МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Систем и Технологий

Кафедра «Информационные системы»

Дисциплина «Информационная безопасность»

**ОТЧЕТ**

**по лаборатороной работе № 1**

Тема Разработка программы разграничения полномочий пользователей на основе парольной аутентификации c использованием встроенных криптопровайдеров

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_Кузнецов В. И.\_\_\_\_\_\_\_/

подпись инициалы, фамилия

Курс\_\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа\_\_\_\_\_\_\_ПИбд-41\_\_\_\_\_\_\_\_\_\_

Направление/специальность\_\_\_09.03.04. Программная инженерия\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Мытарев П.В.

должность, ученая степень, ученое звание

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Дата сдачи:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Дата защиты:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_\_г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

2020 г.

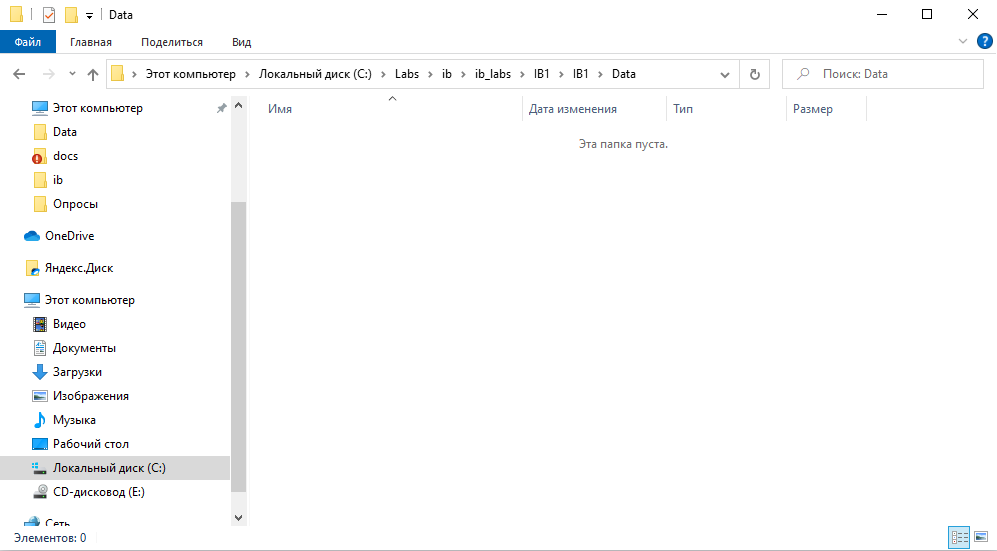
**Задание по варианту 12**

Ограничения на пароль: Наличие строчных и прописных букв, а также цифр.

Шифрование: DES с режимом OFB, без случайного значения.

Хеширование: SHA без случайного значения.

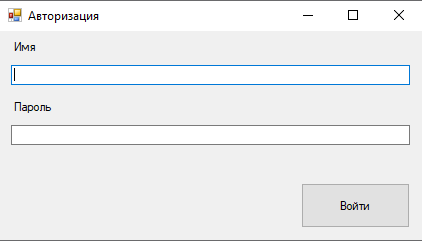
**Решение:**



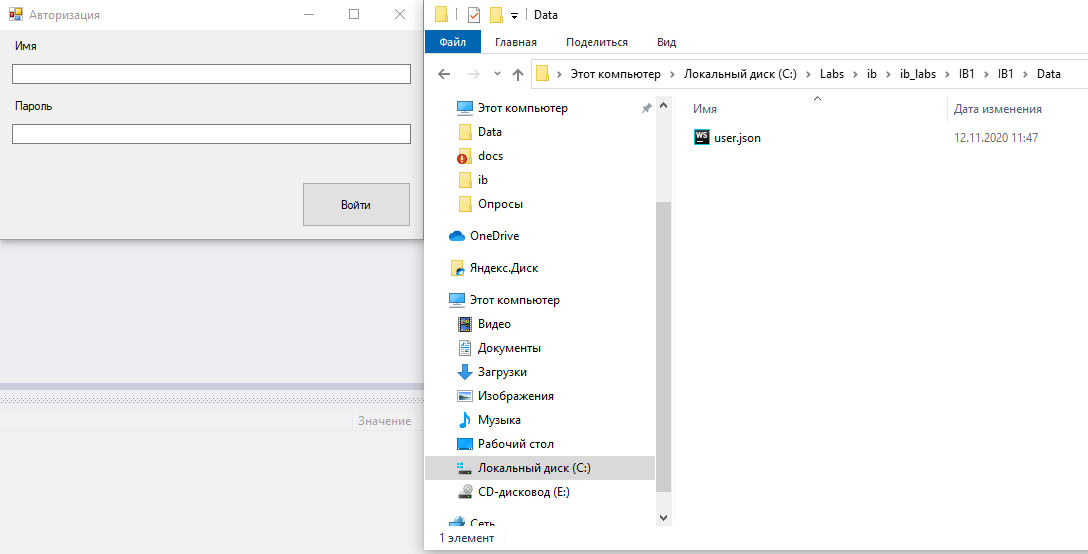
*Рис. 1*

Комментарий к рис.1

Перед первым запуском, файл содержащий информацию о пользователях отсутствует.



