный ключ, также выведется хэш индекса выбранной версии.
3. Т.е., тут загружаем первичный ключ, версию и строим лицензионный ключ. Хэш выбранной версии кодируется алгоритмом *RSA*. Расшифровать без взлома весьма проблематично, а так получить исходную строку можно только с указанной парой открытых ключей. Прелесть открытых ключей в том, что даже зная алгоритм и ключи расшифровать достаточно трудно. Чем сложнее открытые ключи, тем выше криптостойкость для расшифровки. Вот именно в этом месте у меня один раз дал алгоритм осечку. Я получил пару простых чисел от 0 до 100. Один ключ был двухзначным, а второй однозначным, в итоге такой ключ расшифровал только частично код и программа его не пропустила, что логично, коль сам так задумал. Поэтому, после многочисленных тестов по выявлению зависимости и поиска слабого места, я чуть-чуть изменил алгоритм, чтобы меньше 100 ключей не было. Т.е., трехзначные ключи можно смело использовать. И вот обещанная рекомендация, код кодом, но все-таки если генератор выдаст двухзначные ключи или наоборот многомилионные, то просто перегенерировать и все. При миллионных значениях тоже некорректно расшифровывает. Хотя после внесения изменений в алгоритм у меня таких значений больше не появлялось.
4. Саму проверку лицензионного ключа уже описал, тут больше нечего особо добавить.

# **Модуль версий ФАНЗ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможности | Демо-версия | Базовая  версия | Расширенная версия | Профессиональная версия |
|  |  |  |  |
| Расчетный модуль. Запуск алгоритма симплекс М-метода. | - | + | + | + |
| Проверка показателей (логический контроль). Корректировка решения | - | + | + | + |
| Расчет общего отклонения | - | + | + | + |
| Построение графиков 2D | - | + | + | + |
| Возможность построения графиков 3D и экспорт графиков (пункт 2 развития) | - | - | + | + |
| Прогнозирование методом МНК | - | - | + | + |
| Возможность выбора типа диаграммы для прогнозирования (пункт 1 развития). | - | - | + | + |
| Возможность выбора метода прогнозирования. Например, построение полинома при наборе точек более 5. | - | - | - | + |
| Создание блока отчетов (пункт 5 развития) | - | - | - | + |
| Конструктор задачи (пункт 6 развития) выбор метода решения задачи. Например, помимо М-метода реализация численных методов | - | - | - | + |

В соответствии с этой табличкой и настройками, выполненных в версии 1.1.1, построил систему со всеми ограничениями, которые в зависимости от типа внедряются либо при загрузке форм, либо при нажатии на доступные кнопки. Также в версиях, начиная с базовой часть кнопок становятся доступные в ходе решения, но изначально отключены. Дальше предоставлен перечень. Может что-то и не отразил, я по ходу тестов и проверок еще в коде правил, мог забыть отразить в списках, но тем не менее.

1. **Демоверсия.** Отключение при загрузке на главной форме и в форме прогнозирования.

**Главная форма**

Сохранить показатели, очистить все данные, проверка показателей, корректировка решения, вернуть в работу, рассчитать на вкладке результаты расчетов, рассчитать на вкладке отклонения от идеального ф. с., запуск алгоритма решения выводит предупреждающее сообщение, что используется демо-версия.

**Графика и печать**

Предпросмотр диаграммы показателей, предпросмотр диаграммы отклонений, предпросмотр отклонения от идеального фин. состояния, в 3d, в 2d, график отклонения от идеального фин. состояния, диаграмма по показателям, диаграмма по отклонениям, экспорт в xlsx, экспорт в png, экспорт в jpeg, экспорт в XLSX, экспорт в XLS, экспорт в PDF, экспорт в PNG, экспорт в JPG, экспорт в xls, экспорт в pdf.

**Диаграмма радар**

Диаграммы, стили, мастер, шаблоны, печать и экспорт.

**Модуль прогнозирования**

Недоступны все контролы и вкладки. Доступна только справка.

1. **Базовая версия.** Отключение при загрузке на главной форме и в форме прогнозирования, обработка при нажатии кнопок в ходе решения.

**Главная форма**

Сохранить показатели, очистить все данные, проверка показателей, запуск алгоритма решения, корректировка решения, вернуть в работу, кнопка решить в исследовании Гауссом (доступно только администратору), решение М-задачи на вкладке Решение М-задачи (доступно только администратору), рассчитать на вкладке результаты расчетов, рассчитать на вкладке отклонения от идеального ф. с.

