Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

Тема проекта «Продажа автомобилей’LADA’»

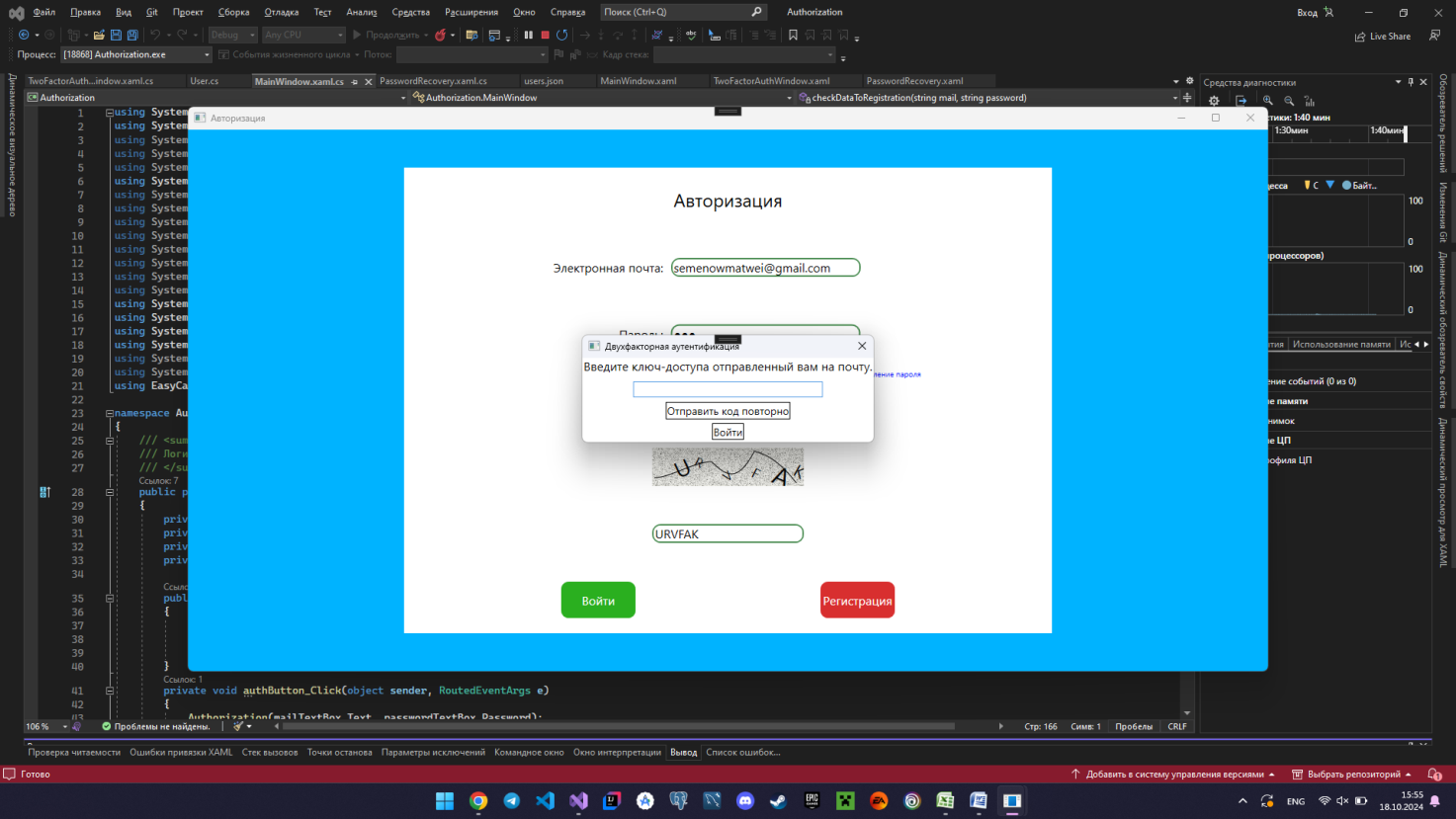
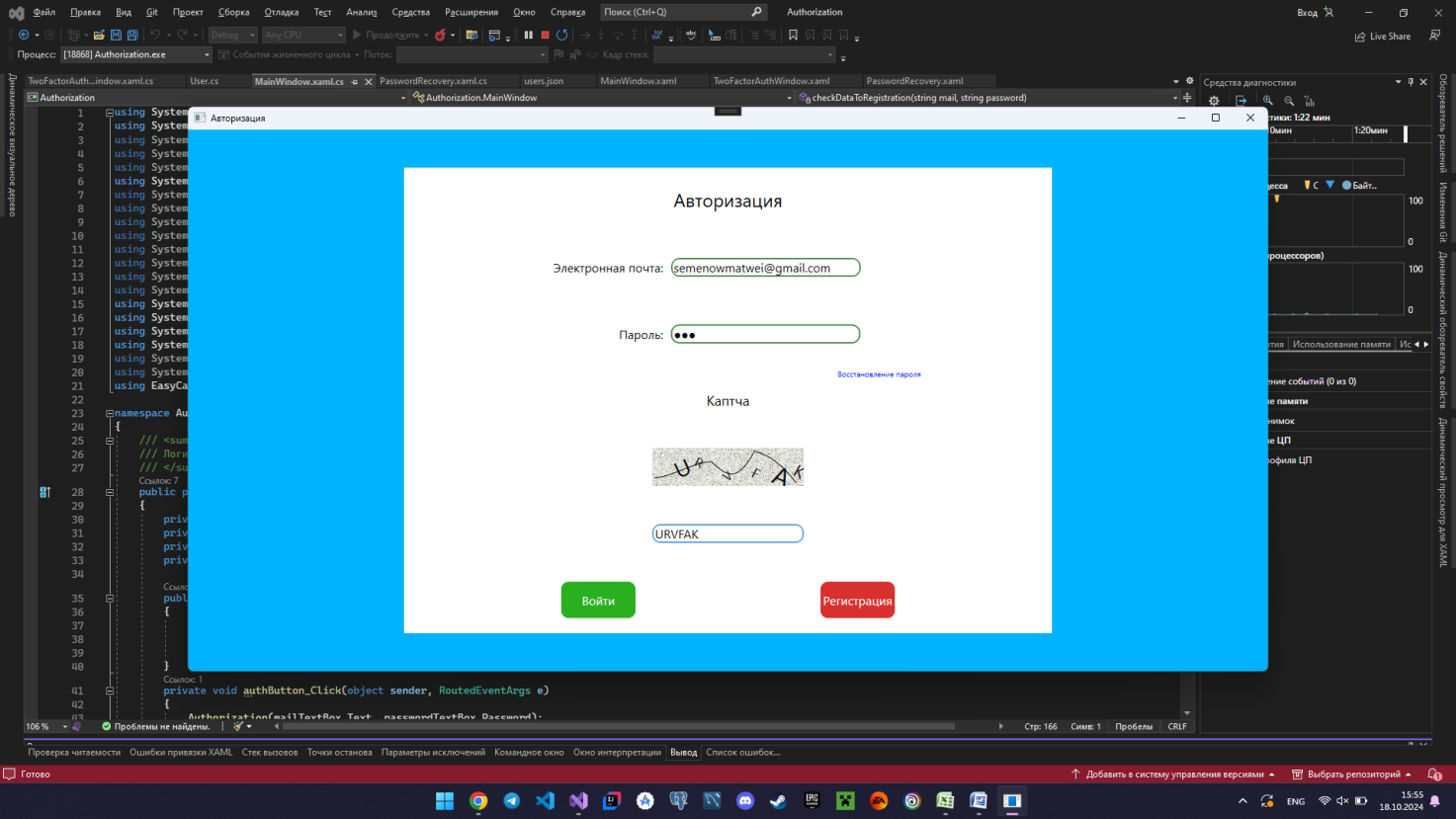
Практическая работа №2 «Разработка тестовых сценариев для аутентификации»

МДК.01.02 Поддержка и тестирование программных модулей

|  |  |
| --- | --- |
|  | Преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л.Р. Набиева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  | Студент гр. 22П-3  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Семенов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

2024

Описание теста: Ввод корректного логина и пароля.



private bool CheckOnEmptyString(string mail, string password)

{

return mail.Length > 0 && password.Length > 0;

}

private bool СheckToUncorrectData(string mail, string password)

{

return Regex.IsMatch(mail, @"^[a-zA-Z0-9]+@[a-zA-Z]+[.][a-zA-Z]+$") && Regex.IsMatch(password, @"^[a-zA-Z0-9]+$");

}

private bool СheckToUser(User user, string password)

{

return user.Password == password;

}

if (!CheckOnEmptyString(mail, password))

{

MessageBox.Show("Поля должны быть заполнены!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

if (!СheckToUncorrectData(mail, password))

{

MessageBox.Show("Неверный формат логина/пароля!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

if (user != null && !СheckToUser(user, password))

{

MessageBox.Show("Не правильно введены данные или такого пользователя не существует!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

failedAttemps++;

if (failedAttemps == 3)

{

failedAttemps = 0;

BlockAccount(user);

