Спецификация разрабатываемого модуля

Модуль предназначен для автоматизации процесса учета и управления заявками на ремонт компьютерной техники. Он включает в себя разработку алгоритмов обработки заявок, пользовательского интерфейса и анализ технического задания

Компоненты модуля:

* Алгоритмы
* Пользовательский интерфейс

Основные функции модуля:

* Добавление заявки
* Редактирование заявки
* Отслеживание статуса заявки
* Назначение ответственных за выполнение работ
* Расчет статистики

Входные данные:

* Данные пользователя
* Данные заявки

Выходные данные:

* Информация о заявке
* Отчетность

Технические характеристики:

* Операционная система: Windows 10 или выше.
* Платформа разработки: .NET Framework.
* База данных: SQL Server.
* Язык программирования: C#.
* Инструменты разработки: Visual Studio.

Отладка с помощью специализированных программных средств

1. Точка остановы
2. Пошаговое выполнение
3. Окно "Locals": В этом окне отображаются все локальные переменные текущего контекста выполнения.
4. Окно "Call Stack": В этом окне отображается стек вызовов, который показывает последовательность вызовов функций, приведшую к текущей точке выполнения.

Тест-кейсы

|  |  |
| --- | --- |
| Test Case # | 4 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Тестирование кнопки «Войти» |
| Резюме испытания | Необходимо, чтобы обработка нажатия на кнопку была корректна |
| Шаги тестирования | 1. Открыть форму авторизации |
| Данные тестирования | Правильные данные:   * Логин: buldakova04e@gmail.com * Пароль: 123456   Неправильные данные пользователя:   * Логин: 123 * Пароль: 123 |
| Ожидаемый результат | Обработка введенных в поле данных и переход на следующую страницу при вводе верного логина и пароля. В случае некорректного ввода данных возвращение на предыдущую страницу. |
| Фактический результат | Обработка введенных в поле данных и переход на следующую страницу при вводе верного логина и пароля. В случае некорректного ввода данных возвращение на предыдущую страницу. |
| Предпосылки | Запустить приложение |
| Постусловия | Должен быть совершен переход на соответствующую коду страницу |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

Предложения по модификации ПО на основании дополнения к ТЗ

1. Добавление роли менеджера по качеству: Добавить менеджера в базу данных, добавить разделы для менеджера в пользовательском интерфейсе, реализовать функционал продления сроков выполнения заявок и консультаций через комментарии в системе
2. Генерация QR-кодов: Добавить возможность генерации QR-кодов для каждой завершенной заявки.
3. Отслеживание качества работы: Внедрить форму опроса для клиентов, включающую основные критерии оценки качества ремонта.

Эти предложения помогут улучшить функциональность программного модуля, сделав его более удобным и полезным для всех участников процесса.

Руководство системного программиста

1. . Общие сведения о программе
   1. Назначение и функции программы

Программный продукт предназначен для управления заявками на ремонт. Основные функции программы включают:

* Создание и управление заявками на ремонт.
* Назначение мастеров для выполнения ремонтных работ.
* Контроль и координация ремонтных работ.
* Добавление новой роли – менеджера по качеству, консультирующего мастеров и продлевающего сроки выполнения заявок.
  1. Технические и программные средства

Программа выполняется на сервере с операционной системой Windows и использует следующие программные средства:

* Операционная система: Windows 10 или выше.
* Платформа разработки: .NET Framework.
* База данных: SQL Server.
* Язык программирования: C#.
* Инструменты разработки: Visual Studio.
* Средства генерации QR-кодов: библиотека qrcoder

1. Структура программы
   1. Составные части программы

Программа состоит из следующих модулей:

* Модуль управления заявками: Обработка и управление заявками на ремонт.
* Модуль назначений: Назначение мастеров и менеджеров по качеству.
* Модуль генерации QR-кодов: Создание QR-кодов для оценки качества работы.
* База данных: Хранение информации о заявках, мастерах, менеджерах и отзывах.
  1. Связи между составными частями
* Модуль управления заявками взаимодействует с базой данных для создания, обновления и удаления заявок.
* Модуль назначений связывает заявки с мастерами и менеджерами по качеству.
* Модуль генерации QR-кодов интегрирован с модулем управления заявками для создания QR-кодов при завершении заявок.

1. Настройка программы
   1. Настройка на состав технических средств

* Установить необходимые зависимости .
* Настроить конфигурационный файл для подключения к базе данных

1. Проверка программы
   1. Описание способов проверки

Контрольные примеры: Использовать заранее подготовленные наборы данных для проверки функциональности.

