Окно авторизации имеет 3 поля для ввода данных. Поле логин изначально доступно для ввода информации. Поле пароль недоступно, пока пользователь не введёт логин и не пройдёт проверка на существование данного логина. Поле для ввода кода и кнопка, которая выдаёт новый код, не видимы для пользователя, пока не пройдёт проверка на правильность логина и пароля.

Рисунок 3.1 Окно авторизации.

<Grid Background="#FFD2EAFF">

<StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock TextWrapping="Wrap" Text="Логин"/>

<TextBox x:Name="TxbLogin" TextWrapping="Wrap" Text="" Width="176" KeyUp="Login\_KeyUp">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

</TextBox>

<TextBlock TextWrapping="Wrap" Text="Пароль"/>

<PasswordBox x:Name="TxbPassword" Width="176" KeyUp="Password\_KeyUp" IsEnabled="False">

<PasswordBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</PasswordBox.Resources>

</PasswordBox>

<TextBlock Text="Код" Visibility="Collapsed" x:Name="CodeBlock"></TextBlock>

<TextBox x:Name="CodeBox" TextWrapping="Wrap" Text="" Width="176" Visibility="Collapsed">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

</TextBox>

<Button Content="Новый код" x:Name="RefreshBtn" Click="Refresh" Width="120" Visibility="Collapsed">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button Content="Войти" x:Name="Autorization" Click="Autorization\_Click" Width="176">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</StackPanel>

</Grid>

Кнопка “Войти” делает переход на основное окно, которое поделено на 2 части. Верхняя часть окна имеет логотип и 3 кнопки переходов между страницами. Нижняя часть окна это сама область в которой отображаются страницы.

Рисунок 3.2 Основное окно.

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="70"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel

HorizontalAlignment="Left"

Orientation="Horizontal"

VerticalAlignment="Center" Width="700" Height="40" Margin="100,0,0,0">

<Button x:Name="Machine" Content="Станки" Width="120" Click="Machine\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button x:Name="Test" Content="Тестирование" HorizontalAlignment="Left" Width="120" Click="Test\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button x:Name="Repair" Content="Ремонт" HorizontalAlignment="Left" Width="120" Click="Repair\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</StackPanel>

<Frame x:Name="MyFrame" Grid.Row="1" Grid.ColumnSpan="2" NavigationUIVisibility="Hidden"/>

<Image Margin="20,0,0,0" Source="/pngwing.com.png" Stretch="Fill" Height="50" VerticalAlignment="Center" HorizontalAlignment="Left" Width="50"/>

</Grid>

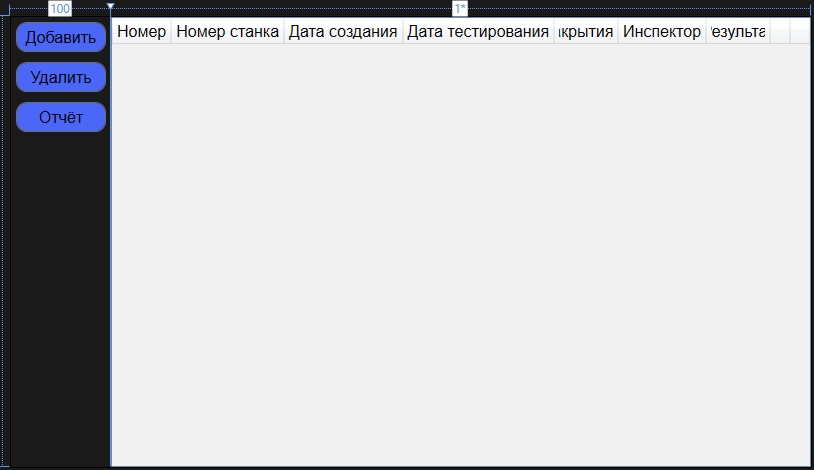
При нажатии любой из 3 кнопок в верхней части экрана, произойдёт отображение страницы. На рисунке (3.3) страница с таблицой “Тестирование” и 3 кнопки в левой части экрана для взаимодействия с этой таблицей, ещё есть 2 кнопки в самой таблице: “Редактировать” и “Закрыть заявку”. Каждая колонка таблицы привязана к атрибуту в базе данных. Кнопки слева от таблицы, скрыты для обычного пользователя, у инжинера нету кнопок: “Удалить” и “Отчёт”, а администратор видит все 3 кнопки.