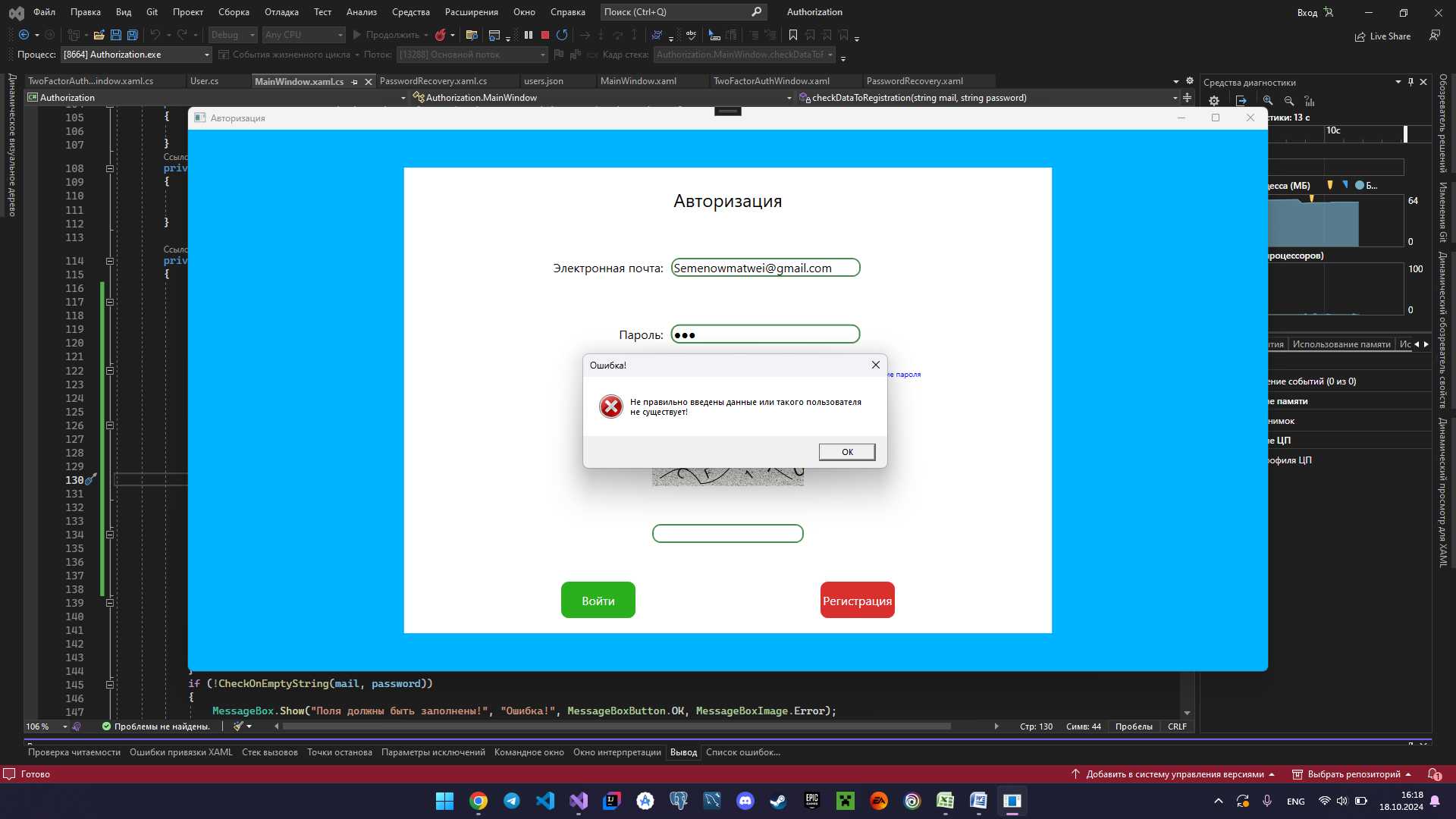
StartTimerBlockAccount();

}

return false;

}

Описание теста: Ввод неправильного логина или пароля.

 private User findUserOnEmail(string mail)

{

foreach(User user in users)

{

if (user.Mail == mail) return user;

}

return null;

}

private bool СheckToUser(User user, string password)

{

return user.Password == password;

}

var user = findUserOnEmail(mail);

if (user == null)

{

MessageBox.Show("Не правильно введены данные или такого пользователя не существует!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

if (!СheckToUser(user, password))

{

MessageBox.Show("Не правильно введены данные или такого пользователя не существует!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

failedAttemps++;

if (failedAttemps == 3)

{

failedAttemps = 0;

BlockAccount(user);

StartTimerBlockAccount();

}

return false;

}

Описание теста: Ввод пустого логина или пароля.

 private bool CheckOnEmptyString(string mail, string password)

{

return mail.Length > 0 && password.Length > 0;

}

if (!CheckOnEmptyString(mail, password))

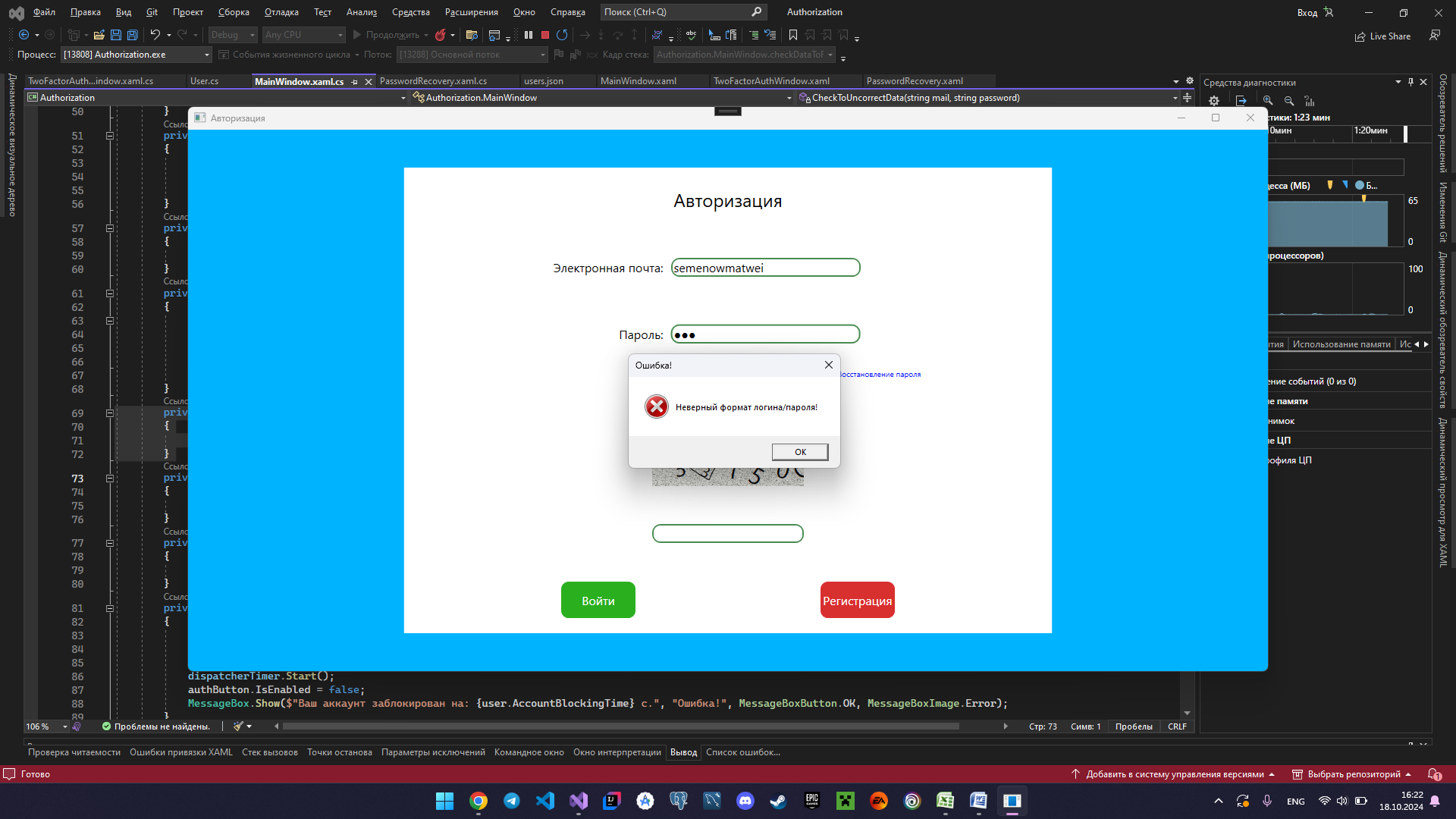
{

MessageBox.Show("Поля должны быть заполнены!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

Описание теста: Ввод логина или пароля с недопустимыми символами.

 private bool СheckToUncorrectData(string mail, string password)

{

return Regex.IsMatch(mail, @"^[a-zA-Z0-9]+@[a-zA-Z]+[.][a-zA-Z]+$") && Regex.IsMatch(password, @"^[a-zA-Z0-9]+$");

}

if (!СheckToUncorrectData(mail, password))

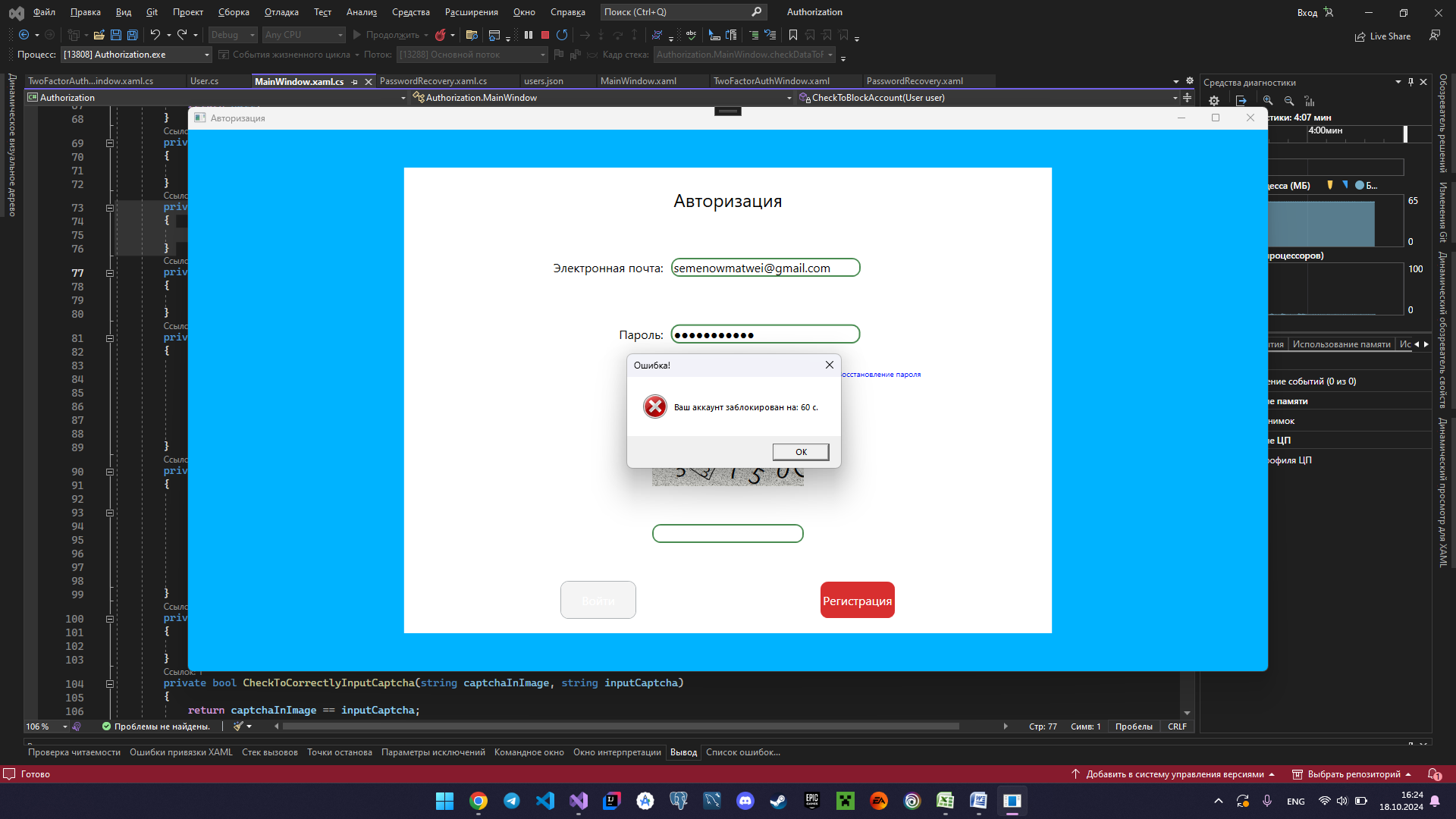
{

MessageBox.Show("Неверный формат логина/пароля!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

