Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

Тема проекта «Настольное приложение для администраторов магазина по продаже готовых компьютеров – “Компьютерный Мир”»

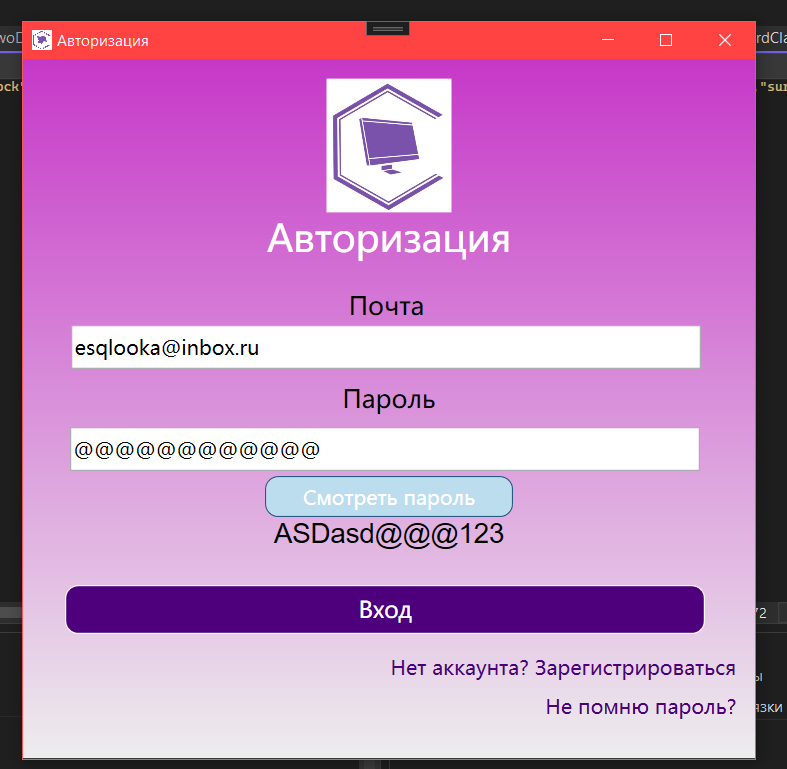
Практическая работа №2 «Разработка тестовых сценариев для аутентификации»

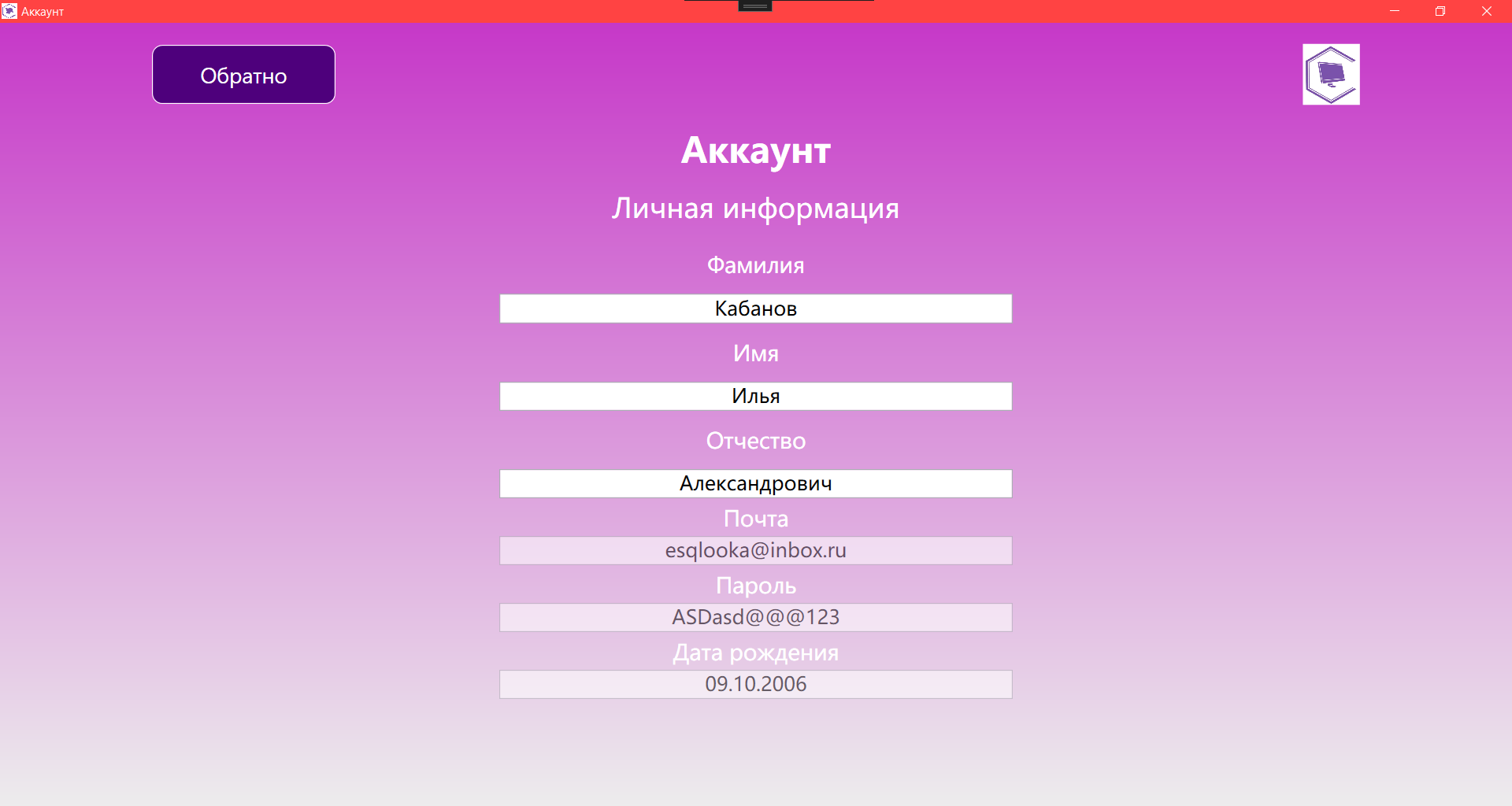
МДК.01.02 Поддержка и тестирование программных модулей

|  |  |
| --- | --- |
|  | Преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л.Р. Набиева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  | Студент гр. 22П-3  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.А. Кабанов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Уфа, 2024

1. Ввод корректного логина и пароля





private void Enters()

{

string email = tb\_Name.Text;

string password = tb\_password.Password;

if (email == "" || password == "")

{

MessageBox.Show("Логин/Пароль - пустые.",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Error

);

return;

}

if (!IsEmailRight(email))

{

return;

}

List<User> users = UserDeserilise();

for (int i = 0; i < users.Count; i++)

{

if (users[i].email == email)

{

if (!IsBlock(users[i]))

{

return;

}

if (users[i].password == password)

{

if (!TwoAuth(users[i].email, users[i].isTwoAuth))

{

numToBlock += 1;

return;

}

if (!Capcha(users[i]))

{

return;

}

GoToMenu(users[i]);

return;

}

}

}

numToBlock += 1;

if (numToBlock == 5)

{

BlockUser(users, email);

numToBlock = 0;

}

MessageBox.Show("Логин/Пароль - введены неверные данные.",

"Ошибка",

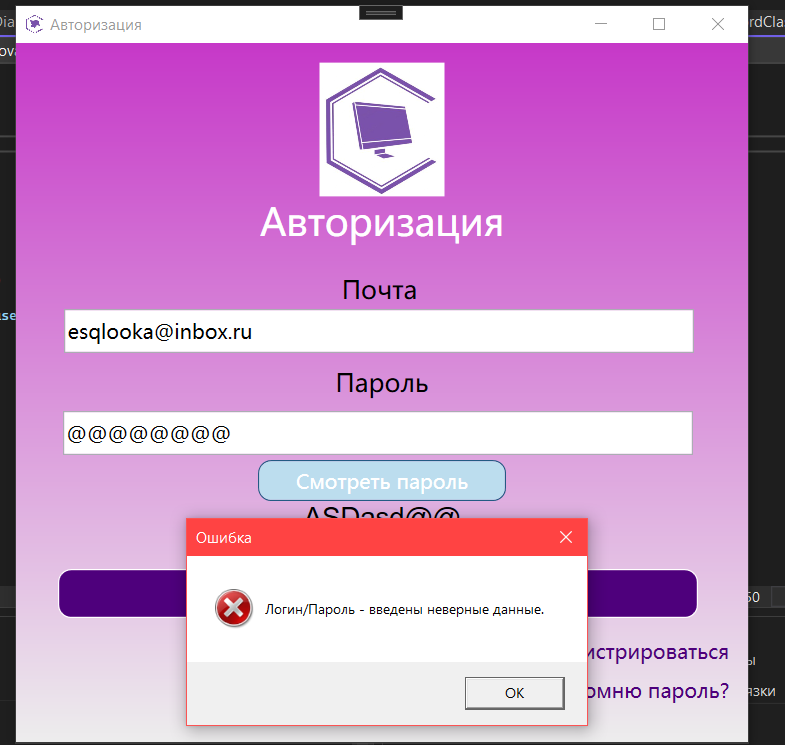
MessageBoxButton.OK,

MessageBoxImage.Error

);