Рисунок 3.3 Страница тестирование.

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<DataGrid

AutoGenerateColumns="False"

x:Name="TestGrid"

IsReadOnly="True" Grid.Column="2"

>

<DataGrid.Columns >

<DataGridTextColumn Header="Номер" Width="auto" Binding="{Binding Id}"/>

<DataGridTextColumn Header="Номер станка" Width="auto" Binding="{Binding IdMachine}"/>

<DataGridTextColumn Header="Дата создания" Width="auto" Binding="{Binding DateOfCreate, StringFormat='{}{0:dd.MM.yyyy}'}"/>

<DataGridTextColumn Header="Дата тестирования" Width="auto" Binding="{Binding DateOfTesting, StringFormat='{}{0:dd.MM.yyyy}'}"/>

<DataGridTextColumn Header="Дата закрытия заявки" Width="\*" Binding="{Binding DateOfClose, StringFormat='{}{0:dd.MM.yyyy}'}" />

<DataGridTextColumn Header="Инспектор" Width="auto" Binding="{Binding User.FullName}" />

<DataGridTextColumn Header="Результат" Width="\*" Binding="{Binding Result}">

<DataGridTextColumn.ElementStyle>

<Style TargetType="{x:Type TextBlock}">

<Setter Property="TextWrapping" Value="Wrap"/>

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Name="BtnEdit" HorizontalAlignment="Left" Click="BtnEdit\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="Close" Content="Закрыть заявку" HorizontalAlignment="Left" Width="120" Click="Close\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Row="0">

<Button x:Name="Add" Content="Добавить" Width="90" Click="Add\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button x:Name="Delete" Content="Удалить" Width="90" Click="Delete\_Click" VerticalAlignment="Top">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button Width="90" Content="Отчёт" Name="Export" Click="Export\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

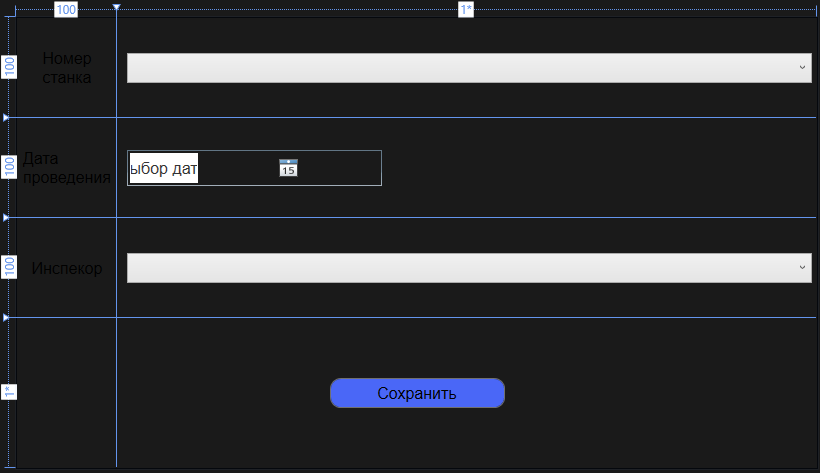
</Button.Resources>

</Button>

</StackPanel>

</Grid>

При нажатии кнопки “Добавить” открывается страница, вместо предыдущей. На ней имеются 3 элемента: 2 “ComboBox” и “DatePicker”. Комбо боксы нужны для выбора информации из других таблиц, для заполнения новой строки в таблице. Каждый элемент привязан к атрибуту в базе данных. При нажатие на кнопку “Редактировать”, появляется эта же страница, только с заполненными данными той строки, на которой была нажата кнопка. После нажатия кнопки сохранить, происходит проверка на заполненность полей, если какое то поле не заполненно, на экране появляется “MessageBox” с указанием на строку в которой не заполненное поле. Если все поля заполненны корректно, происходит добавление в базу данных.