Описание теста: Превышение максимального количества неудачных попыток входа.



private void StartTimerBlockAccount()

{

DispatcherTimer dispatcherTimer = new DispatcherTimer();

dispatcherTimer.Tick += new EventHandler(dispatcherTimer\_Tick);

dispatcherTimer.Interval = new TimeSpan(0, 0, 1);

dispatcherTimer.Start();

authButton.IsEnabled = false;

MessageBox.Show($"Ваш аккаунт заблокирован на: {user.AccountBlockingTime} c.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

private void BlockAccount(User user)

{

this.user = user;

user.AccountBlockingTime = 60;

}

if (!СheckToUser(user, password))

{

MessageBox.Show("Не правильно введены данные или такого пользователя не существует!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

failedAttemps++;

if (failedAttemps == 3)

{

failedAttemps = 0;

BlockAccount(user);

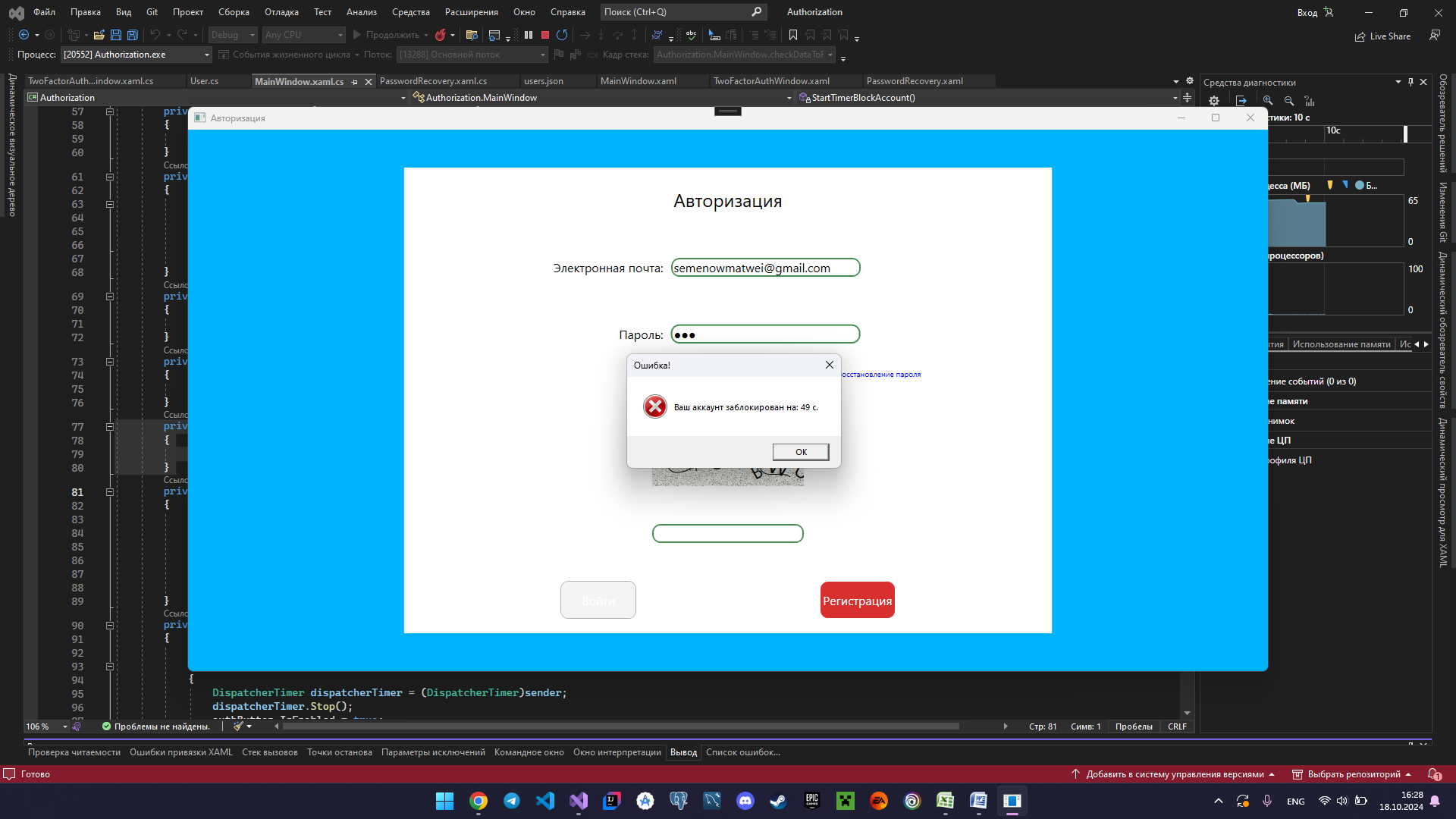
StartTimerBlockAccount();

}

return false;

}

Описание теста: Попытка входа после блокировки аккаунта.

 private bool CheckToBlockAccount(User user)

{

return user != null && user.AccountBlockingTime > 0;

}

private void StartTimerBlockAccount()

{

DispatcherTimer dispatcherTimer = new DispatcherTimer();

dispatcherTimer.Tick += new EventHandler(dispatcherTimer\_Tick);

dispatcherTimer.Interval = new TimeSpan(0, 0, 1);

dispatcherTimer.Start();

authButton.IsEnabled = false;

MessageBox.Show($"Ваш аккаунт заблокирован на: {user.AccountBlockingTime} c.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

if (CheckToBlockAccount(user))

{

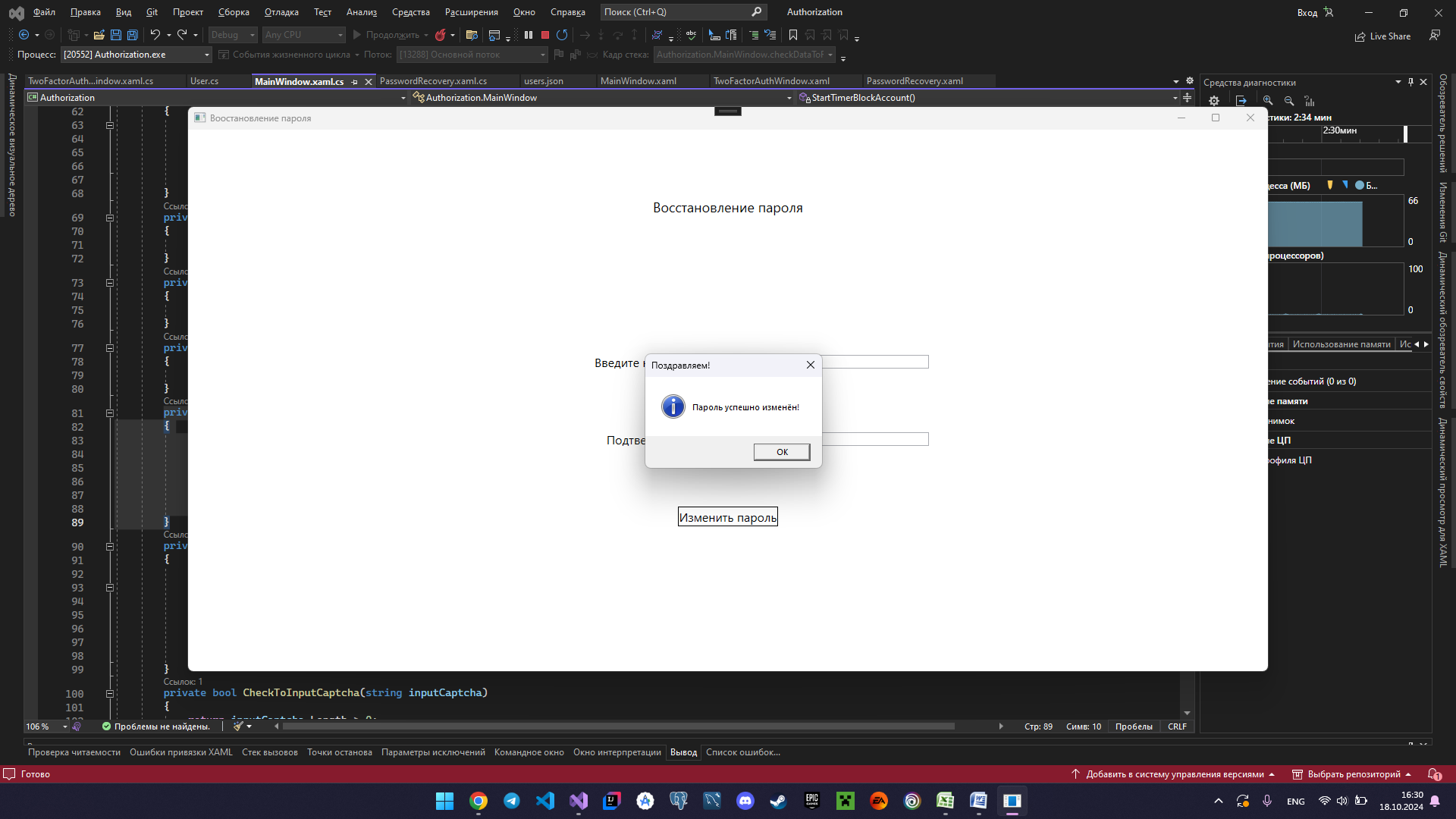
this.user = user;

StartTimerBlockAccount();

return false;

}

Описание теста: Успешное восстановление пароля.

 private User FindUserOnEmail()

{

foreach(User user in AllUser())

{

if (user.Mail == mailSend) return user;

}

return null;

}

private void WriteNewData(User updateDataUser)

{

List<User> users = AllUser();

int index = 0;

foreach (User user in users) {

if (user.Id == updateDataUser.Id)

{

break;

}

index++;

}

users.RemoveAt(index);

users.Add(updateDataUser);

WriteToJson(users);

}

User user = FindUserOnEmail();