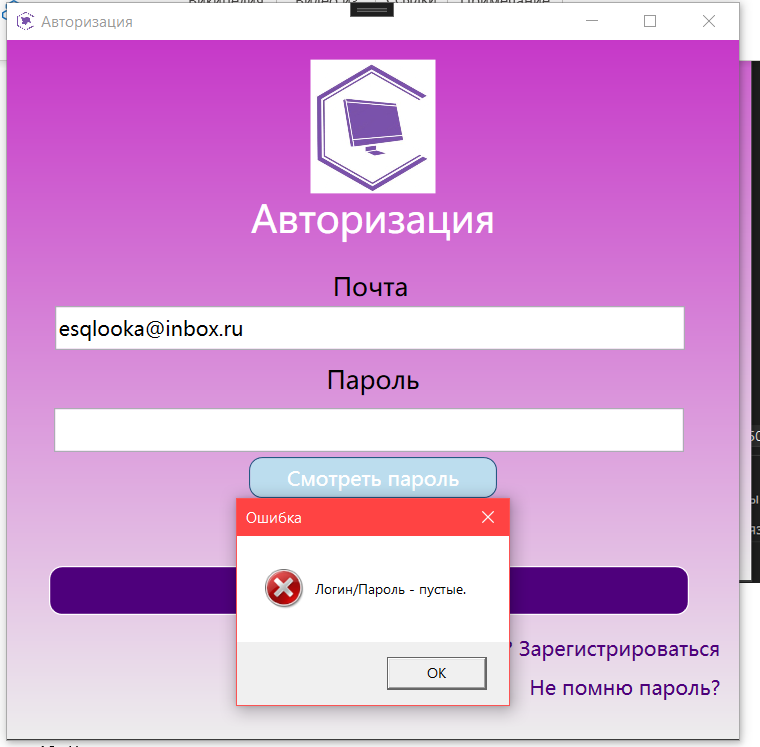
}

1. Ввод неправильного логина или пароля



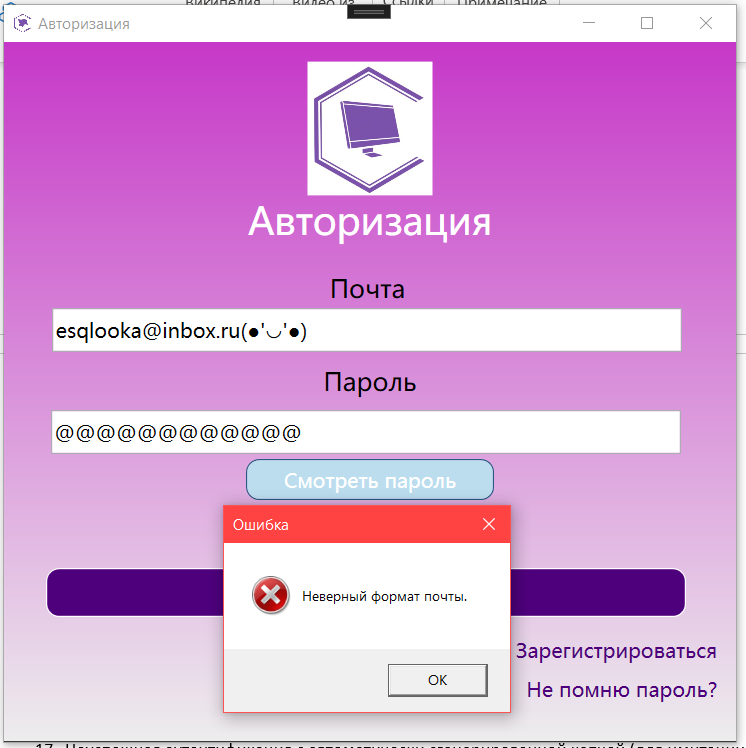
Код аналогичен коду из пункта №1

1. Ввод пустого логина или пароля



Код аналогичен коду из пункта №1

1. Ввод логина или пароля с недопустимыми символами



private bool IsEmailRight(string email)

{

Regex rg = new Regex(@"[A-Za-z0-9]+@[A-Za-z]+**\.**[A-Za-z]+$");

if (email == "" || !rg.IsMatch(email))

{

MessageBox.Show("Неверный формат почты.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

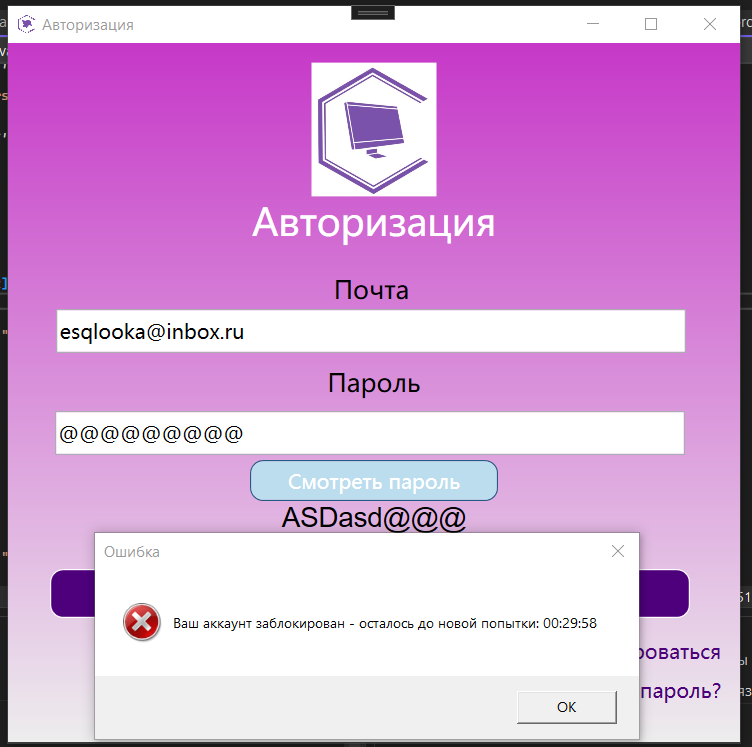
return false;

}

return true;

}

1. Превышение максимального количества неудачных попыток входа



private void BlockUser(List<User> users, string email)

{

User user = users.Where(c => c.email == email).FirstOrDefault();

if (user != null)

{

user.block = DateTime.Now.ToString();

ClearJsonFile("users\_test.json");

var userIndex = users.FindIndex(c => c.email == email);

if (userIndex != -1)

{

users[userIndex] = user;

}

UserSerialize(users);

}

}

private bool IsBlock(User user)

{

DateTime date\_now = DateTime.Now;

string date\_block = user.block;

try

{

DateTime datetime\_block = Convert.ToDateTime(date\_block);

if (date\_now.AddMinutes(-30.0) > datetime\_block)

{

return true;

}

TimeSpan ostatok = datetime\_block.AddMinutes(30) - date\_now;

MessageBox.Show($"Ваш аккаунт заблокирован - осталось до новой попытки: {ostatok.ToString().Split('.')[0]}",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Error);

return false;

}

catch (FormatException)

{

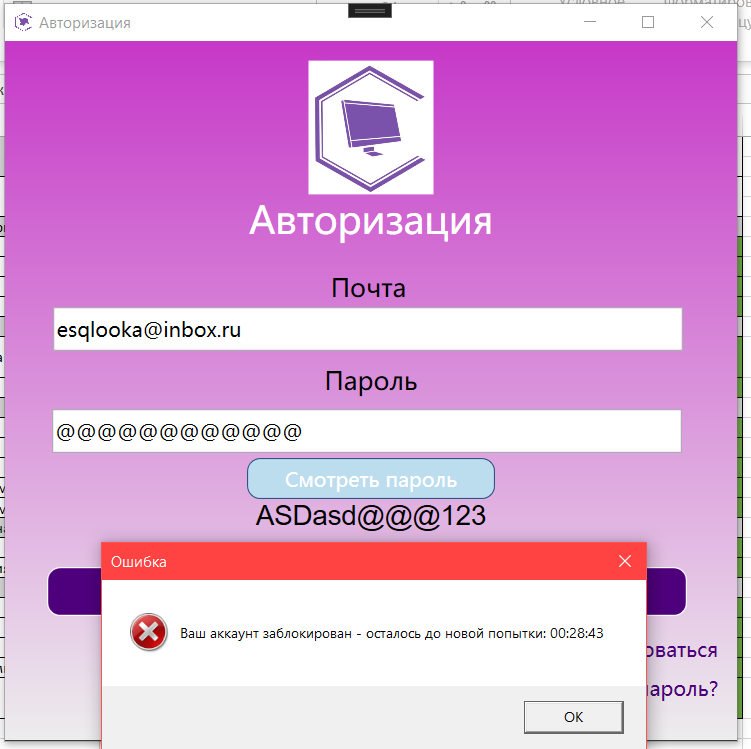
return true;