**Графика и печать**

В 3d, печать (потом будет доступно при смене статуса), просмотр (потом будет доступно при смене статуса), экспорт в xlsx, экспорт в png, экспорт в jpeg, экспорт в XLSX, экспорт в XLS, экспорт в PDF, экспорт в PNG, экспорт в JPG, экспорт в xls, экспорт в pdf.

**Диаграмма радар**

Диаграммы, стили, мастер, шаблоны, печать и экспорт.

**Модуль прогнозирования**

Недоступны все контролы и вкладки. Доступна только справка.

1. **Расширенная версия.** Отключение при загрузке на главной форме и в форме прогнозирования, обработка при нажатии кнопок в ходе решения.

**Главная форма**

Сохранить показатели, очистить все данные, проверка показателей, запуск алгоритма решения, корректировка решения, вернуть в работу, кнопка решить в исследовании Гауссом (доступно только администратору), решение М-задачи на вкладке Решение М-задачи (доступно только администратору), рассчитать на вкладке результаты расчетов, рассчитать на вкладке отклонения от идеального ф. с.

**Графика и печать**

Печать (потом будет доступно при смене статуса), просмотр (потом будет доступно при смене статуса).

**Диаграмма радар[[1]](#footnote-1)**

Мастер, шаблоны, печать и экспорт.

**==Модуль прогнозирования==**

Мастер, шаблоны, печать и экспорт, выбор линии тренда, доступно только выбор линейного тренда. Хотя у нас и заявлено, что выбор метода прогнозирования недоступен, но учитывая, что доступен сам МКН, выбор типа диаграммы, автоматически и стили, то полный запрет на выбор метода не имеет смысла, тогда либо все закрывать, как в демо и базовой версиях, либо предлагаю по умолчанию оставить только линейный тип и все, как я сейчас и сделал. И на нем уже менять тип диаграммы и т.д. Тогда все логично. В ограниченном режиме модуль прогнозирования будет функционировать. Выглядит это примерно следующим образом (скриншот на следующей странице).

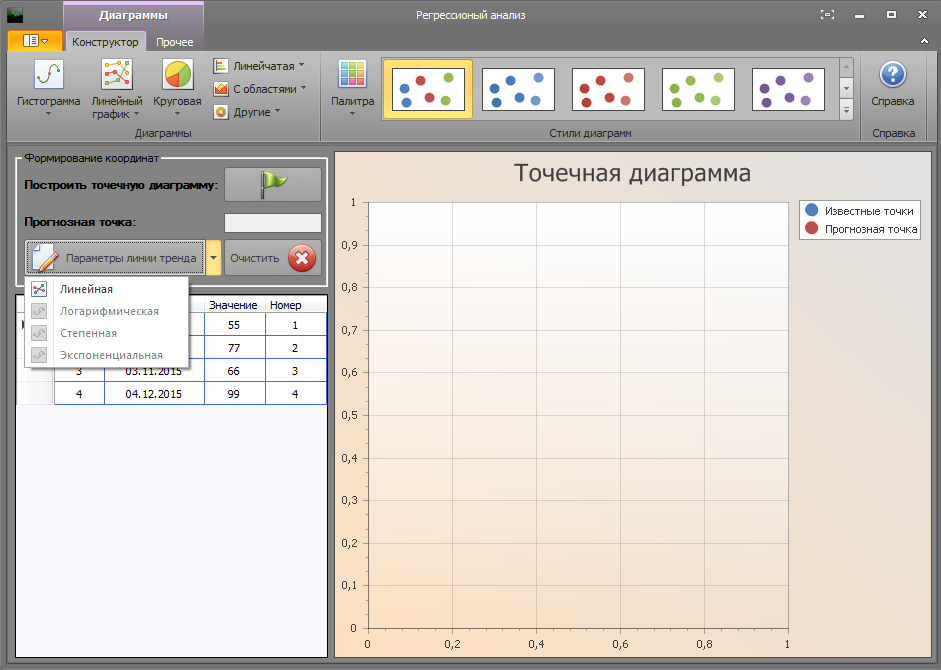
1. **Профессиональная.** В особом представлении не нуждается. Это все механизмы, которые были сделаны до ввода версий.

**Главная форма**

Сохранить показатели, очистить все данные, проверка показателей, запуск алгоритма решения, корректировка решения, вернуть в работу, кнопка решить в исследовании Гауссом (доступно только администратору), решение М-задачи на вкладке Решение М-задачи (доступно только администратору), рассчитать на вкладке результаты расчетов, рассчитать на вкладке отклонения от идеального ф. с.