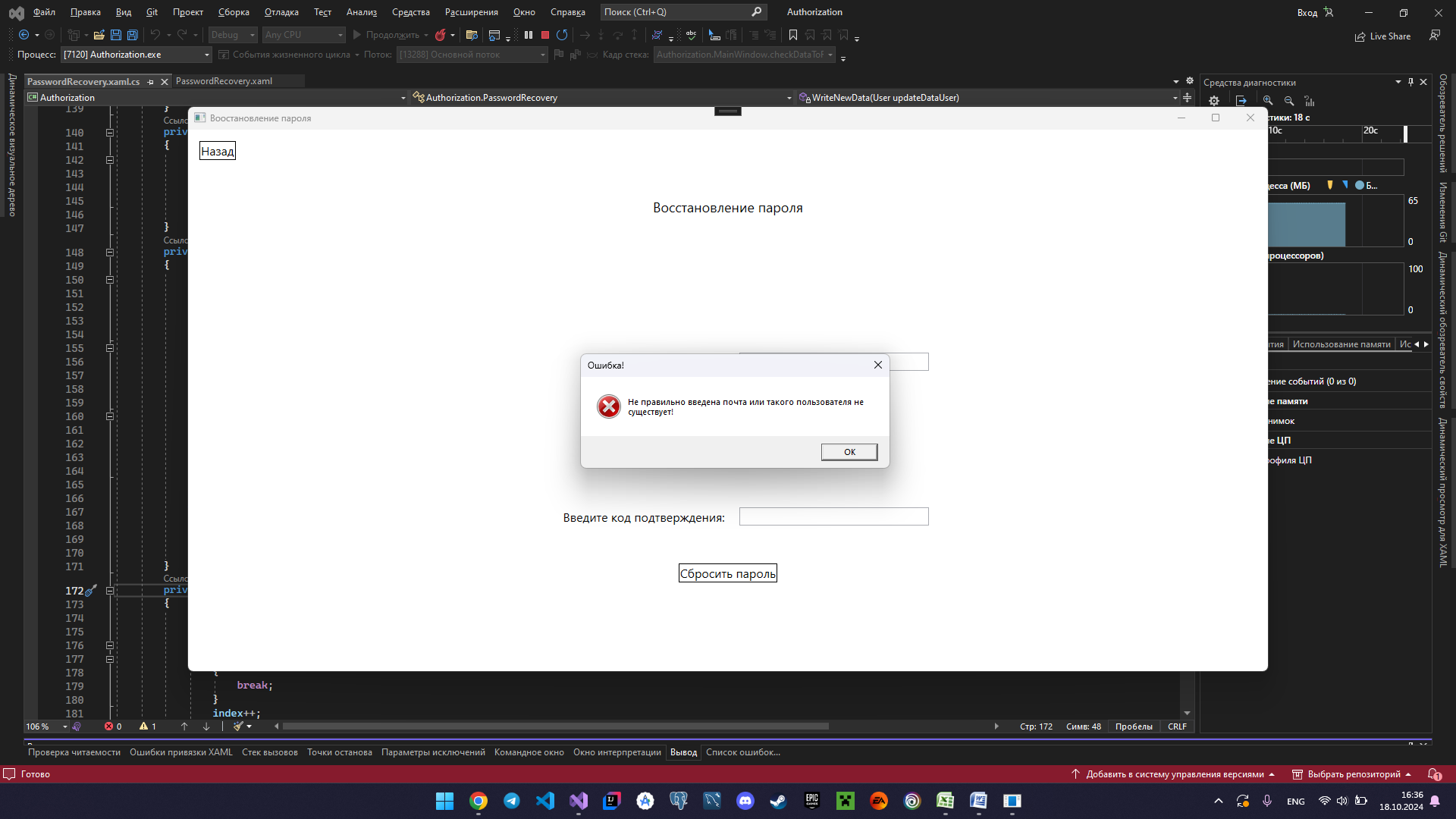
user.Password = newPasswordTextBox.Password;

WriteNewData(user);

MessageBox.Show("Пароль успешно изменён!", "Поздравляем!" ,MessageBoxButton.OK, MessageBoxImage.Information);

new MainWindow().Show();

this.Close();

Описание теста: Неуспешное восстановление пароля с несуществующим email. 

private User FindUserOnEmail()

{

foreach(User user in AllUser())

{

if (user.Mail == mailSend) return user;

}

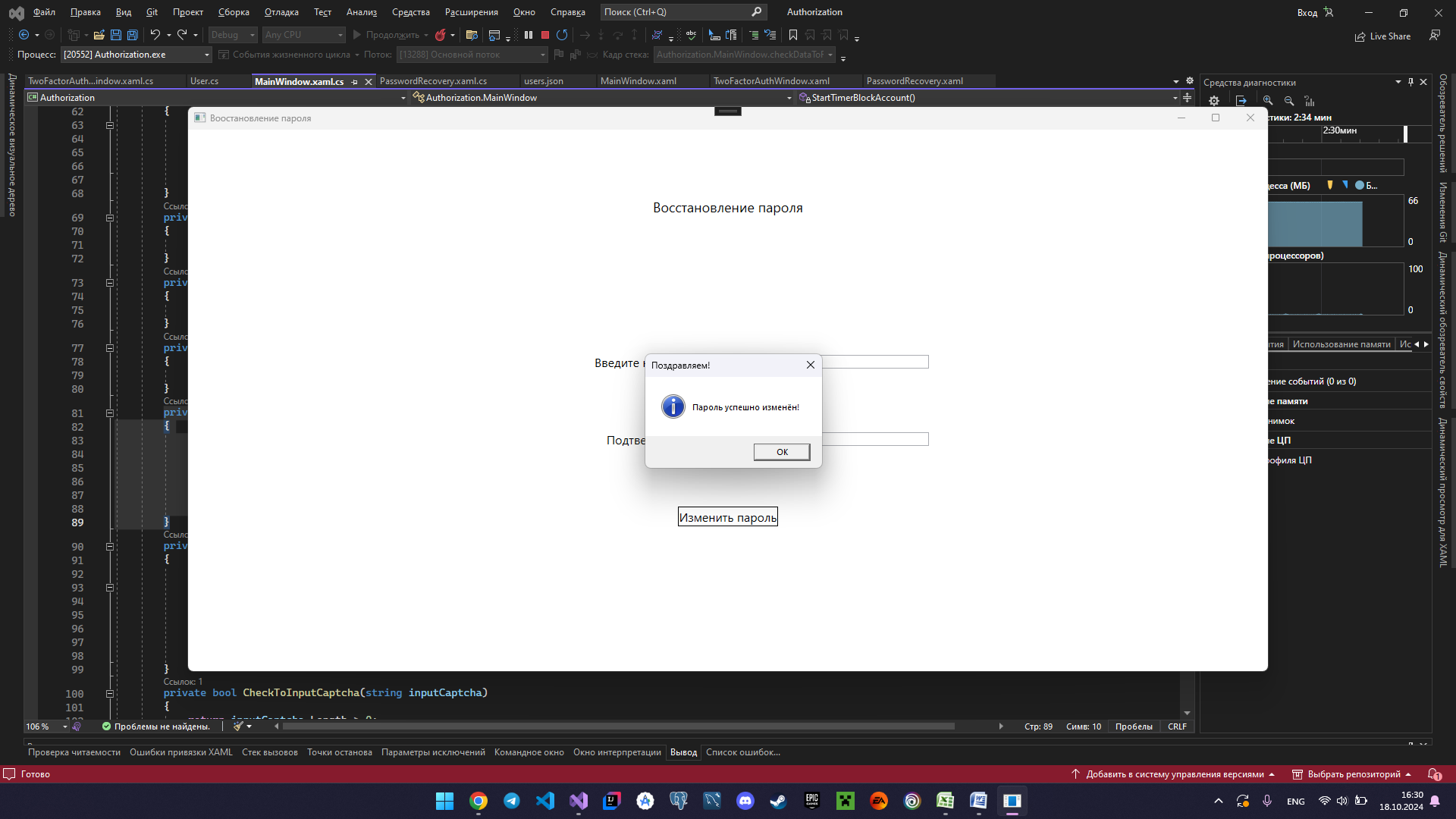
return null;

}

if (FindUserOnEmail() != null && SendMessage()) MessageBox.Show("Сообщение с кодом подтвердения отправлен вам на почту.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

else MessageBox.Show("Не правильно введена почта или такого пользователя не существует!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

Описание теста: Успешное восстановление пароля через отправку на почту.

 private bool SendMessage()

{

string smtpServer = "smtp.yandex.ru";

int smtpPort = 587;

string smtpUsername = "LoginProject@yandex.ru";

string smtpPassword = "nsqwjwarmovcowml";

using (SmtpClient smtpClient = new SmtpClient(smtpServer, smtpPort))

{

smtpClient.Credentials = new NetworkCredential(smtpUsername, smtpPassword);

smtpClient.EnableSsl = true;

using (MailMessage mailMessage = new MailMessage())

{

mailMessage.From = new MailAddress(smtpUsername);

mailMessage.To.Add(mailSend);

mailMessage.Subject = "Восстановление пароля";

mailMessage.Body = $"Вот ваш код подтверждения:\n{codeConfirmation}";

try

{

smtpClient.Send(mailMessage);

}

catch (Exception ex)

{

return false;

}

}

}

return true;

}

private User FindUserOnEmail()

{

foreach(User user in AllUser())

{

if (user.Mail == mailSend) return user;

}

return null;

}

private void WriteNewData(User updateDataUser)

{

List<User> users = AllUser();

int index = 0;

foreach (User user in users) {

if (user.Id == updateDataUser.Id)

{

break;

}

index++;

}

users.RemoveAt(index);

users.Add(updateDataUser);

WriteToJson(users);

}

User user = FindUserOnEmail();

user.Password = newPasswordTextBox.Password;