}

}

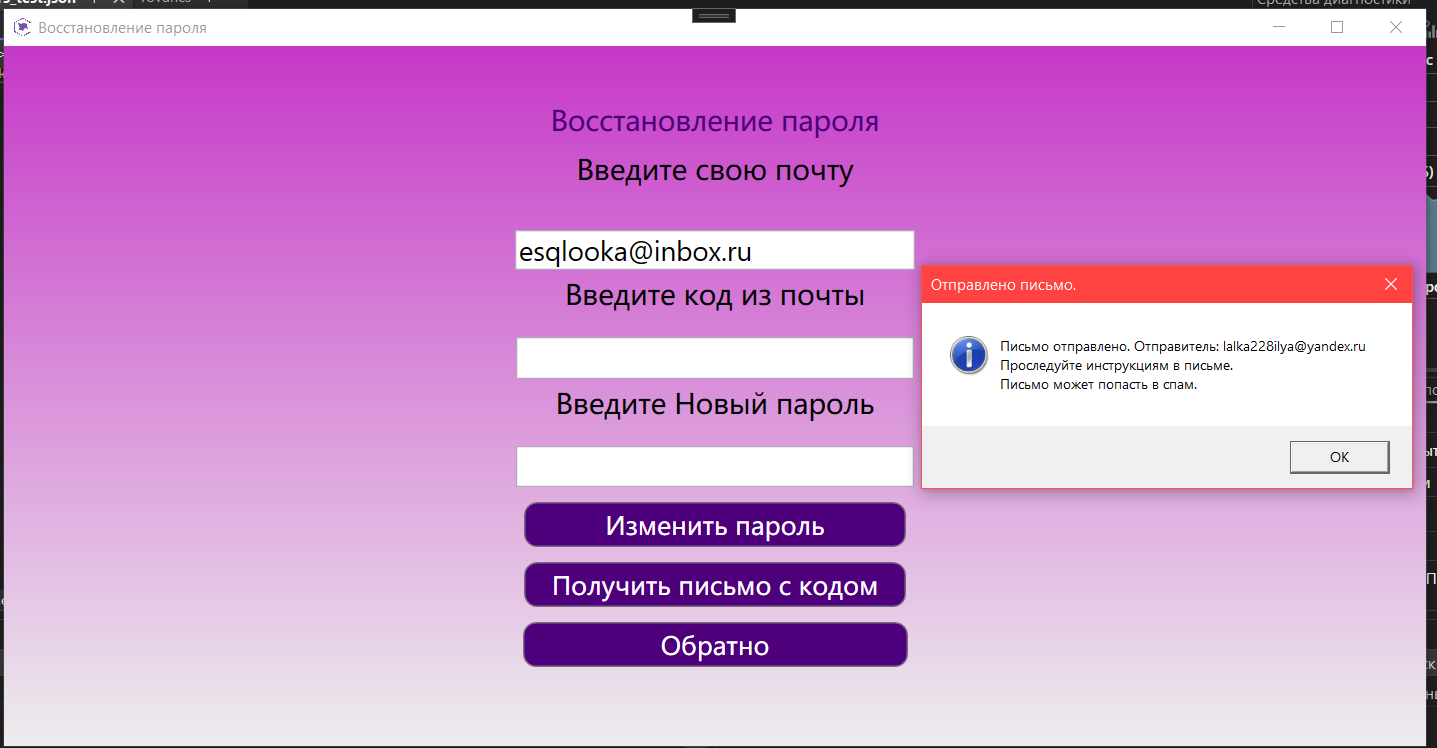
1. Попытка входа после блокировки аккаунта

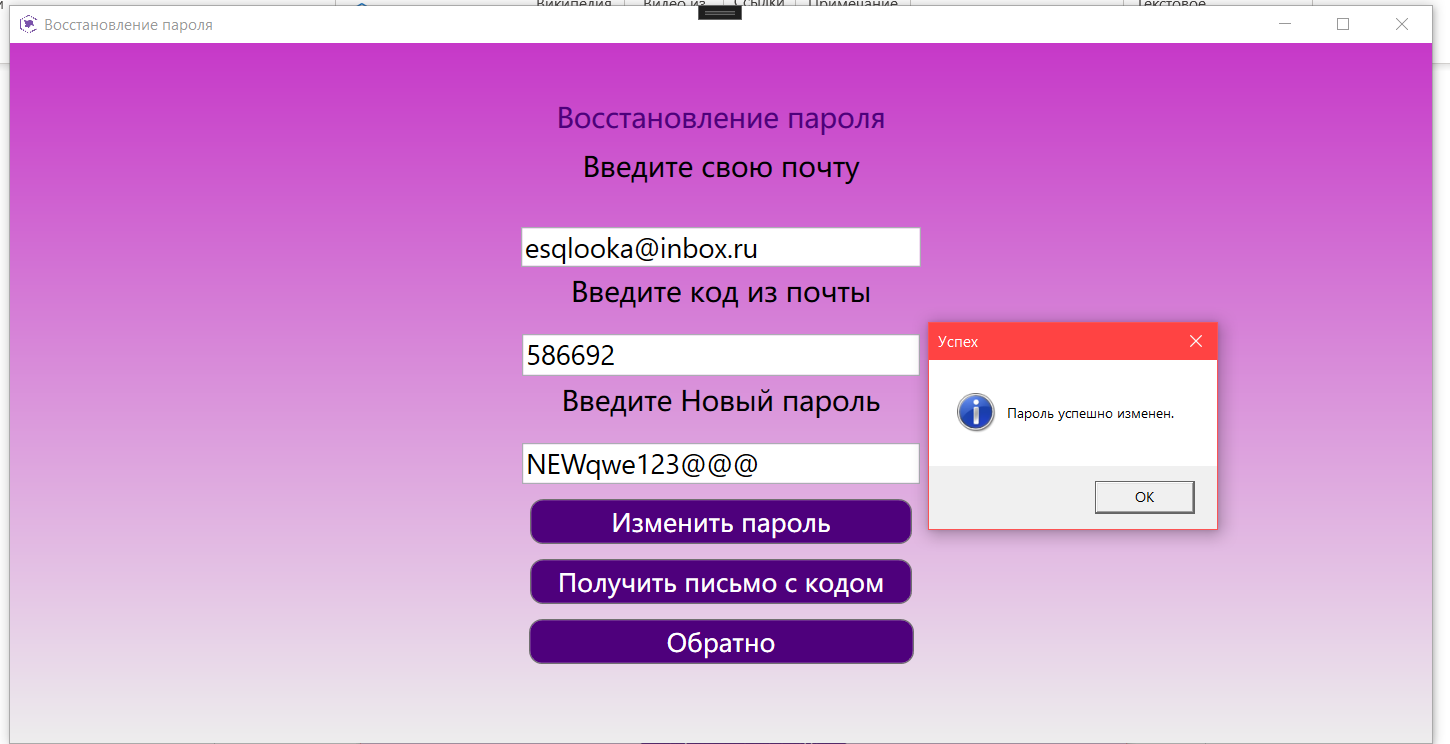
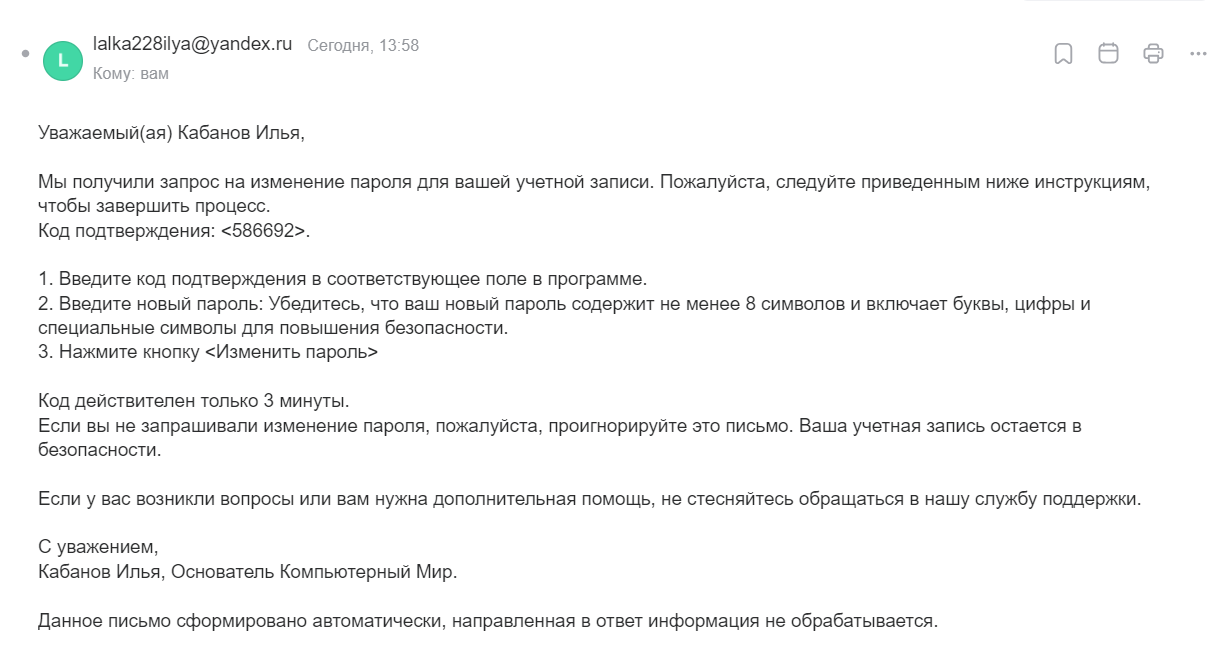
Пароль верный, но аккаунт заблокирован до истечения срока блокировки.