Рисунок 3.4 Страница добавление и редактирования.

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock TextWrapping="Wrap" HorizontalAlignment="Center" VerticalAlignment="Center" Text="Номер станка"/>

<TextBlock Grid.Row="1" TextWrapping="Wrap" HorizontalAlignment="Center" VerticalAlignment="Center"><Run Text="Дата"/><Run Language="ru-ru" Text=" проведения"/></TextBlock>

<TextBlock Grid.Row ="2" TextWrapping="Wrap" HorizontalAlignment="Center" VerticalAlignment="Center" Text="Инспекор"/>

<ComboBox SelectedItem="{Binding Machine}" Grid.Column="1" x:Name="CmbMachine" DisplayMemberPath="Id" Margin="10,0,5,0"/>

<DatePicker SelectedDate ="{Binding DateOfTesting}" Grid.Column="1" Margin="10,0,435,0" Grid.Row="1" DisplayDateStart="01.01.2023" VerticalAlignment="Center" Name="DateOfTesting"/>

<ComboBox SelectedItem="{Binding User}" Grid.Row="2" Grid.Column="1" x:Name="CmbUsers" DisplayMemberPath="FullName" Margin="10,0,5,0"/>

<Button Grid.ColumnSpan="2" Grid.Row="3" Content="Сохранить" HorizontalAlignment="Center" Name="AddEdit" Click="AddEdit\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</Grid>

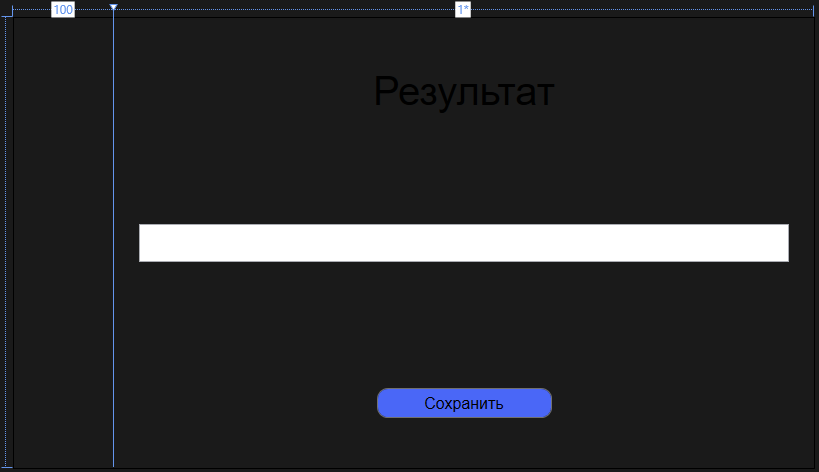
При нажатие кнопки “Закрыть заявку”, открывается страница. Которая имеет одно поле. Так же с привязкой к атрибуту в базе данных. Это поле так же имеет проверку на заполненность поля и при нажатии кнопки “Сохранить” сохраняет изменения в базе данных.

Рисунок 3.5 Страница закрытия заявки.

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Center" Orientation="Vertical" Width="700" Grid.Column="1">

<TextBox Text="{Binding Result}" TextWrapping="Wrap" x:Name="NameTxb" Width="650"/>

</StackPanel>

<TextBlock Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0 50 0 0" FontSize="40" TextWrapping="Wrap" Text="Результат" />

<Button Grid.Column="1" Content="Сохранить" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="0 0 0 50" x:Name="Save" Click="Save\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</Grid>

На основе этих окон, были сделаны другие страницы для добавления/редактирования , страница закрытия заявок и страницы с таблицами данных.