*Рис. 2*

**

*Рис. 3*

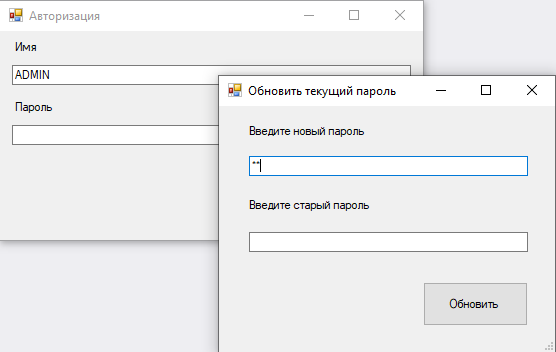
Комментарий к рис.2

При запуске загружается экран авторизации в приложении, содержащий поля имени пользователя и пароля.

Комментарий к рис.3

При первом запуске приложения, создаётся файл, содержащий информацию о пользователях, в котором сначала содержится только учетная запись только админа с пустым паролем, установленным флагом на ограничения для пароля и флагом того, что пользователь не является заблокированным.

Все последующие запуски приложения сопровождаются проверкой расшифрованного пользователя ADMIN, в противном случае программа будет завершаться с ошибкой.

**

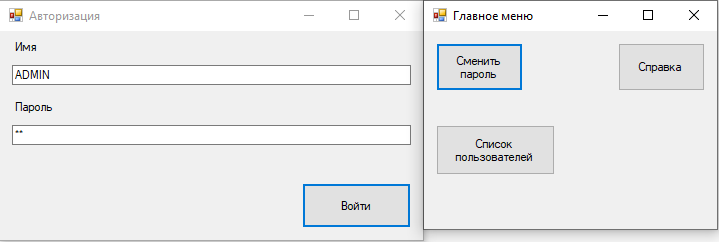
*Рис. 4*

Комментарий к рис.4

При первом входе в приложение, пароль у пользователя пустой, поэтому при нажатии на кнопку войти запускается окно смены пароля.

Логика смены пароля: новый пароль должен либо отвечать заданным ограничениям, либо не должен, в зависимости оттого были ли установлены ограничения для учётной записи или нет (в данном случае, так как первых вход осуществляет админ, пароль должен отвечать ограничениям). Смена пароля будет отклонена, в случае:

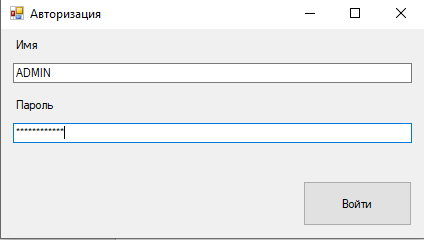
* если хеш введённого старого пароля не будет идентичен хешу того пароля, который указан в файле учетных записей
* если введенный старый пароль, будет идентичен введенному новому паролю (в нормальном представлении)



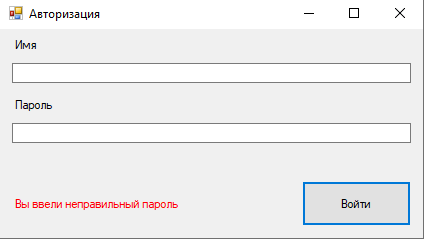
*Рис. 5*

Комментарий к рис.5

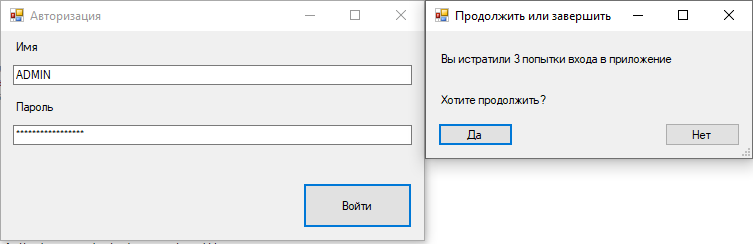
Успешная авторизация пользователя, после смены пароля при первом входе.