Код аналогичен коду их пункта 5.

1. Успешное восстановление пароля





private void ReturnPs()

{

string new\_pass = tb\_new\_pass.Text;

if (!Validation(new\_pass))

{

return;

}

string email = tb\_name.Text;

if (email == "")

{

return;

}

else if (!IsUserReal(email))

{

return;

}

string auth = tb\_surname.Text;

if (!IsKodRight(auth))

{

return;

}

List<User> users = UserDeserilise();

for (int i = 0; i < users.Count; i++)

{

if (email == users[i].email)

{

RedactPasswordJson(new\_pass, email, users);

MessageBox.Show("Пароль успешно изменен.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

Back();

return;

}

}

}

private void RedactPasswordJson(string new\_pass, string email, List<User> users)

{

User user = users.Where(c => c.email == email).FirstOrDefault();

if (user != null)

{

user.password = new\_pass;

ClearJsonFile("users\_test.json");

var userIndex = users.FindIndex(c => c.email == email);

if (userIndex != -1)

{

users[userIndex] = user;

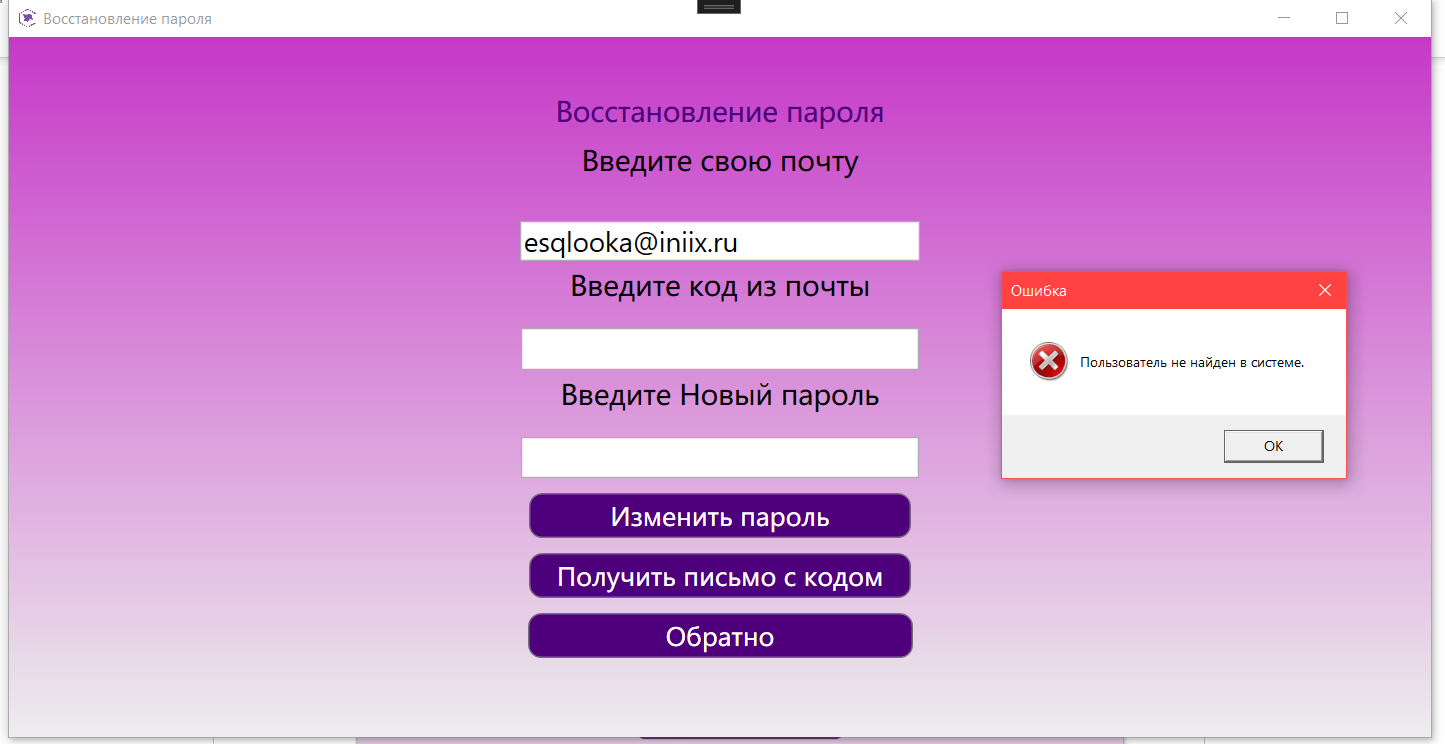
}

UserSerialize(users);

}

}

1. Неуспешное восстановление пароля с несуществующим email



private bool IsUserReal(string email)

{

List<User> users = UserDeserilise();

User user = users.Where(us => us.email == email).FirstOrDefault();

if (user == null)

{

MessageBox.Show("Пользователь не найден в системе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

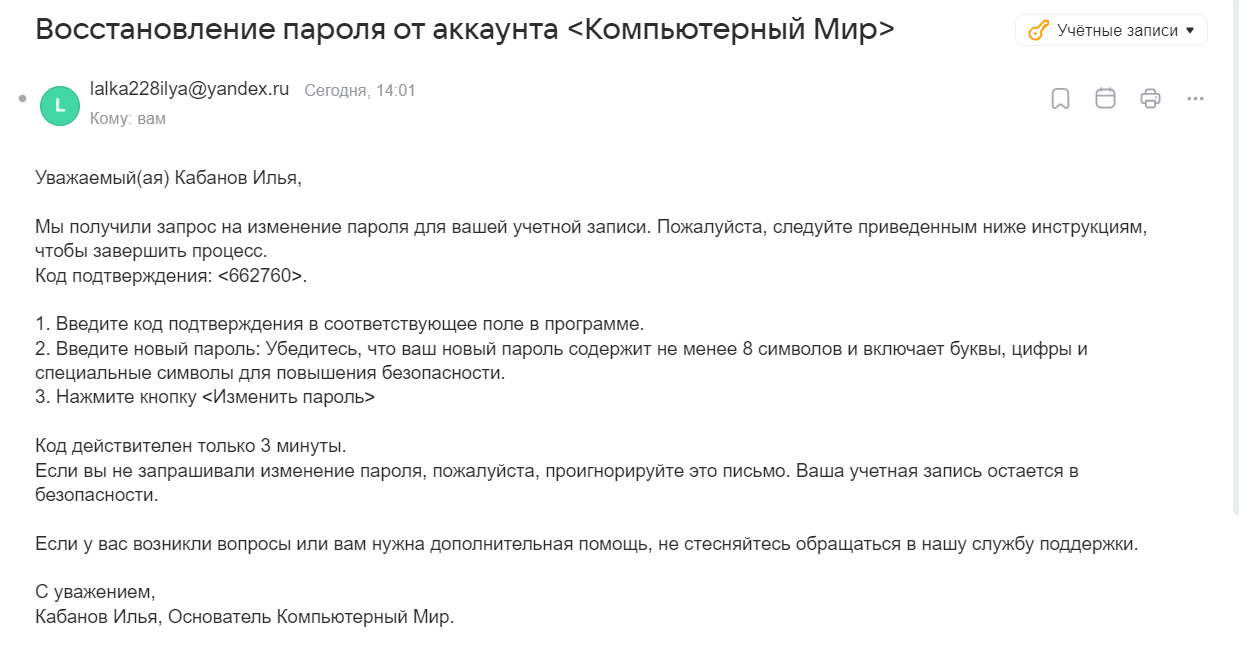
return false;