## 3.2 Программирование информационной системы

Окно авторизации (рисунок 3.1) имеет такую логику.

Этот класс предназначен для хранения информации о авторизованном пользователи.

public static class Globals

{

public static int UserRole;

public static User userinfo { get; set; }

}

При нажатии на кнопку “Войти” срабатывает этот код, который делает проверку на соответствие логина, пароля и кода. Переводит на основное окно (рисунок 3.2) и передаёт информацию о авторизированном пользователе в класс Globals.

private void Autorization\_Click(object sender, RoutedEventArgs e)

{

using (var db = new StankiEntities())

{

var auth = db.User.AsNoTracking().FirstOrDefault(m => m.Login == TxbLogin.Text && m.Password == TxbPassword.Password);

if (auth != null & code == CodeBox.Text)

{

timer.Stop();

Globals.UserRole = auth.RoleId;

Globals.userinfo = auth;

MainWindow main = new MainWindow();

main.Show();

Close();

}

else

{

MessageBox.Show("Неверный код, повторите попытку!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

timer.Stop();

}

}

}

Страница с заявками на тестирование (рисунок 3.3) имеет следующую логику.

Разграничение прав доступа, выглядит таким образом, что если авторизованный пользователь имеет роль с значением 1, ему видны все кнопки, роль с значением 2, не видит и не может взаимодействовать с кнопками: удаления и формирования отчёта и роль с значением 3, не видит и не может взаимодействовать со всеми кнопками левой части экрана.

public TestPage()

{

InitializeComponent();

if (Authorization.Globals.UserRole == 1)

{

Delete.Visibility = Visibility.Visible;

}

else if(Authorization.Globals.UserRole == 3)

{

Delete.Visibility = Visibility.Collapsed;

Add.Visibility = Visibility.Collapsed;

Export.Visibility = Visibility.Collapsed;

}

else

{

Export.Visibility = Visibility.Collapsed;

Delete.Visibility = Visibility.Collapsed;

}

}

При нажатие кнопки редактирования, происходит данный участок кода, который передаёт данные из таблицы в форму редактирования и заполняет все поля который связаны с помощью {Binding}.

private void BtnEdit\_Click(object sender, RoutedEventArgs e)

{

Manager.MyFrame.Navigate(new AddEditTest((sender as Button).DataContext as TestRequests));

}

На данном участке кода, происходит удаление выбранных данных таблицы.

private void Delete\_Click(object sender, RoutedEventArgs e)

{

var testForRemoving = TestGrid.SelectedItems.Cast<TestRequests>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {testForRemoving.Count()} элементов?", "Внимание",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

StankiEntities.GetContext().TestRequests.RemoveRange(testForRemoving);

StankiEntities.GetContext().SaveChanges();

MessageBox.Show("Данные удалены!");

TestGrid.ItemsSource = StankiEntities.GetContext().TestRequests.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

При нажатии на кнопку формирования отчёта. Выполняется данный участок кода, который создаёт новый Word документ, формирует в нём таблицу и заполняет её.

private void Export\_Click(object sender, RoutedEventArgs e)

{

var allRequest = StankiEntities.GetContext().TestRequests.ToList();

var application = new Word.Application();

Word.Document document = application.Documents.Add();

Word.Paragraph userParagraph = document.Paragraphs.Add();

Word.Range userRange = userParagraph.Range;

userRange.Text = "Заявки на тестирование";

userRange.InsertParagraphAfter();

Word.Paragraph tableParagraph = document.Paragraphs.Add();

Word.Range tableRange = tableParagraph.Range;

Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count()+1, 6);

paymentsTable.Borders.InsideLineStyle = paymentsTable.Borders.OutsideLineStyle

= Word.WdLineStyle.wdLineStyleSingle;

paymentsTable.Range.Cells.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

Word.Range cellRange;

cellRange = paymentsTable.Cell(1, 1).Range;

cellRange.Text = "Номер заявки";

cellRange = paymentsTable.Cell(1, 2).Range;

cellRange.Text = "Номер станка";

cellRange = paymentsTable.Cell(1, 3).Range;

cellRange.Text = "Дата Создания";

cellRange = paymentsTable.Cell(1, 4).Range;

cellRange.Text = "Создатель заявки";

cellRange = paymentsTable.Cell(1, 5).Range;

cellRange.Text = "Инженер";

cellRange = paymentsTable.Cell(1, 6).Range;

cellRange.Text = "Результат";

paymentsTable.Rows[1].Range.Bold = 1;

paymentsTable.Rows[1].Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

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

{

var currentCategory = allRequest[i];

if (currentCategory.Result != null)

{

cellRange = paymentsTable.Cell(i + 2, 1).Range;

cellRange.Text = Convert.ToString(currentCategory.Id);

cellRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

cellRange = paymentsTable.Cell(i + 2, 2).Range;

cellRange.Text = Convert.ToString(currentCategory.IdMachine);

cellRange = paymentsTable.Cell(i + 2, 3).Range;

cellRange.Text = currentCategory.DateOfCreate.ToString("dd.MM.yyyy");

cellRange = paymentsTable.Cell(i + 2, 4).Range;

cellRange.Text = Convert.ToString(currentCategory.Creator);

cellRange = paymentsTable.Cell(i + 2, 5).Range;

cellRange.Text = Convert.ToString(currentCategory.User.FullName);

cellRange = paymentsTable.Cell(i + 2, 6).Range;

cellRange.Text = Convert.ToString(currentCategory.Result);

}

}

application.Visible = true;

}

Страница с добавлением/редактированием (рисунок 3.4) имеет следующую логику.

В данном участке кода происходит создания новой строки в базе данных.

private TestRequests \_currentTest = new TestRequests();

Если передаётся какое то значение “selectedTest” со страницы с заявками, идёт присвоение текущего теста к выбранному тесту.

public AddEditTest(TestRequests selectedTest)

{

InitializeComponent();

if (selectedTest != null)

{

\_currentTest = selectedTest;

}

else

{

\_currentTest.DateOfTesting = DateTime.Now;

}

DataContext = \_currentTest;

CmbMachine.ItemsSource = StankiEntities.GetContext().Machine.ToList();

CmbUsers.ItemsSource = StankiEntities.GetContext().User.ToList();

}

В данной части кода, происходит предотвращение ошибок, допущенных пользователем. Сохранение новой заявки или редактирование старой. Так же идёт автоматическое заполнение значения “Creator” и “DateOfCreate”. Поле создателя заявки заполняется на основе авторизированного пользователя, а дата создания заполняется на основе сегодняшней даты.

private void AddEdit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

var CurrentMachine = CmbMachine.SelectedItem as Machine;

var CurrentUser = CmbUsers.SelectedItem as User;

\_currentTest.DateOfCreate = DateTime.Now;

var Creator = Authorization.Globals.userinfo.FullName;

\_currentTest.Creator = Creator;

if (CurrentMachine == null)

errors.AppendLine("Выберите станок");

if (DateOfTesting.SelectedDate == null)

errors.AppendLine("Выберите дату");

if (CurrentUser == null)

errors.AppendLine("Выберите инспектора");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (\_currentTest.Id >= 0)

StankiEntities.GetContext().TestRequests.AddOrUpdate(\_currentTest);

try

{

StankiEntities.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Manager.MyFrame.GoBack();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

Страница с закрытием заявки (рисунок 3.5) имеет такую же логику, как и редактирование заявки.

Остальные страницы сделаны по аналогии описанных.