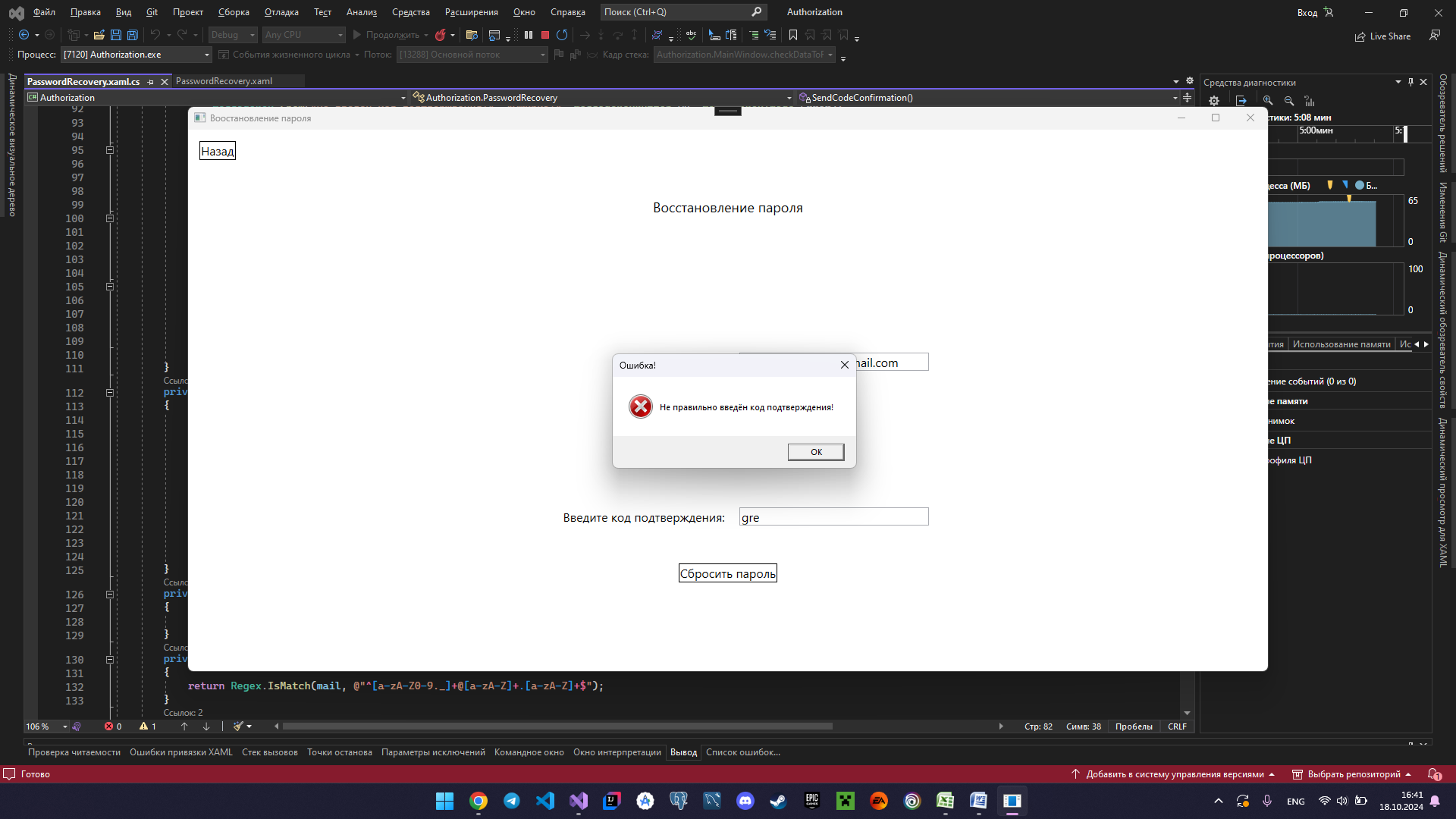
WriteNewData(user);

MessageBox.Show("Пароль успешно изменён!", "Поздравляем!" ,MessageBoxButton.OK, MessageBoxImage.Information);

new MainWindow().Show();

this.Close();

Описание теста: Неуспешное восстановление пароля с неправильной ссылкой из письма.

 if (сodeConfirmationsTextBox.Text.Length == 0)

{

MessageBox.Show("Не введён код подтверждения!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (codeConfirmation.ToString() != сodeConfirmationsTextBox.Text)

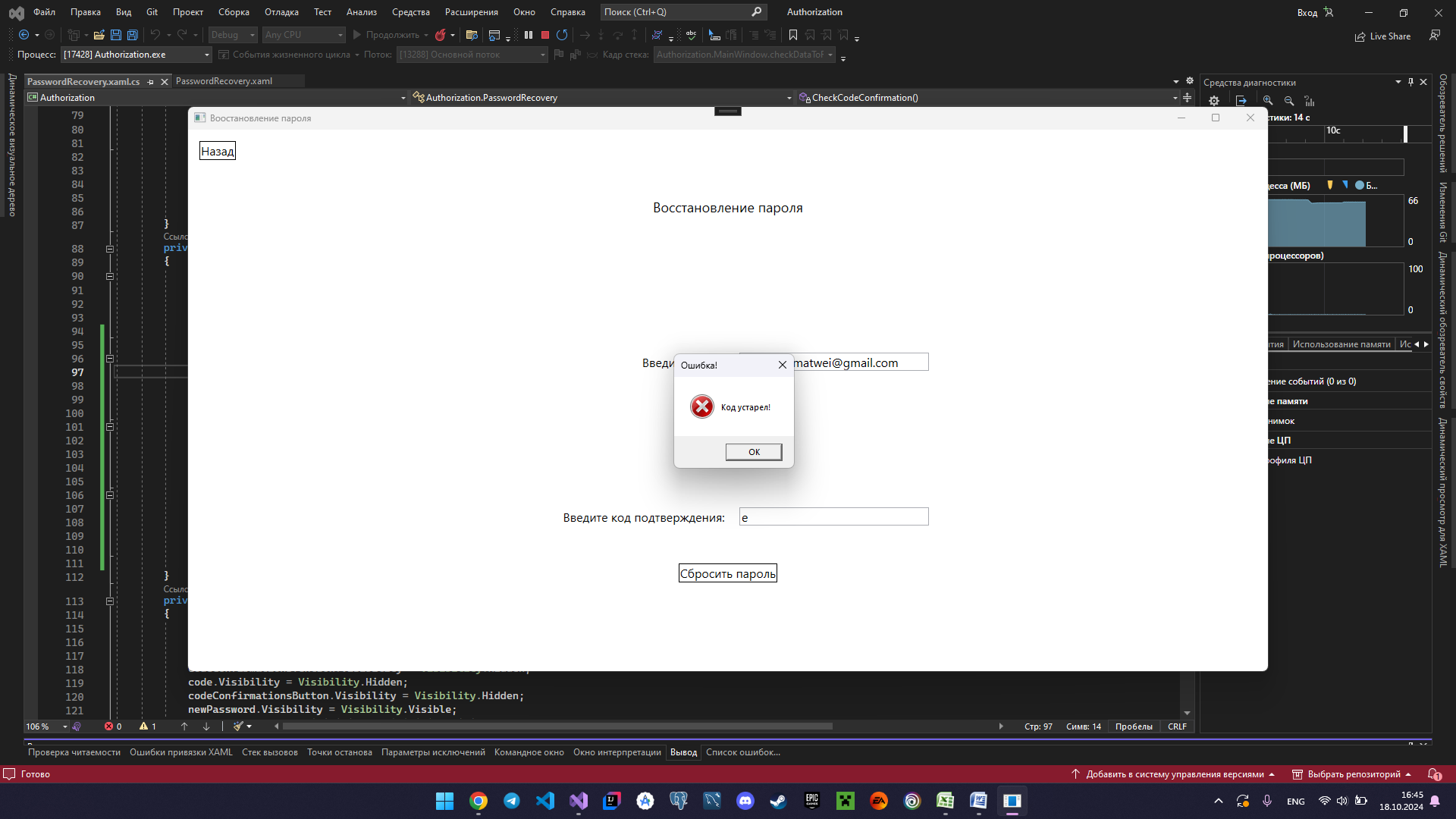
{

MessageBox.Show("Не правильно введён код подтверждения!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

Описание теста: Неуспешное восстановление пароля с неправильной ссылкой из письма.



if (DateTime.Now.Subtract(timeToSend).Minutes > 4)

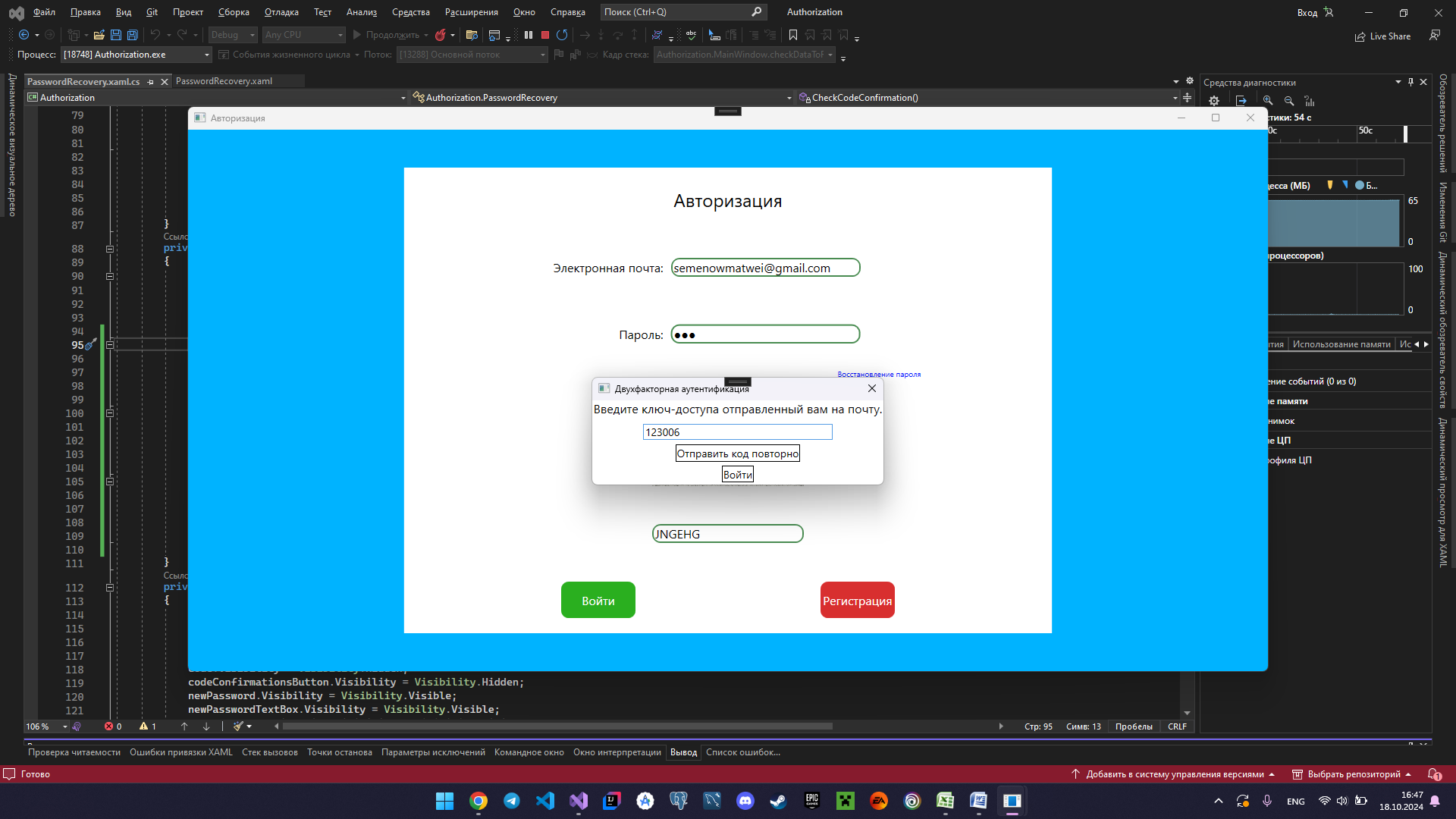
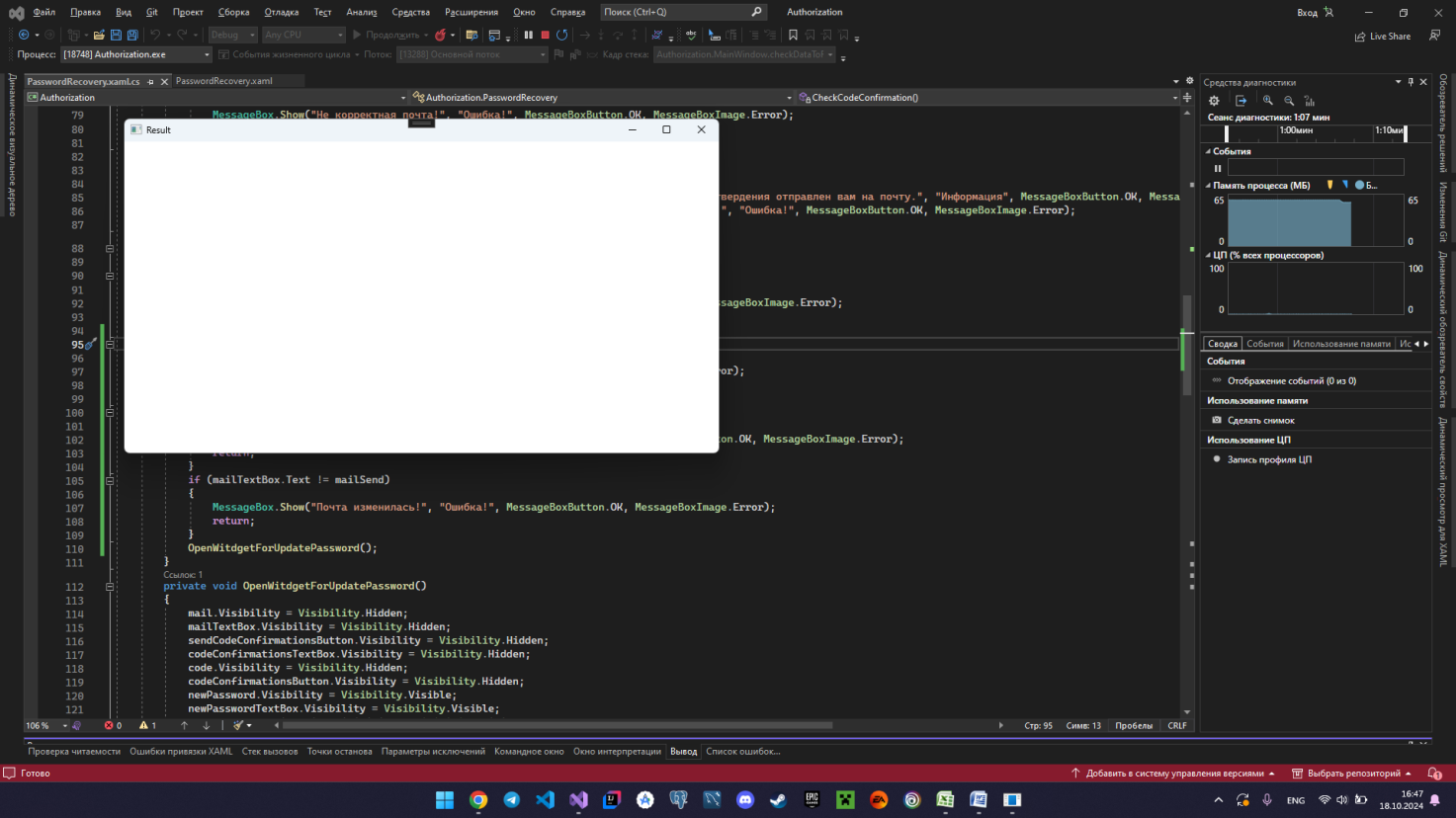
{