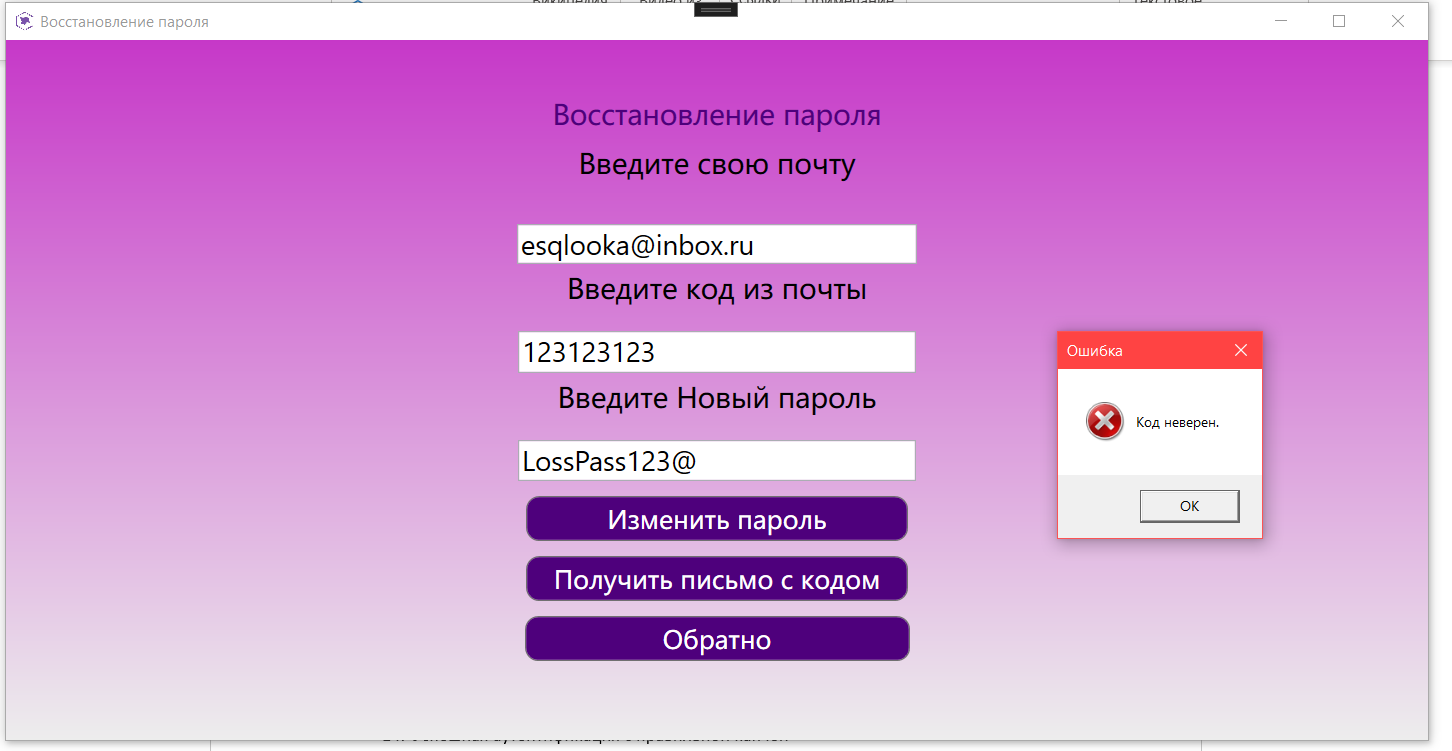
}

return true;

}

1. Неуспешное восстановление пароля с неправильным кодом из письма





private bool IsKodRight(string auth)

{

DateTime now = DateTime.Now;

if (auth != auth\_pass)

{

MessageBox.Show("Код неверен.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

else if (now.AddMinutes(-3) > date\_auth\_pass)

{

MessageBox.Show("Код истек.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

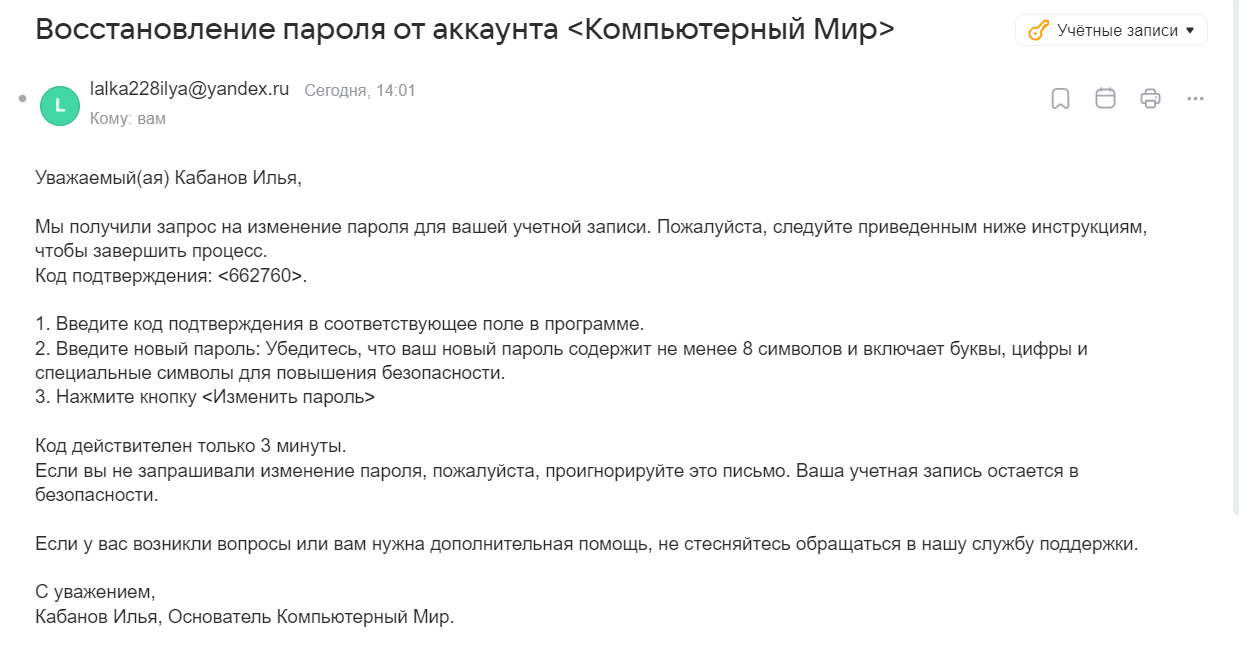
return false;

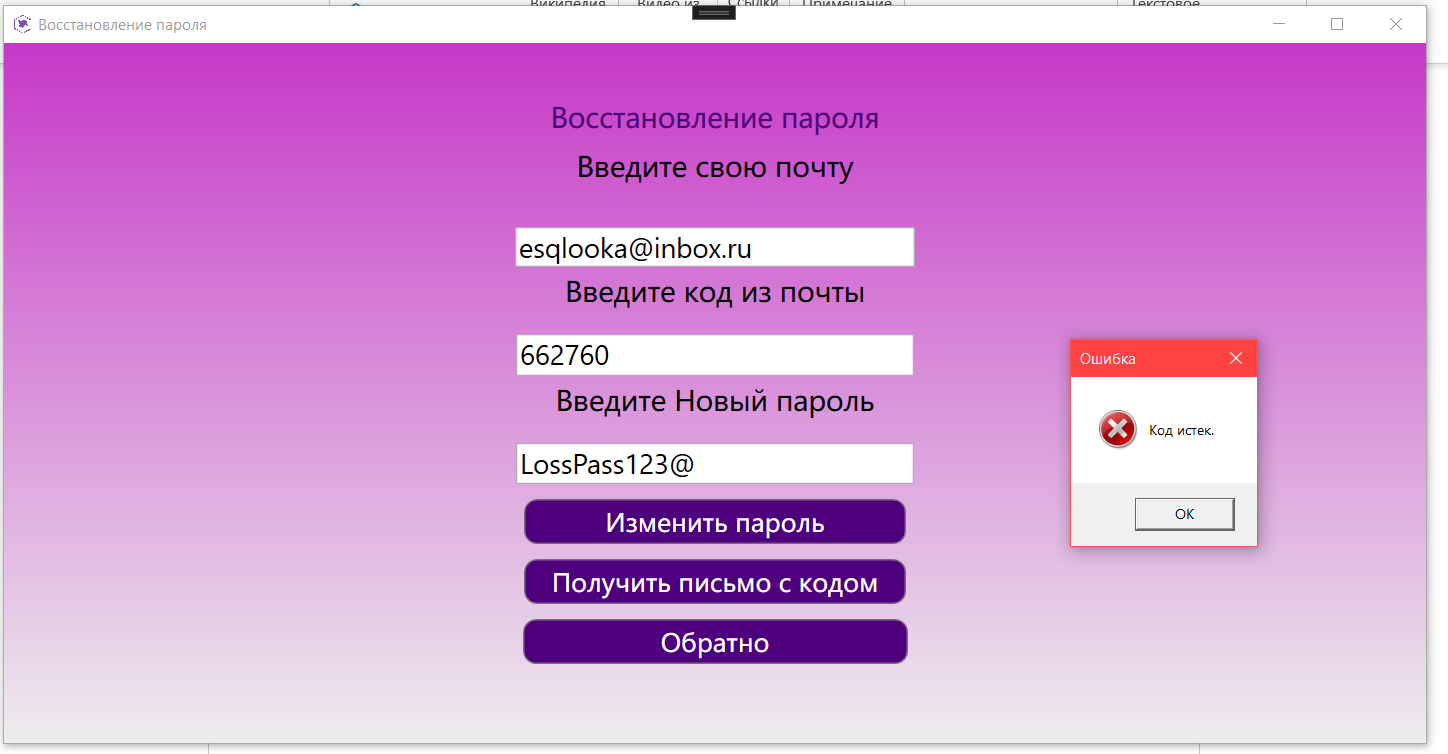
}

return true;

}

1. Неуспешное восстановление пароля с просроченным кодом из письма





private bool IsKodRight(string auth)

{

DateTime now = DateTime.Now;

if (auth != auth\_pass)

{

MessageBox.Show("Код неверен.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

else if (now.AddMinutes(-3) > date\_auth\_pass)