1. Дополнительные возможности
   1. Описание дополнительных функциональных возможностей

* Роль менеджера по качеству: Консультирование мастеров, продление сроков выполнения заявок.
* Оценка качества работы: Генерация QR-кодов для опросов клиентов.

Кью ар

var generator = new QRCodeGenerator();

var qrCodeData = generator.CreateQrCode("123123", QRCodeGenerator.ECCLevel.L);

var qrCode = new QRCode(qrCodeData);

var picture = qrCode.GetGraphic(100);

using (var ms = new MemoryStream())

{

picture.Save(ms, ImageFormat.Png);

ms.Seek(0, SeekOrigin.Begin);

BitmapImage bitmapimg = new BitmapImage();

bitmapimg.BeginInit();

bitmapimg.StreamSource = ms;

bitmapimg.CacheOption = BitmapCacheOption.OnLoad;

bitmapimg.EndInit();

imageQr.Source = bitmapimg;

}

Страница авторизации

1

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition Width="6\*"></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="6\*"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<StackPanel Grid.Row="1" Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Center">

<TextBlock FontSize="50" Text="Авторизация" HorizontalAlignment="Center"></TextBlock>

</StackPanel>

<StackPanel Grid.Column="1" Grid.Row="1" VerticalAlignment="Center" Width="500" >

<Label Content="Логин"></Label>

<TextBox ToolTip="Login" x:Name="txtLogin" Height="35" ></TextBox>

<Label Content="Пароль"></Label>

<PasswordBox x:Name="txtPass" Height="35"></PasswordBox>

<StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="10">

<Button Content="Войти" Width="160" Height="35" FontSize="20" Click="LogIn"></Button>

<Button Content="нет аккаунта?" Background="White" Width="160" Height="35" FontSize="20"

Foreground="Black" Click="GoToRegPage"></Button>

</StackPanel>

</StackPanel>

</Grid>

2

private void LogIn(object sender, RoutedEventArgs e)

{

var data = App.Connection.Auth.Where(z => z.Login == txtLogin.Text && z.Password == txtPass.Password).FirstOrDefault();

user = App.Connection.User.Where(x => x.IdAuth == data.IdAuth).FirstOrDefault();

if (user != null)

{

UserData.IdUser = user.IdUser;

UserData.IdRole = user.IdRole;

MessageBox.Show("Успешно", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

if (user.IdRole == 1)

{

NavigationService.Navigate(new UserPage());

}

else if (user.IdRole == 2)

{

NavigationService.Navigate(new OperatorPage());

}

else if (user.IdRole == 3)

{

NavigationService.Navigate(new MasterPage());

}

}

else

{

MessageBox.Show("Неверный логин или пароль", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Главная страница

1

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="30"></ColumnDefinition>

<ColumnDefinition Width="2\*"></ColumnDefinition>

<ColumnDefinition Width="30"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="10"></RowDefinition>

<RowDefinition Height="70"></RowDefinition>

<RowDefinition Height="90"></RowDefinition>

<RowDefinition Height="7\*"></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition Height="20"></RowDefinition>

</Grid.RowDefinitions>

<StackPanel Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" Orientation="Horizontal" HorizontalAlignment="Center">

<TextBlock Text="Страница клиента" FontSize="40" HorizontalAlignment="Center"></TextBlock>

</StackPanel>

<StackPanel Grid.Row="4" Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="5" Orientation="Horizontal">

<Button x:Name="AddRequest" Margin="5" Click="GoToAddPage" Content="Добавить заявку" Width="220" Height="35" FontSize="20"></Button>

</StackPanel>

<StackPanel Grid.Column="1" Grid.Row="4" HorizontalAlignment="Left" Margin="10" Orientation="Horizontal">

<TextBlock x:Name="tbDataCount" Margin="5" Width="100" Height="35"></TextBlock>

</StackPanel>

<ScrollViewer HorizontalAlignment="Center" Grid.Row="3" Grid.Column="1">

<StackPanel>

<ListView x:Name="ListItems" ItemsSource="{Binding requestList}" Width="1150">

<ListView.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns="4"></UniformGrid>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<StackPanel>

<StackPanel Orientation="Horizontal" Width="260" Margin="5">

<Image Source="{Binding Photo}" Width="120" Height="120" />

<StackPanel Orientation="Vertical" Margin="5">

<TextBlock Text="{Binding Description}" FontSize="16" Width="140" TextWrapping="Wrap" />