*Рис. 6*



*Рис. 7*

**

*Рис. 8*

Комментарий к рис.6

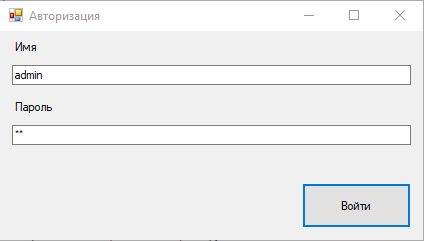
Ввод неправильного пароля

Комментарий к рис.7

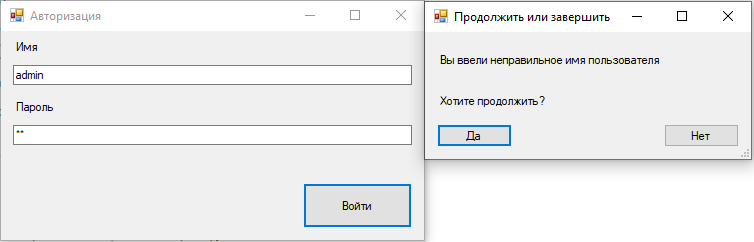
В случае если, хеш введённого пароля, не идентичен хешу пароля, записанного в файл, приложение инкрементирует данную попытку входа, выводит ошибку и очищает поля ввода.

Комментарий к рис.8

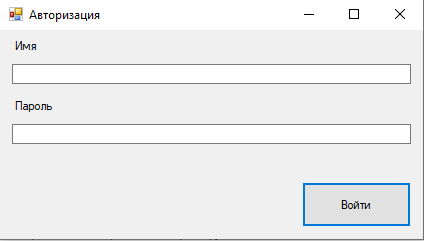
Если счётчик попыток входа стал равен 3, то приложение запускает экран с выводом ошибки и предлагает завершить работу с приложением. В случае если пользователь продолжит работу с приложением, последующие попытки входа не будут выполняться, т.к. при значении счётчика равным 3, метод отвечающий за авторизацию выбрасывает Exception, что приводит к повторному запуску экрана с ошибкой.



*Рис. 9*

**

*Рис. 10*

**

*Рис. 11*

Комментарий к рис.9

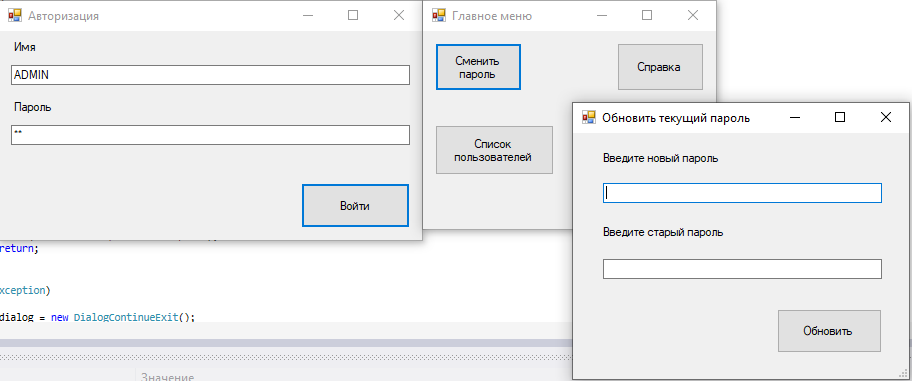
Ввод неправильного имени пользователя

Комментарий к рис.10

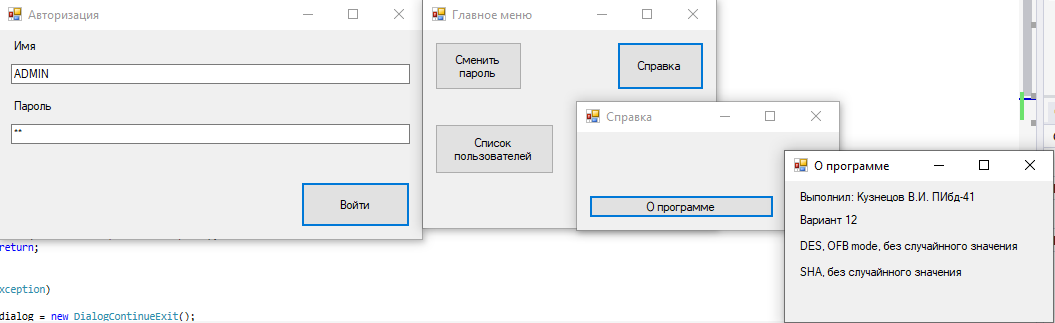
В случае если пользователь с введённым именем отсутствует в файле, приложение запускает экран с выводом ошибки и предлагает завершить работу с приложением, либо продолжить.

Комментарий к рис.11

В случае если на экране вывода ошибки был выбран вариант продолжить работу, на экране авторизации очищаются поля ввода и можно выполнить вход снова. Ограничения на кол-во попыток входа, в таком случае, не устанавливаются.