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

return;

}

Описание теста: Успешная аутентификация с двухфакторной аутентификацией.

private bool CheckCode()

{

return codeConfirmationsTextBox.Text == Convert.ToString(codeConfirmations);

}

private void CheckCodeInput()

{

if (!CheckCode())

{

MessageBox.Show("Не правильно введён код подтверждения.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

trying++;

VzlomAcc();

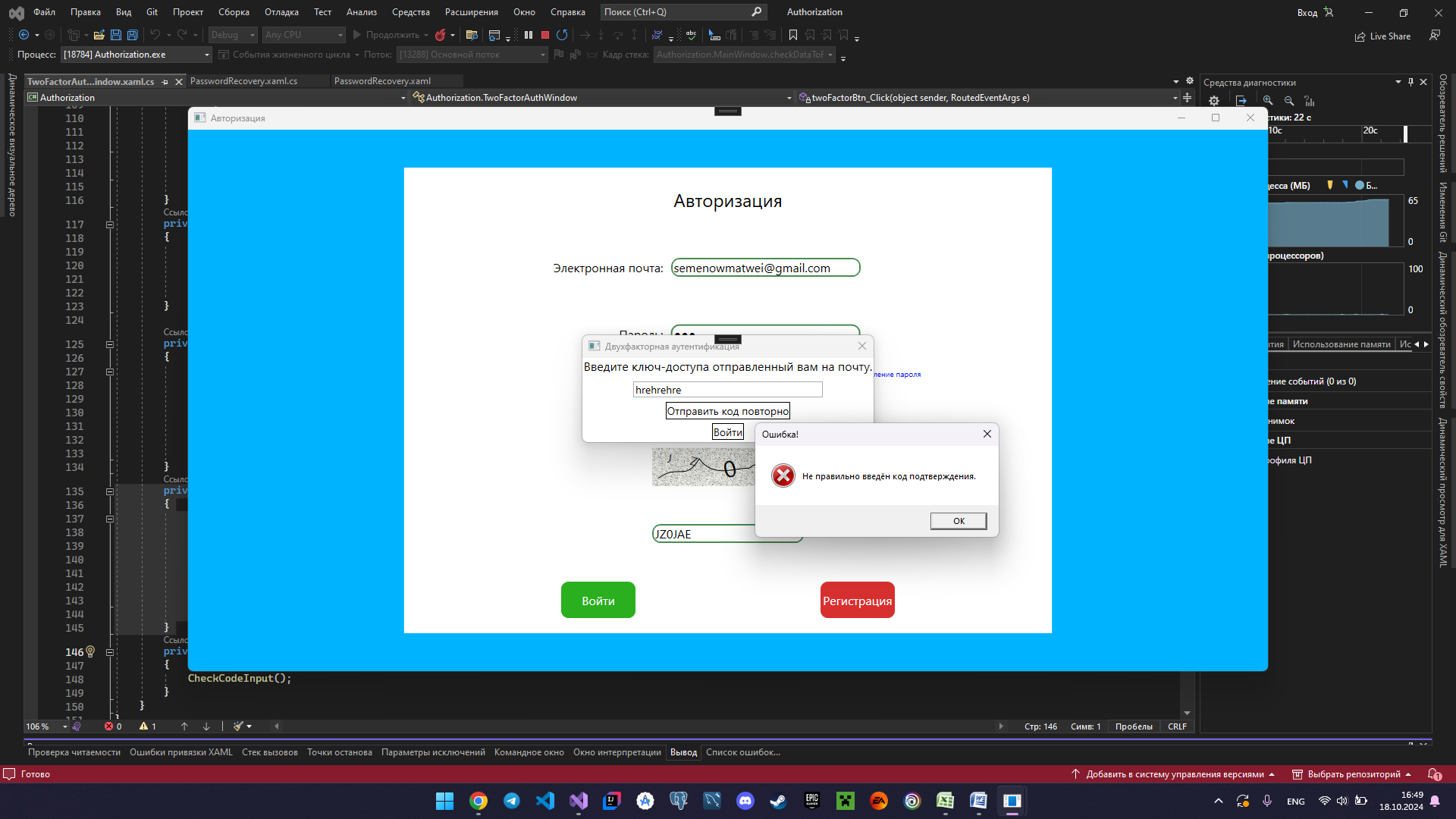
return;

}

this.DialogResult = true;

}

Описание теста: Неуспешная аутентификация с неправильным кодом подтверждения.

 private bool CheckCode()

{

return codeConfirmationsTextBox.Text == Convert.ToString(codeConfirmations);

}

private void CheckCodeInput()

{

if (!CheckCode())

{

MessageBox.Show("Не правильно введён код подтверждения.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

trying++;

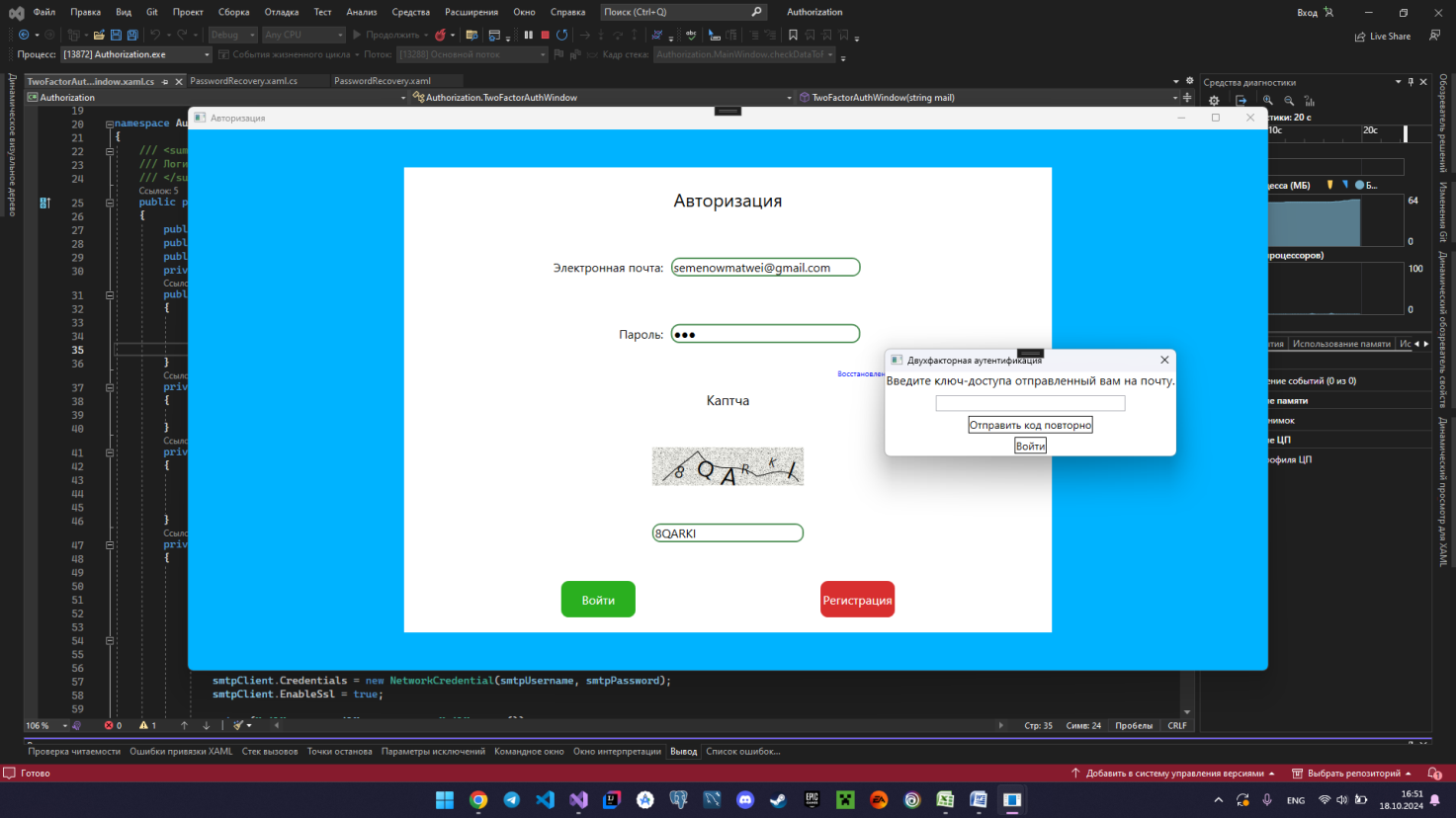
VzlomAcc();

return;