{

MessageBox.Show("Код истек.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

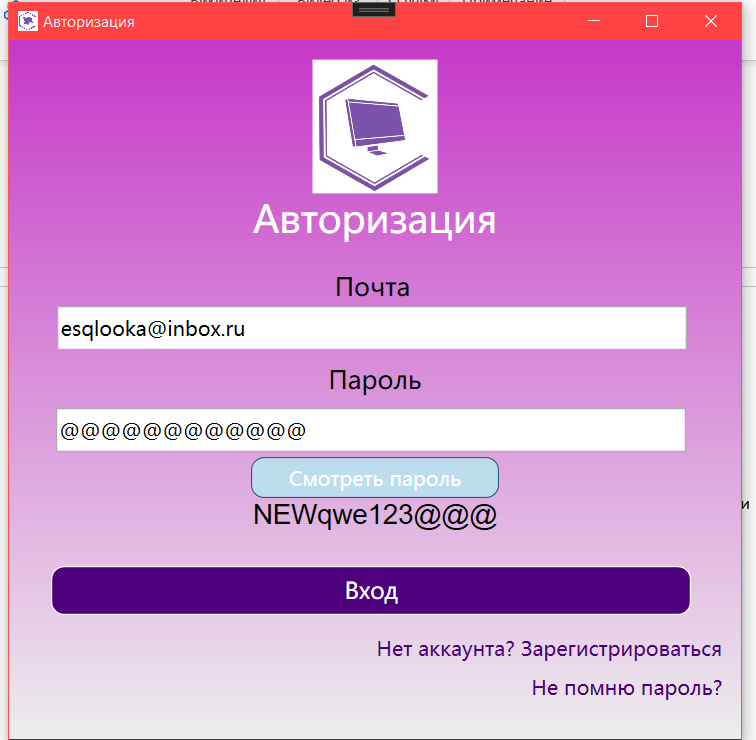
return false;

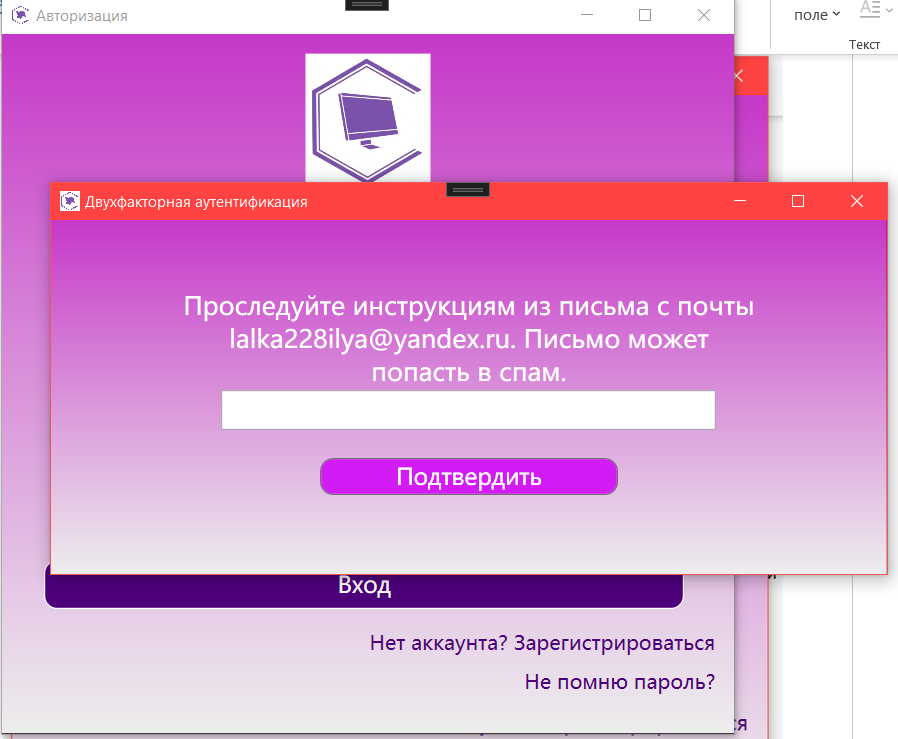
}

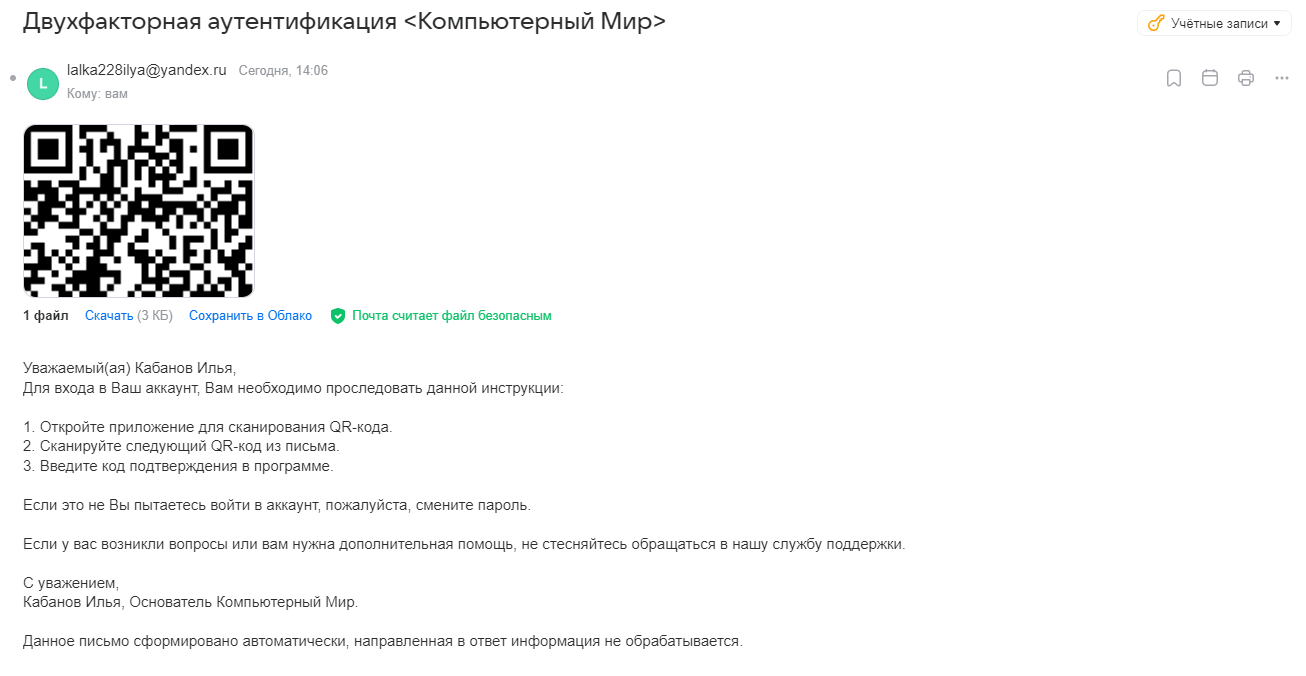
return true;