*Рис. 12*

**

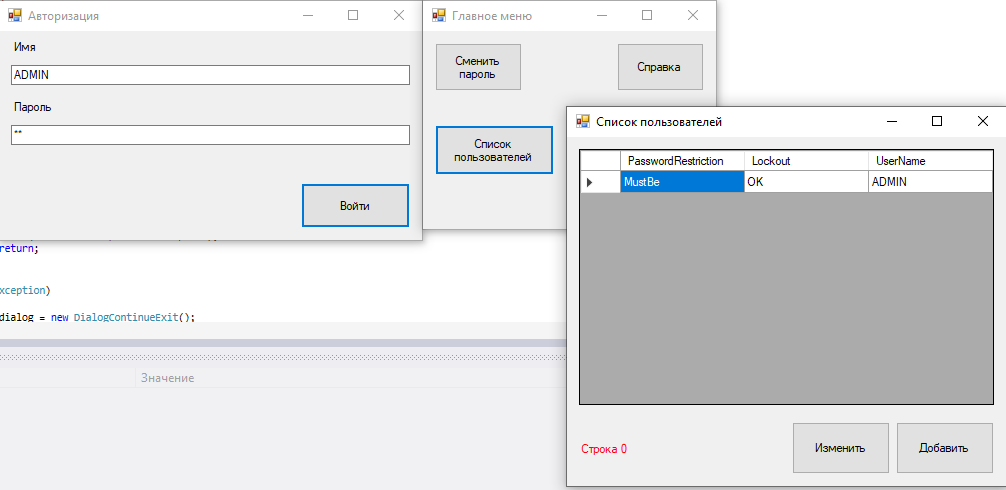
*Рис. 13*

Комментарий к рис.12

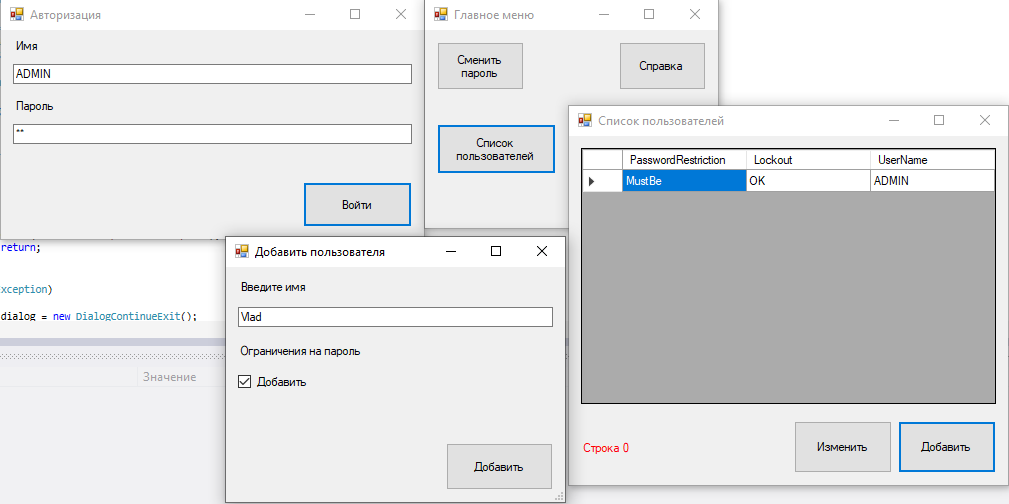
Функционал доступный всем пользователям: смена пароля (логика была описана выше в комментарии к рис. 4)

Комментарий к рис.13

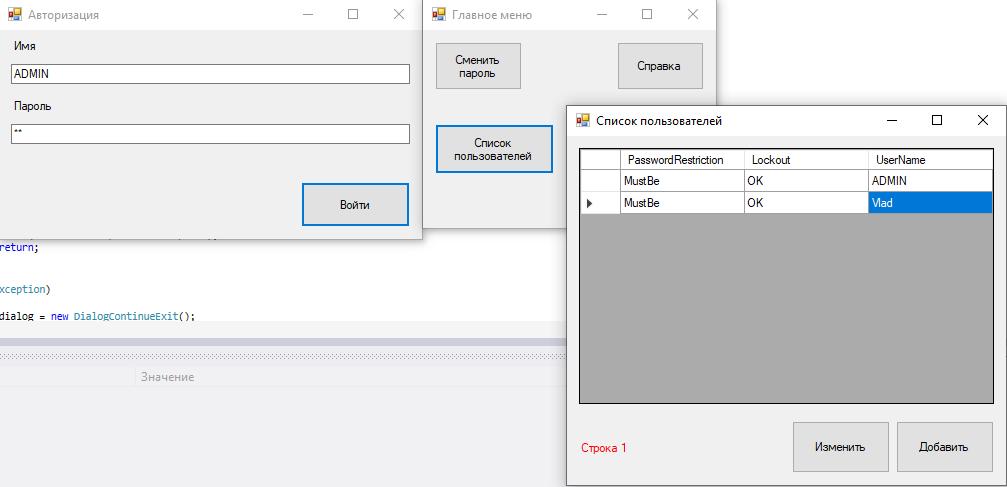
Функционал доступный всем пользователям: окно справки, содержащее кнопку просмотра информации о программе.