**Графика и печать==**

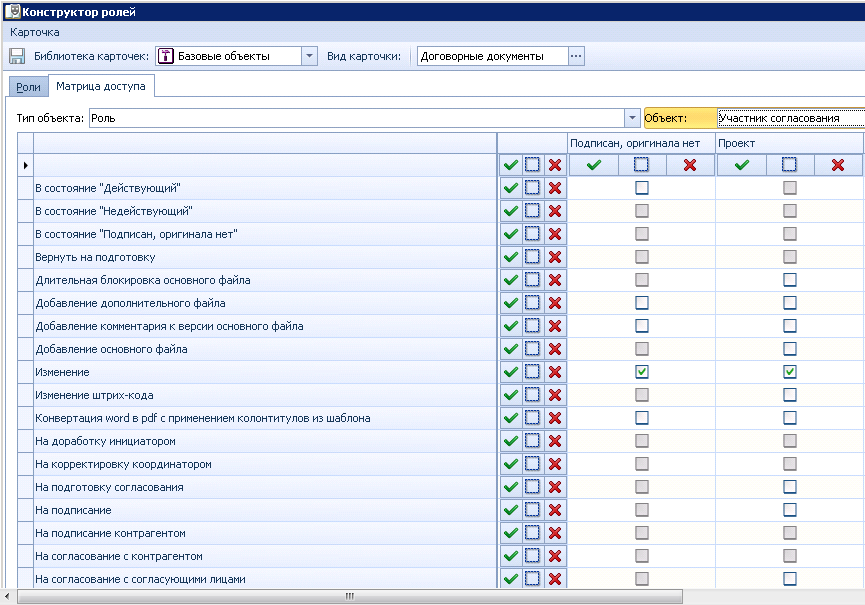
Печать (потом будет доступно при смене статуса), просмотр (потом будет доступно при смене статуса).



А все, что связано с запуском мастера, продвинутому экспортированию и остальным типа линии тренда отключено.

Таким образом все, что указывается в таблице выполняется. Просто не все пункты полноценно дают интерпретацию элементов, т. к, эта табличка делалась еще летом, когда была самая первая версия программы. Может имеет смысл ее уточнить.

В качестве примера, вот скриншот с конструктора ролей в нашем docsvision.



Т.е., структурировано выводит доступ к операциям, в зависимости от состояния объекта и роли в системе сотрудника.

Не затрагивая расчетного модуля, по сути основные ограничения были на вкладках с экспортом и импортом диаграммы радар, самими вкладками конструктора и по такому же сценарию с окном прогнозирования. Ну и также по необходимости обработка ограничений при нажатии кнопок. Это базовые вещи, такие как загрузка данных, логический контроль, запуск решения. На них у нас меняются статусы, меняются доступность других кнопок. Более подробно все можно будет вживую посмотреть, иначе скриншотов не хватит.

По часть дальнейшей разработки. Теперь сам модуль вынесен отдельно и в нем сделаны: классификация типов объектов, которые подвергаются ограничениям, методы для управления этими объектами и необходимые все функции, в самой программе заданы переключатели для работы с модулем. Соответственно, при добавлении новых модулей отразить их в версиях не потребует много времени, по сути просто в классификации типов объектов добавить дополнительные объекты, все методы и структура работы в программе не меняются.

Можно даже чуть проще сказать, в dll версий заданы строго типизированные списки объектов, а для управления этими списками и самими объектами написаны методы.

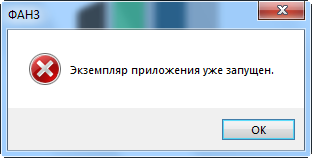
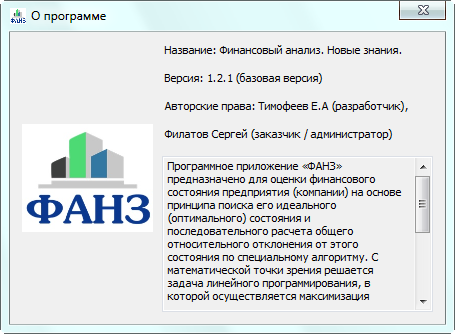
# **Дополнительно**

Как уже оговаривалось ранее в почте, переделал маленько форму о программе, нашел одну ошибку (точнее получение ошибки, если неверно работать), добавил ряд новых проверок для перехвата и заменил по всей программе название окошек с предупреждениями, ошибками и т.д. Система, уведомление и system заменил везде на ФАНЗ.