<TextBlock Text="{Binding Model.Title}" FontSize="13" TextWrapping="Wrap" />

<TextBlock Text="{Binding Type.Title}" FontSize="13" TextWrapping="Wrap" />

<TextBlock Text="{Binding Status.Title}" FontSize="13" TextWrapping="Wrap" />

</StackPanel>

</StackPanel>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="5">

<Button Content="Редактировать" Click="EditButton\_Click" Tag="{Binding}" Margin="5"/>

<Button Content="Удалить" Click="DeleteButton\_Click" Tag="{Binding}" Margin="5"/>

</StackPanel>

</StackPanel>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackPanel>

</ScrollViewer>

<Grid Grid.Row="2" Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Orientation="Vertical" HorizontalAlignment="Right" Margin="0,0,50,0">

<TextBlock Margin="5" HorizontalAlignment="Center" FontSize="20">Статус</TextBlock>

<ComboBox FontSize="16" x:Name="cmbFilter" Width="160" Height="35" HorizontalAlignment="Left" SelectionChanged="Sort" DisplayMemberPath="Title" SelectedValuePath="id">

</ComboBox>

</StackPanel>

<StackPanel Grid.Row="0" Grid.Column="1" Orientation="Vertical">

<TextBlock Margin="5" HorizontalAlignment="Center">Поиск</TextBlock>

<TextBox x:Name="tbFinder" FontSize="18" Width="400" Height="35" HorizontalAlignment="Center" TextChanged="Find"></TextBox>

</StackPanel>

</Grid>

</Grid>

2

public partial class UserPage : Page

{

List<Request> sortProduct = new List<Request>();

List<Request> requestList = new List<Request>();

List<Status> statuses = new List<Status>();

Request request;

User user;

public UserPage()

{

InitializeComponent();

cmbFilter.SelectedIndex = 0;

request = ListItems.SelectedItem as Request;

}

private void GoToChange(object sender, RoutedEventArgs e)

{

if (ListItems.SelectedItem != null)

{

request = ListItems.SelectedItem as Request;

var changeProd = App.Connection.Request.Where(x => x.IdRequest == request.IdRequest).FirstOrDefault();

NavigationService.Navigate(new ItemPage(changeProd));

}

else

{

MessageBox.Show("Выберите товар для редактирования", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void Sort(object sender, SelectionChangedEventArgs e)

{

SortList();

}

private void SortList()

{

sortProduct = requestList.Where(x => x.Description.ToLower().Contains(tbFinder.Text.ToLower())).ToList();

if (cmbFilter.SelectedItem is Status selectedStatus) {

ListItems.ItemsSource = requestList.Where(item => item.Status.IdStatus == selectedStatus.IdStatus).ToList();

}

}

private void Find(object sender, TextChangedEventArgs e)

{

SortList();

}

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

{

if (sender is Button button && button.Tag is Request selectedItem)

{

// Навигация на страницу редактирования

NavigationService.Navigate(new ItemPage(selectedItem));

}

}

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

{

if (sender is Button button && button.Tag is Request selectedItem)

{

try

{

if (selectedItem != null)

{

var deleteService = App.Connection.Request.FirstOrDefault(x => x.IdRequest == selectedItem.IdRequest);

App.Connection.Request.Remove(deleteService);

App.Connection.SaveChanges();

MessageBox.Show("Товар успешно удалён!", "Информация!", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Выберите товар для удаления", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

catch

{

MessageBox.Show("Товар невозможно удалить", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

}

private async void Page\_Loaded(object sender, RoutedEventArgs e)

{

requestList = await db.GetListProduct();

ListItems.ItemsSource = requestList;

ListItems.Items.Refresh();

cmbFilter.ItemsSource = App.Connection.Status.ToList();

}

private void GoToAddPage(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new ItemPage(null));

}

}

Страница заявки

1

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="30"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="30"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="10"/>

<RowDefinition Height="70"></RowDefinition>

<RowDefinition Height="11\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" Orientation="Horizontal" HorizontalAlignment="Center">

<TextBlock Text="Заявка" FontSize="40" HorizontalAlignment="Center"></TextBlock>

</StackPanel>

<StackPanel Grid.Row="2" Grid.Column="1" >

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Column="0" Grid.RowSpan="2">

<TextBlock Text="Название" Margin="10" />

<TextBlock Text="Статус" Margin="10"/>

</StackPanel>

<StackPanel x:Name="DataProduct" Grid.Column="1" Grid.Row="0">

<TextBox x:Name="txtName" FontSize="20" Height="35" Text="{Binding Comment}" MaxLength="100" Margin="30,20,30,20"/>