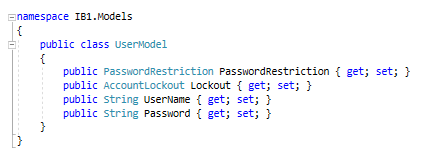
*Рис. 14*



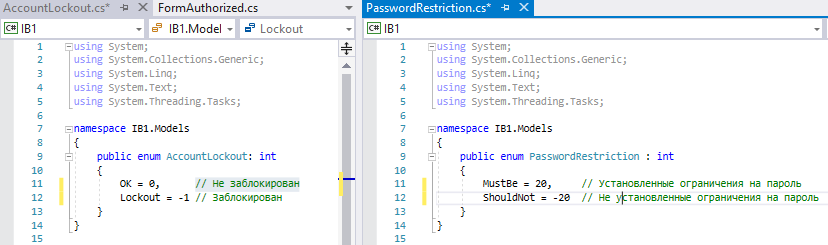
*Рис. 15*

**

*Рис. 16*

**

*Рис. 17*

**

*Рис. 18*

Комментарий к рис.14

Функционал доступный только администратору: просмотр списка зарегистрированных пользователей

* Столбец PasswordRestriction: показывает наличие ограничений на пароль
* Столбец Lockout: показывает заблокирован пользователь или нет
* Столбец UserName: показывает имя пользователя

Комментарий к рис.15

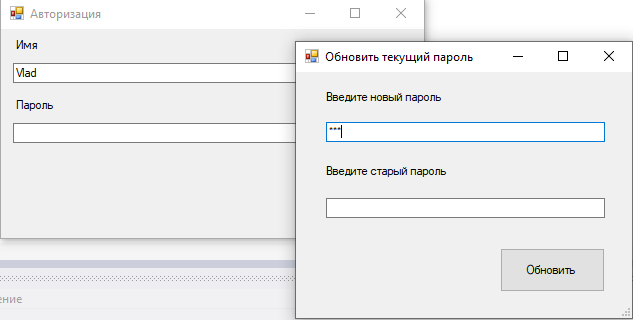
Функционал доступный только администратору: добавление нового пользователя с указанием имени пользователя, наличием ограничений на пароль и пустым паролем.

Комментарий к рис.16

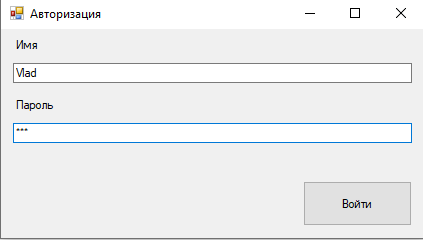
Обновленный список пользователей, после добавления администратором, нового пользователя.

Комментарий к рис.17, 18

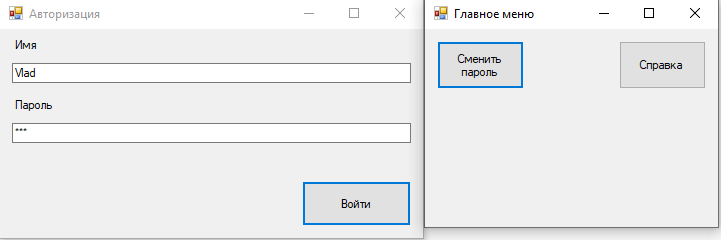
Описание модели пользователя и перечислений для соответствующих свойств.