Также для удобства пользователя добавил проверку запуска количества копий приложения. Чтобы больше одного экземпляра не запускалось. Просто бывает программу запускаешь, а не сразу появляется окно программы, пользователь может просто еще раз тыкнуть по иконке, в итоге будет грузиться обе копии, что неэффективно, т.к., общее время загрузки увеличивается. Если программа уже запущена, то выходит соответствующее сообщение.

Кроме этого внес несколько корректив в форму о программе, чтобы было единство оформления и выводилась актуальная информация. Заменил логотип, убрал лишние строки в таблице, вывел актуальную версию программы, а также как формы программы реализуют функционал вывел названия типа лицензии.

Вот так выглядят сообщение и обновленная форма.



# **Схема взаимосвязей объектов**

На схеме ниже обозначено как устроена программа. Полностью. Она как раз отображает те расширения, которые приходилось разрабатывать, выносить код и снова дорабатывать. В целом, ранее я уже пояснял как сейчас работает программа. Поэтому привожу полную и обновленную схему. Ниже приведено краткое пояснение по объектам.

1. **RSA.dll.** Методы шифрования по алгоритму с открытым ключам *RSA*. Применяется в лицензиаторе.
2. **RSAdelib.dll.** Метод для расшифровки строки по полученному открытому ключу (паре ключей). Применяется при проверке лицензионного ключа.
3. **Lib.dll.** Методы для дополнительного шифрования (по сути запутывания) и расшифровки лицензионного ключа. Применяется при записи лицензионного ключа в систему, а также при проверке.
4. **Kernel.dll.** Вынесены из *ФАНЗ* все методы по загрузке базы с начальной матрицей, а также все методы по работе с данными xml. Используется в библиотеках п.5 и п.6, а также непосредственно в *ФАНЗ*.
5. **Envention.dll.** Вынесены из *ФАНЗ* все методы для работы с базовым Симплекс методом и М-задачей. Содержит основные функции решения, использует *kernel.dll*. Используется *ФАНЗ*.
6. **Edition.dll.** Модуль определения версий, использует *kernel.dll*. Используется *ФАНЗ*.
7. **SplashScreen1.** Самое первое, что загружает ПО. Тут добавлена новая форма **License**, которая возникает для формирования первичного и установки лицензионного ключей. В **SplashScreen1** проходит вся логическая система всевозможных проверок. Вот то, что на схеме в фигурных скобках, по сути вся эта обработка происходит тут.

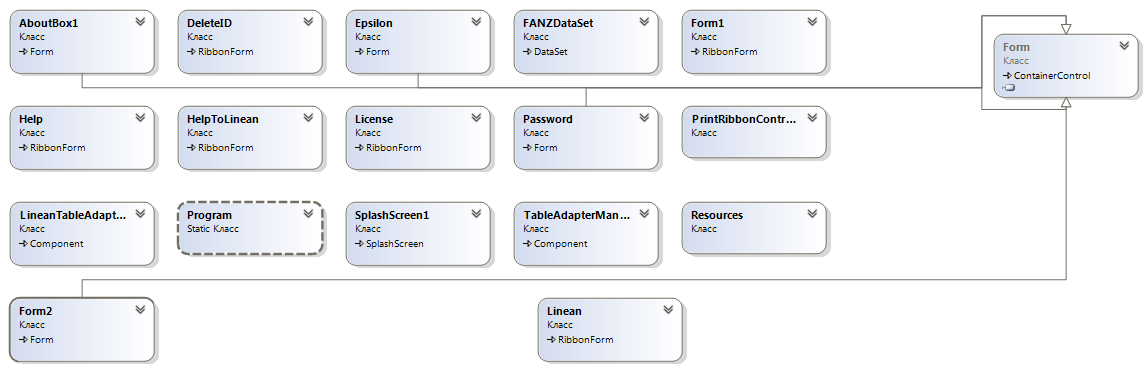


# C:\Users\Timofeevs\Desktop\DLL File.pngC:\Users\Timofeevs\Desktop\exe-icon.png

**RSA.dll**

**lib.dll**

**RSAdelib.dll**



Дешифровка

Лицензии

Проверки

Шифрование

**Edition.dll**

**Лицензиатор**

**ФАНЗ**

**Kernel.dll**

**Envention.dll**

1. Открыл первую вкладку конструктора с выбором типа диаграмм и стилей. Т.к., такой же функционал доступен в МНК. [↑](#footnote-ref-1)