}

this.DialogResult = true;

}

Описание теста: Успешная аутентификация с правильной капчей.  private bool CheckToCorrectlyInputCaptcha(string captchaInImage, string inputCaptcha)

{

return captchaInImage == inputCaptcha;

}

if (!CheckToCorrectlyInputCaptcha(captchaImage.CaptchaText, captchaTextBox.Text))

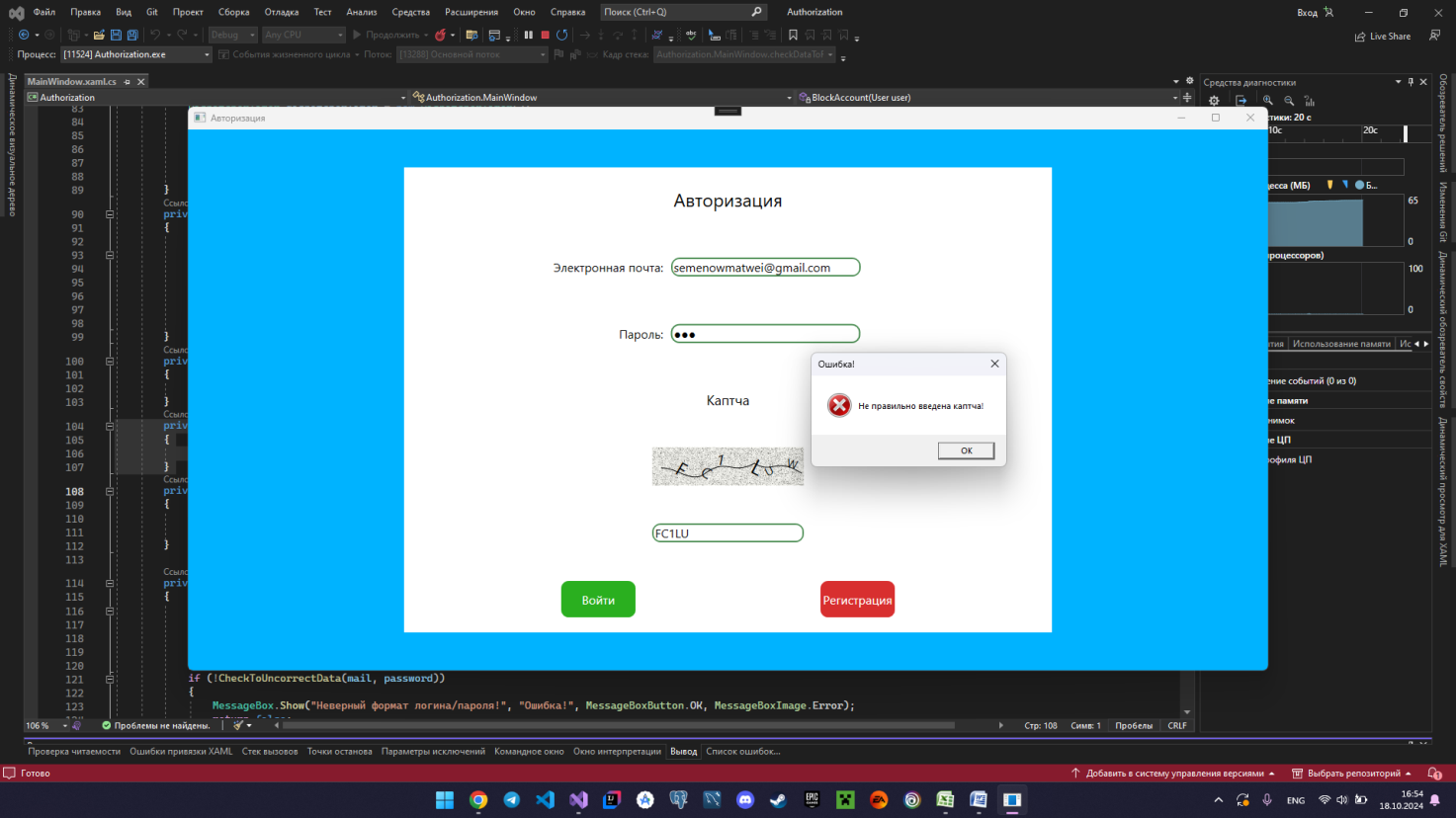
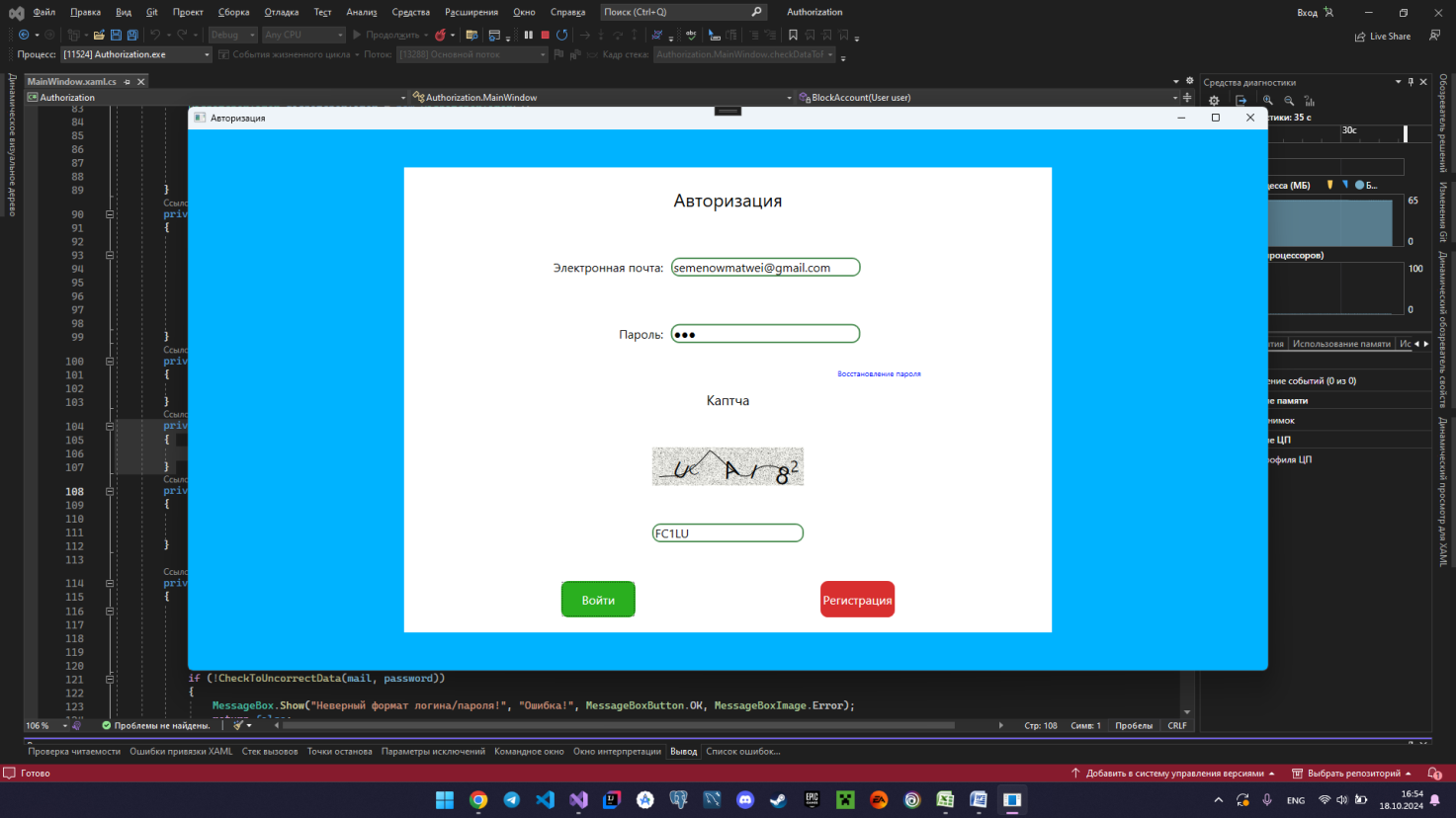
{

MessageBox.Show("Не правильно введена каптча!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

captchaImage.CreateCaptcha(Captcha.LetterOption.Alphanumeric, 6);

return false;

}

Описание теста: Неуспешная аутентификация с неправильной капчей.   private bool CheckToCorrectlyInputCaptcha(string captchaInImage, string inputCaptcha)

{

return captchaInImage == inputCaptcha;

}

if (!CheckToCorrectlyInputCaptcha(captchaImage.CaptchaText, captchaTextBox.Text))