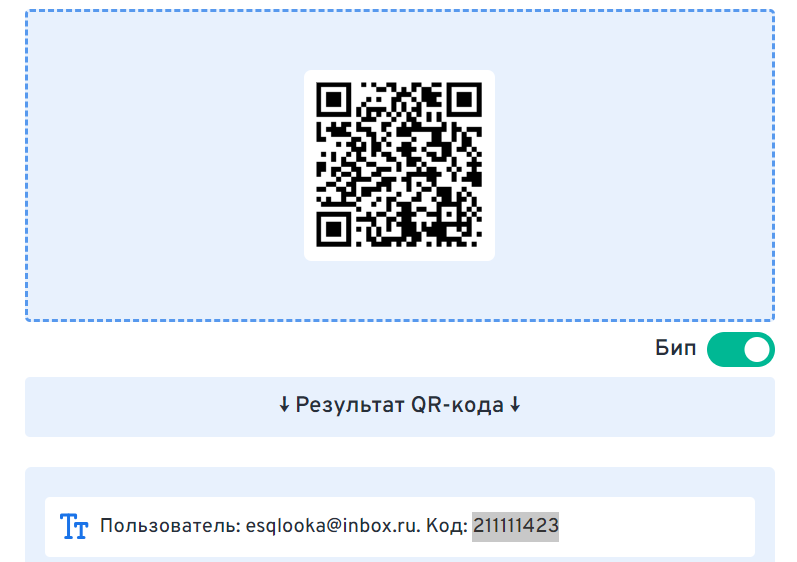
}

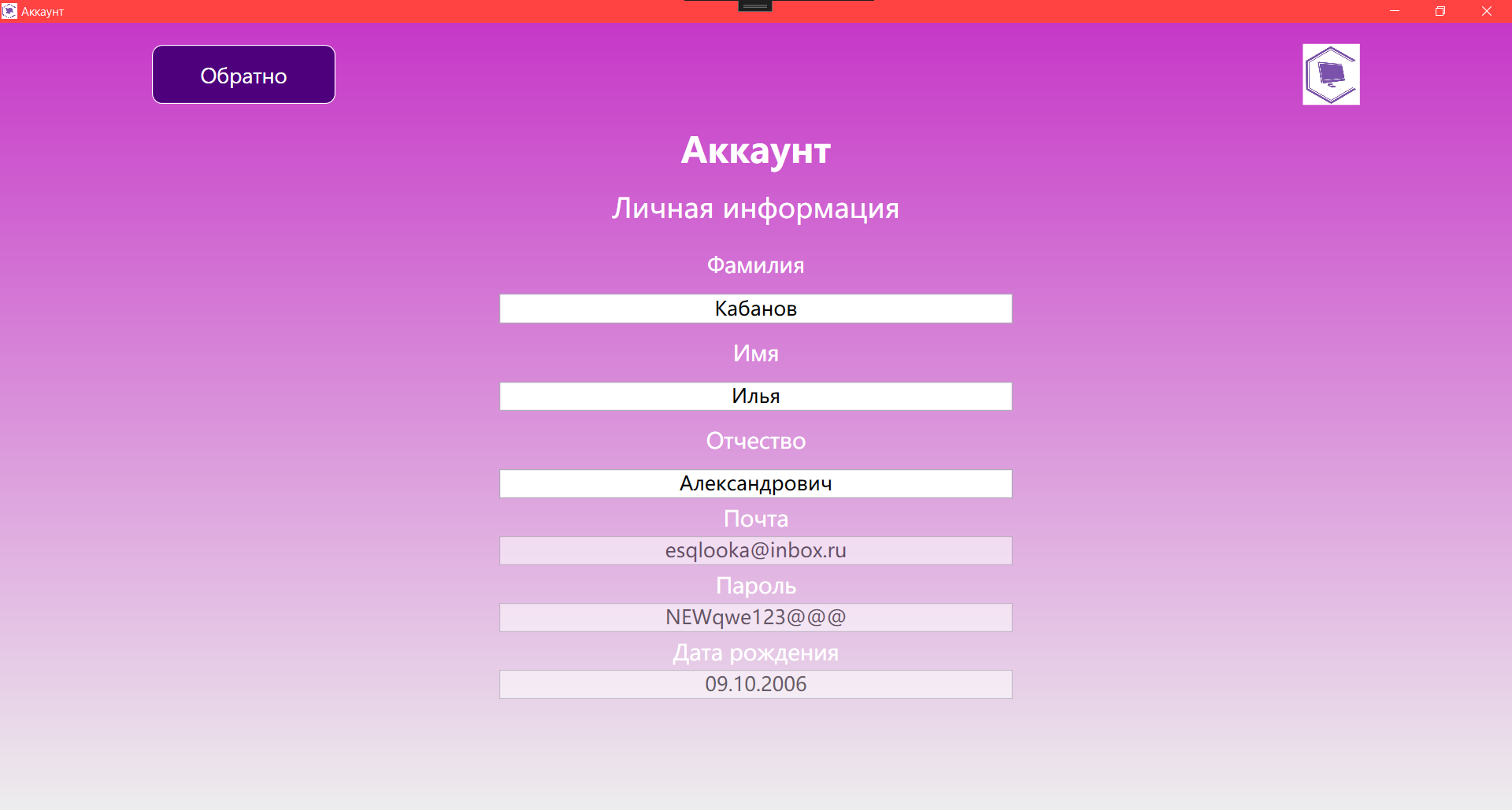
1. Успешная аутентификация с двухфакторной аутентификацией











private void IsRightKod()

{

string kod\_user = tb\_kod.Text;

if (auth\_pass == kod\_user)

{

this.DialogResult = true;

this.Close();

}

else

{

MessageBox.Show("Неверный код.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

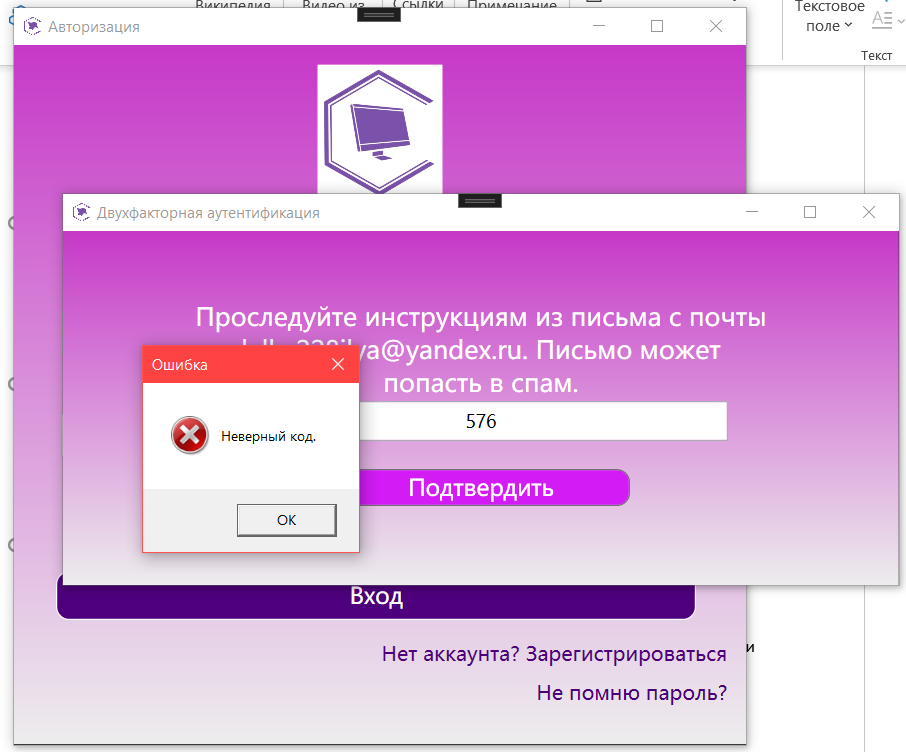
this.DialogResult = false;

this.Close();

}

}

1. Неуспешная аутентификация с неправильным кодом подтверждения



private void IsRightKod()

{

string kod\_user = tb\_kod.Text;

if (auth\_pass == kod\_user)

{

this.DialogResult = true;

this.Close();

}

else

{

MessageBox.Show("Неверный код.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

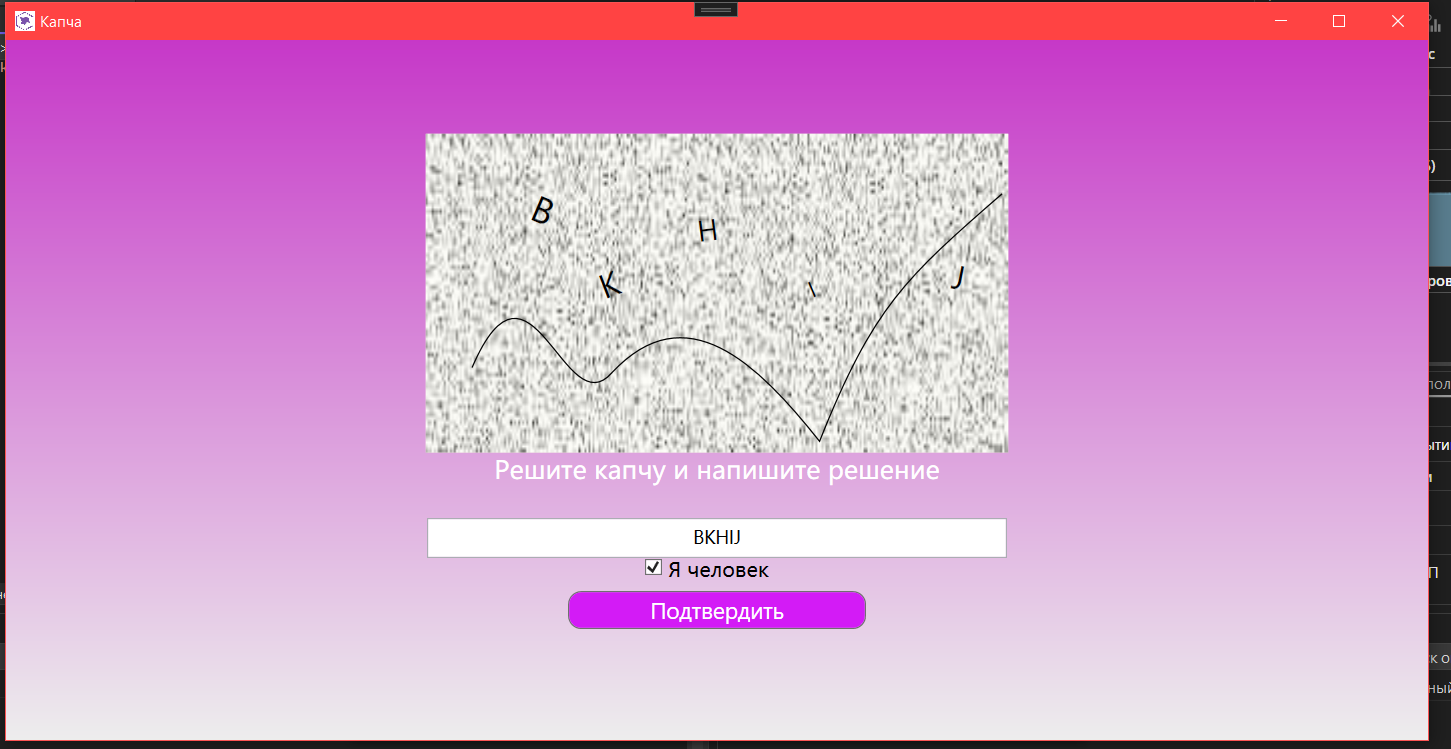
this.DialogResult = false;