*Рис. 19*

**

*Рис. 20*

**

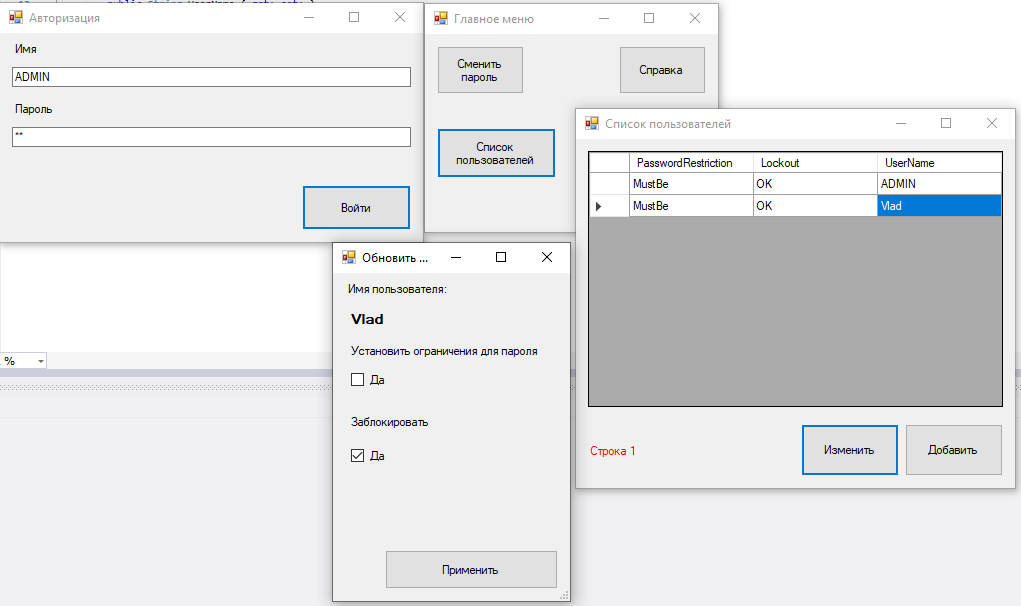
*Рис. 21*

Комментарий к рис.19, 20

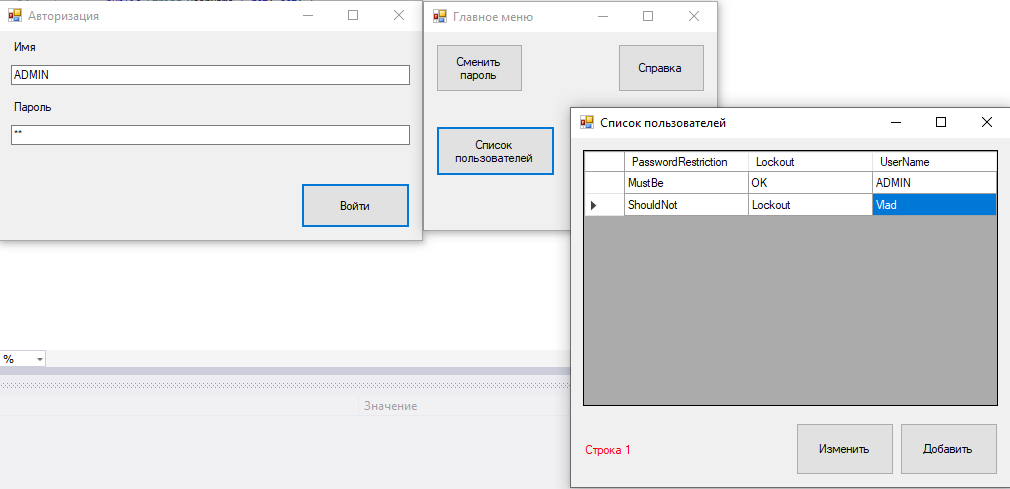
Первый вход в приложение обычного пользователя. Логика была описана выше в комментарии к рис. 4.

Комментарий к рис.21

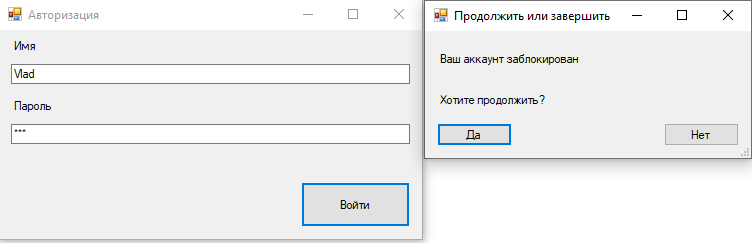
Окно приложения для обычного пользователя. Подробное описание было изложено в комментариях к рис. 12,13



*Рис. 22*

**

*Рис. 23*

**

*Рис. 24*

Комментарий к рис.22

Блокировка администратором пользователя с именем Vlad и отмена ограничений на новый пароль для этого пользователя.

Комментарий к рис.23

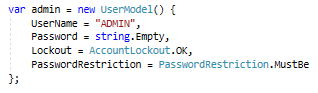
Обновленный список пользователей, после изменений администратором свойств пользователя с именем Vlad.