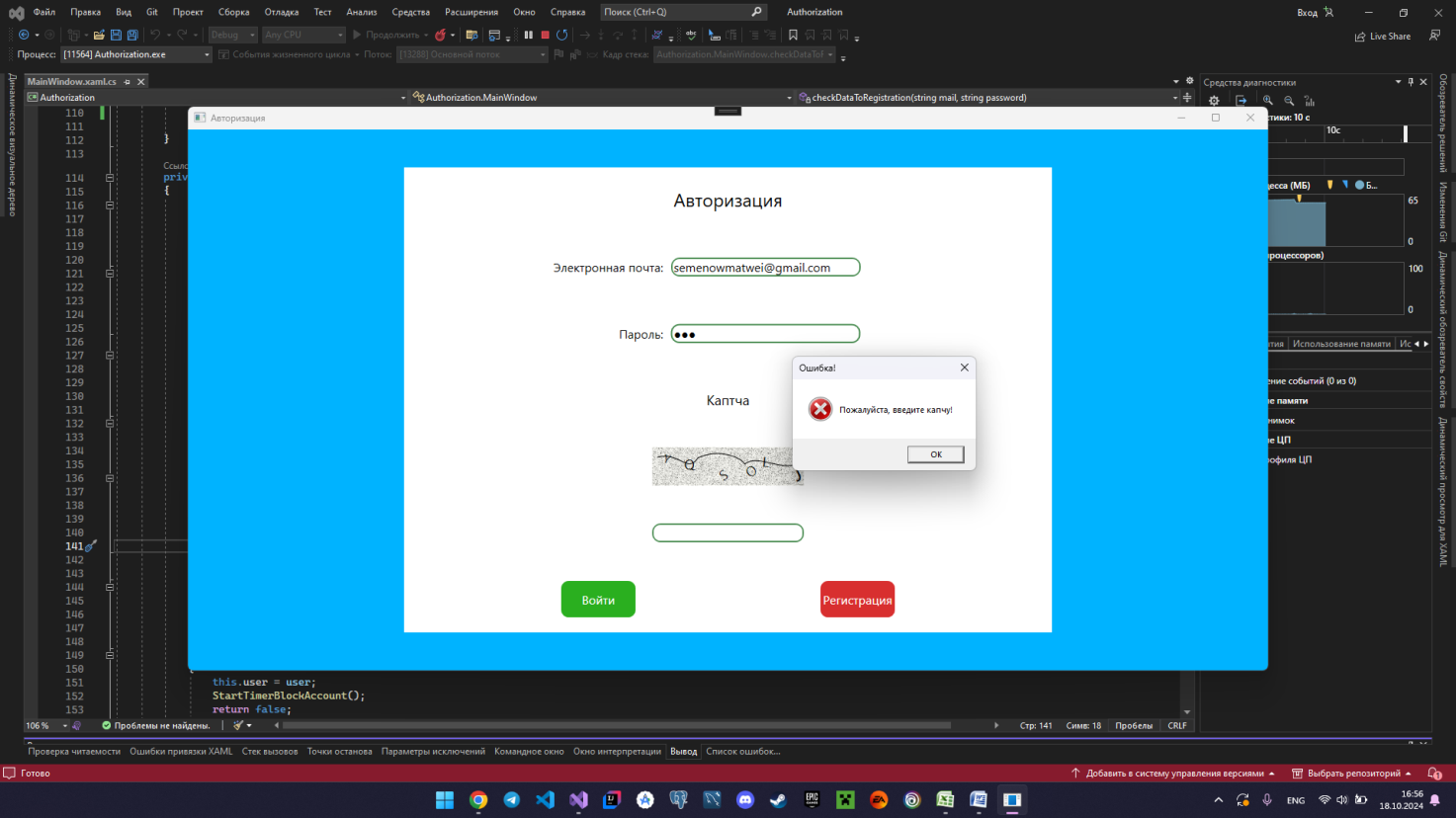
{

MessageBox.Show("Не правильно введена каптча!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

captchaImage.CreateCaptcha(Captcha.LetterOption.Alphanumeric, 6);

return false;

}

Описание теста: Неуспешная аутентификация с пустой капчей. 

private bool CheckToInputCaptcha(string inputCaptcha)

{

return inputCaptcha.Length > 0;

}

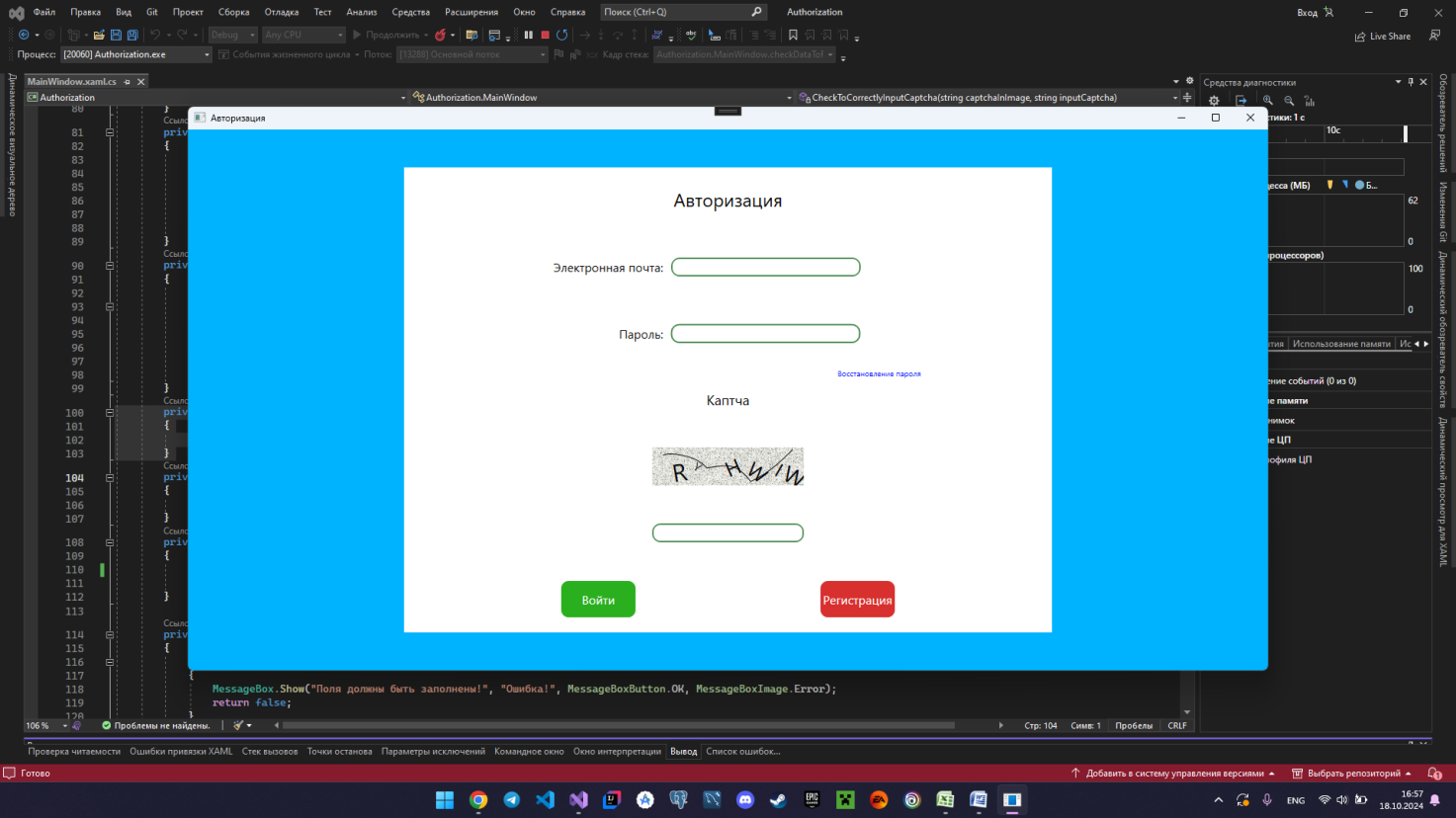
if (!CheckToInputCaptcha(captchaTextBox.Text))

{

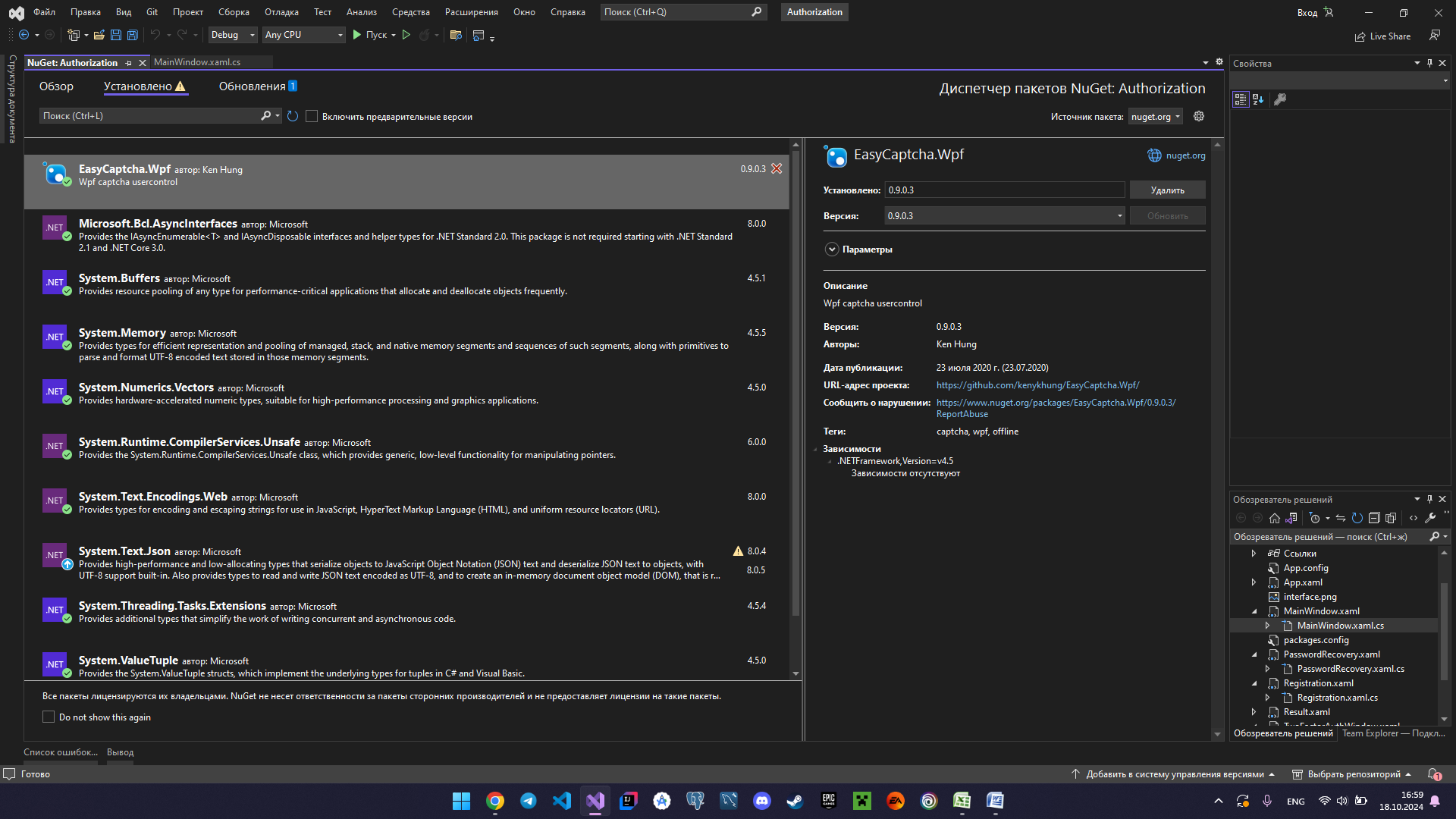
MessageBox.Show("Пожалуйста, введите капчу!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return false;

}

Описание теста: Неуспешная аутентификация с автоматически сгенерированной капчей (для имитации бота). 

Отдельная библиотека для создания каптчи: EasyCaptcha.Wpf.



using EasyCaptcha.Wpf;

captchaImage.CreateCaptcha(Captcha.LetterOption.Alphanumeric, 6);