<ComboBox x:Name="ListStatus" Margin="30,10,30,0" FontSize="20" DisplayMemberPath="Title" SelectedValuePath="IdStatus">

</ComboBox>

</StackPanel>

<StackPanel Grid.Row="2" Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0 0 0 0">

<Button Content="Сохранить" Width="150" Height="35" FontSize="20" Margin="0 0 10 0" Click="Save"/>

<Button Content="Отмена" Width="150" Height="35" FontSize="20" Margin="10 0 0 0" Click="GoBack"/>

</StackPanel>

</Grid>

</StackPanel>

</Grid>

2

public partial class ItemPage : Page

{

Request currentRequest;

User user;

public ItemPage(Request request)

{

InitializeComponent();

currentRequest = request;

LoadData();

ShowStatistics();

DataContext = currentRequest;

}

private void LoadData()

{

var statuses = App.Connection.Status.ToList();

ListStatus.ItemsSource = statuses;

if (currentRequest != null)

{

ListStatus.SelectedValue = currentRequest.IdStatus;

}

ListStatus.IsEnabled = UserData.IdRole != 1;

}

private void Save(object sender, RoutedEventArgs e)

{

if (currentRequest != null)

{

currentRequest.Description = txtName.Text;

currentRequest.Status = ListStatus.SelectedItem as Status;

App.Connection.SaveChanges();

MessageBox.Show("Успешное редактирование", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

var selectedStatus = ListStatus.SelectedItem as Status;

if (selectedStatus != null)

{

// Установка EndDate, если статус изменен на "завершено"

if (selectedStatus.Title == "Завершено" && currentRequest.Status.Title != "Завершено")

{

currentRequest.DateEnd = DateTime.Now;

}

currentRequest.Status = selectedStatus;

}

if (DeadLinePicker.SelectedDate.HasValue)

{

currentRequest.Deadline = DeadLinePicker.SelectedDate.Value;

}

App.Connection.SaveChanges();

}

else

{

var newRequest = new Request

{

Description = txtName.Text,

Status = ListStatus.SelectedItem as Status,

DateStart = DateTime.Now,

Deadline = DeadLinePicker.SelectedDate.Value

};

App.Connection.Request.Add(newRequest);

App.Connection.SaveChanges();

MessageBox.Show("Успешное создание новой заявки", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

NavigationService.Navigate(new UserPage());

}

}

void GoBack(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new UserPage());

}

public TimeSpan? CalculateAverageExecutionTime()

{

// Извлечение данных из базы данных

var completedRequests = App.Connection.Request

.Where(r => r.DateEnd != null && r.DateStart != default(DateTime))

.ToList();

if (completedRequests.Any())

{

// Выполнение вычислений на уровне приложения

var averageTicks = completedRequests.Average(r => (r.DateEnd.Value - r.DateStart).Ticks);

var averageTime = TimeSpan.FromTicks((long)averageTicks);

return averageTime;

}

return null; // Если нет завершенных заявок

}

public TimeSpan? CalculateExecutionTime(Request request)

{

if (request.DateStart != default(DateTime) && request.DateEnd != null)

{

return request.DateEnd.Value - request.DateStart;

}

return null; // Если EndDate не установлено или StartDate не установлен

}

private void ShowStatistics()

{

var averageExecutionTime = CalculateAverageExecutionTime();

if (averageExecutionTime != null)

{

AverageTime.Text = $"Среднее время выполнения: {averageExecutionTime.Value.Days} дней {averageExecutionTime.Value.Hours} часов {averageExecutionTime.Value.Minutes} минут";

}

else

{

AverageTime.Text = "Нет данных о выполненных заявках.";

}

}

}Регистрация

private void RegUser(object sender, RoutedEventArgs e)

{

if (txtName.Text == "" || txtLogin.Text == "" || txtPass.Text == "")

{

MessageBox.Show("Заполните все поля", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

try

{

var log = App.Connection.Authorization.Where(x => x.Login == txtLogin.Text).FirstOrDefault();

if (log != null)

{

MessageBox.Show("Пользователь с таким логином уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

Data.Authorization auth = new Data.Authorization(txtLogin.Text, txtPass.Text);

App.Connection.Authorization.Add(auth);

App.Connection.SaveChanges();

User user = new User(txtName.Text, 2, auth.IdAuth);

App.Connection.User.Add(user);

App.Connection.SaveChanges();

MessageBox.Show("Пользователь успешно зарегестрирован", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch

{

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

return;

}

NavigationService.Navigate(new AuthPage());

}}