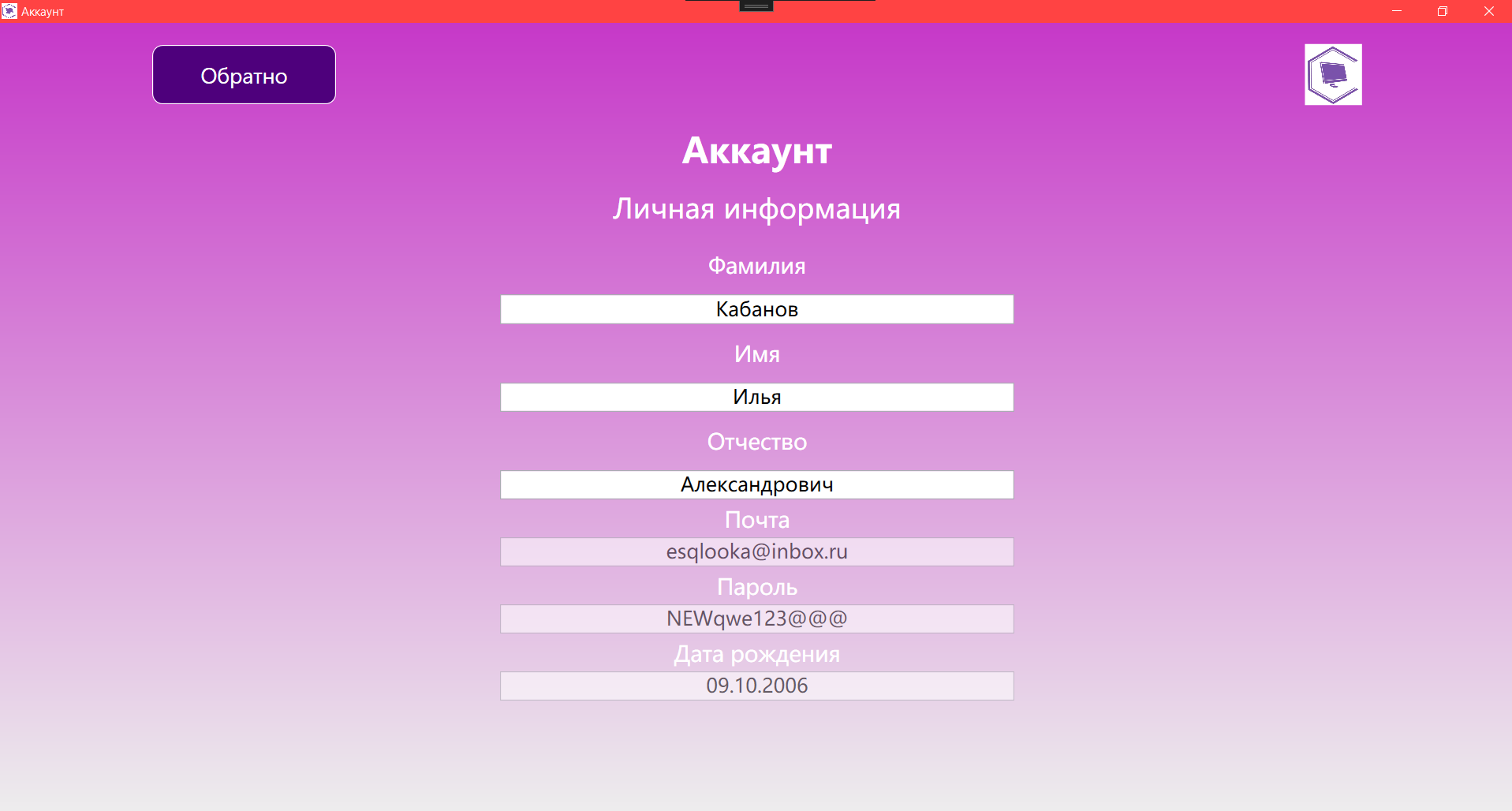
this.Close();

}

}

1. Успешная аутентификация с правильной капчей





private void Generate(Captcha captcha)

{

Random rnd = new Random();

captcha.CreateCaptcha(Captcha.LetterOption.Alphanumeric, rnd.Next(4,7));

pass = MyCaptcha.CaptchaText;

}

private void bt\_go\_Click(object sender, RoutedEventArgs e)

{

IsRight();

}

private void IsRight()

{

string pass\_user = tb\_kod.Text;

if (pass\_user == pass && cb\_i\_people.IsChecked == true)

{

this.DialogResult = true;

this.Close();

}

else if (pass\_user == "")

{

Generate(MyCaptcha);

tb\_reenter.Text = "Пустой ввод. Решите новую капчу.";

BlockUser++;

IsBlock();

}

else

{

Generate(MyCaptcha);

tb\_reenter.Text = "Неправильный ввод. Решите новую капчу.";

tb\_kod.Text = "";

BlockUser++;

IsBlock();

}

}

1. Неуспешная аутентификация с неправильной капчей



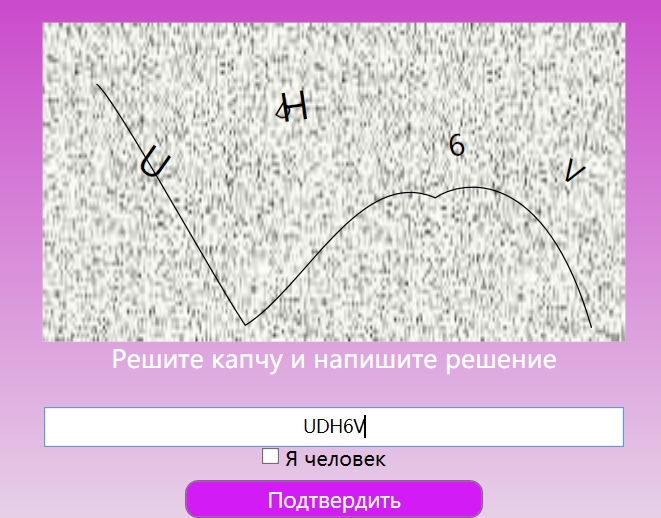
Код аналогичен коду из пункта 13.

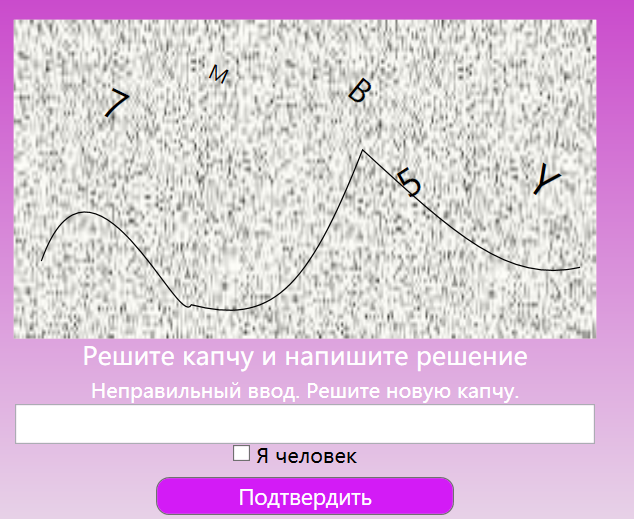
1. Неуспешная аутентификация с пустой капчей



Код аналогичен коду из пункта 13.

1. Неуспешная аутентификация с автоматически сгенерированной капчей (для имитации бота)

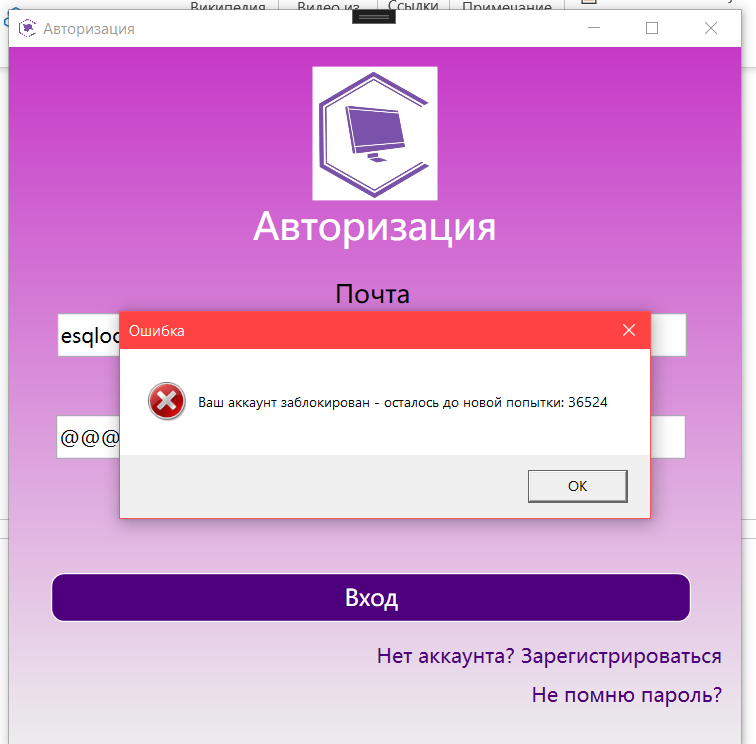




Код аналогичен коду из пункта 13.

1. Блокировка аккаунта при продолжительном неверном вводе капчи

При неверном вводе капчи более 3 раз – блокировка на 100 лет.



private void IsBlock()

{

if (BlockUser >= 3)

{

DateTime dt = DateTime.Now;

dt = dt.AddYears(100);

user.block = dt.ToString();

List<User> users = UserDeserilise();

ClearJsonFile("users\_test.json");

var userIndex = users.FindIndex(c => c.id == user.id);

if (userIndex != -1)

{

users[userIndex] = user;

}

UserSerialize(users);

this.DialogResult = false;

}

}