Комментарий к рис.24

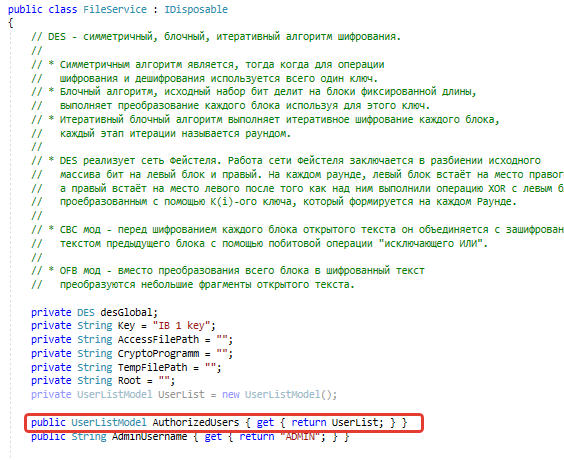
Заблокированный пользователь, не может войти в приложение, поэтому запускается экран с выводом ошибки и предложением завершить работу с приложением.

**Комментарии к самому важному функционалу приложения.**

При первом запуске приложения, создаётся файл user.json, в который будет сохранен в зашифрованном виде самый первый пользователь – администратор (Рис.1). Содержимое сохраняется в памяти приложения в виде массива дессериализованных объектов UserModel (Рис. 2).



*Рис. 1*

**

*Рис. 2*

Шифрование и запись в файл осуществляется с помощью приложения написанного на языке Golang, так как стандартный функционал .Net не реализует мод (режим) OFB для алгоритма DES, поэтому мною был выбран кратчайший путь решения этой проблемы. Я не стал искать криптографические библиотеки для C# и разбираться с их документацией (которая у половины из них попросту отсутствует) и принял решение написать приложение на языке Golang, так как этот язык я используется для работы и повседневного решения своих задач, а также потому что в его стандартной библиотеке уже реализован OFB мод для алгоритма DES.

C# приложение запускает процесс main.exe (Рис. 3), который лежит в папке проекта, создает временный файл для передачи (Сериализация) и получения (Дессериализация) расшифрованных данных.

Данные во временном файл представляются в формате строки валидной для json:

{

"Users":[

{

"Lockout":0,

"UserName":"ADMIN",

"Password":"",

"Lockot":"0",

"PasswordRestriction":"20"

}

]

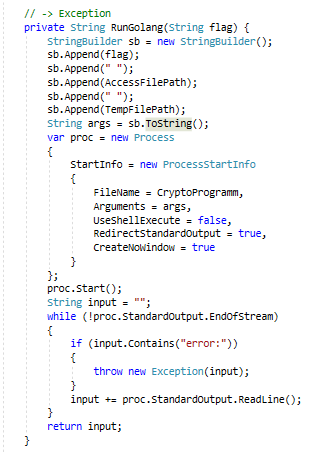
}

Golang приложению в зависимости от того, что ему необходимо сделать, передается флаг:

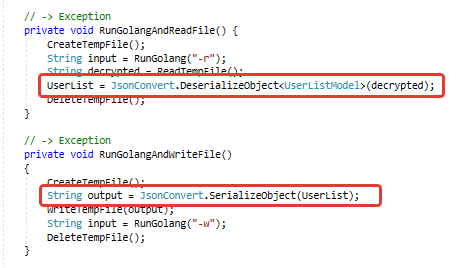
* –r - чтение из основного файла, расшифровка данных и записи расшифрованных данных во временный файл
* -w – запись в основной файл, чтение расшифрованных данных из временного файла, шифрование данных и запись зашифрованных данных в основной файл.

И два обязательных аргумента: путь до основного файла и путь до временного файла.

После завершения работы Golang приложения, C# приложение удаляет временный файл.

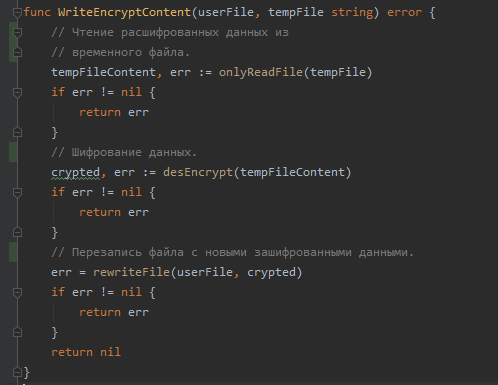


*Рис 3.*

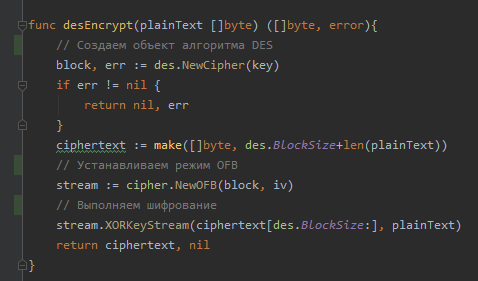


*Рис 4.*

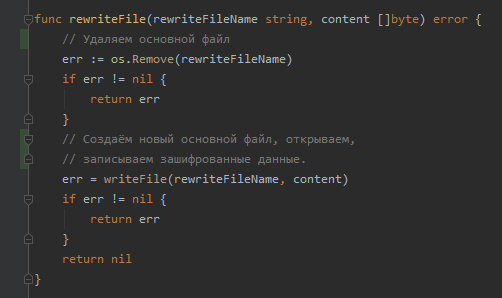
**Реализация DES в Golang:**



*Рис. 5*

**

*Рис. 6*

**

*Рис. 7*

Комментарий рис. 5:

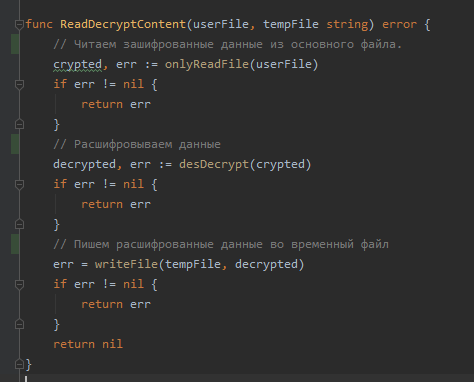
Перезапись файла с новыми зашифрованными данными.

Комментарий рис. 6:

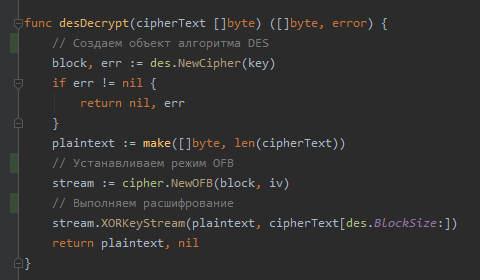
Зашифровываем данные с помощью DES в режиме OFB.

Комментарий рис. 7:

Удаляем основной файл, создаем его заново и записываем в него зашифрованные данные.



*Рис. 8*



*Рис. 9*

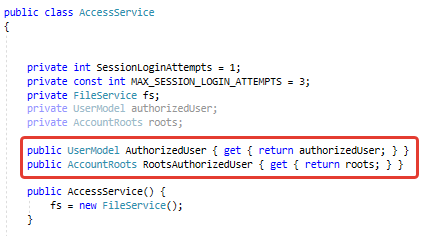
Комментарий рис. 8:

Читаем данные из основного файла, расшифровываем и записываем во временный файл.

Комментарий рис. 9:

Расшифровываем данные с помощью DES в режиме OFB.

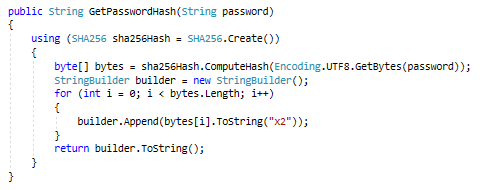
**Разграничение прав пользователей. Ограничения пароля. Хэширование.**

****

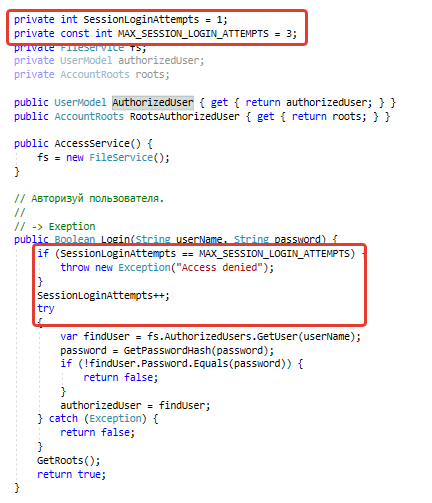
*Рис. 10*

**

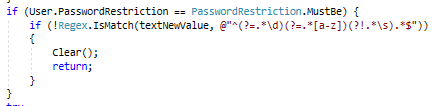
*Рис. 11*

**

*Рис. 12*

**

*Рис. 13*

**

*Рис. 14*

Комментарии к рис. 10, 11:

AccessService: Для работы с сущностью пользователь предусмотрен класс AccessService, который выполняет все возможный операции с пользователем: Авторизация, регистрация, смен пароля и т.д.

AccessService: Этот класс хранит в себе поля RootsAuthorizedUser определяющий уровень прав авторизованного пользователя и объект самого пользователя AuthorizedUser, поэтому в его методах всегда проверяется уровень доступа пользователя.

Комментарии к рис. 12:

AccessService: Реализация SHA для получения дайжеста (хеша) пароля.

Комментарии к рис. 13:

AccessService: Реализация ограничения попыток входа.

Комментарии к рис. 14:

DialogUpdatePassword: проверка ограничения пароля.