1.СОСТАВЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ НА РАЗРАБОТКУ

МОДУЛЯ «ПРОГРАММИСТЫ В АПЗ» ИНФОРМАЦИОННОЙ СИСТЕМЫ «АПЗ»

Техническое задание — это важный документ проекта, который помогает команде разработчиков и клиенту лучше понять, каким должен стать готовый продукт. Оно устанавливает требования к IT-решению, такие как его функциональность, сценарии использования, технологическая основа и многое другое.

Стандартное ТЗ содержит следующие разделы:

Цель проекта. Здесь описываются цели и задачи, которые ставит перед собой продукт.

Категории пользователей. Описываются различные типы целевых аудиторий, которым адресован проект.

Содержание функционала. Подробно рассматриваются ключевые функции и сценарии использования продукта.

Интеграции. Указывается взаимодействие с внешними системами, например, с Яндекс.Картами.

Дизайн интерфейсов. Формулируются рекомендации по структуре и внешнему виду страниц, возможно, с учетом фирменного стиля заказчика.

Требования к безопасности. Описываются меры защиты, которые должны быть реализованы, такие как антивирусные программы и системы шифрования.

Технические аспекты разработки. Если у клиента есть предпочтения по технологиям, например, предпочитаемые языки программирования, это указывается в данном разделе.

Аппаратное обеспечение. Приводятся технические характеристики оборудования, на котором будет работать программа.

Для проектов, финансируемых государством или государственными компаниями, техзадание является обязательным документом. Его оформление осуществляется в соответствии с установленными стандартами, такими как ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы», который регламентирует контроль и подготовку к внедрению системы.

Основная задача составления ТЗ — точное определение ожиданий от проекта. Это обеспечивает уверенность команды и заказчика в том, что конечный продукт будет соответствовать всем установленным требованиям.

Процесс создания ТЗ состоит из нескольких этапов:

Заполнение анкеты. Клиент заполняет анкету, в которой указывает основные пожелания к проекту.

Сбор информации. Проводятся встречи с клиентом для уточнения деталей и фиксации всех требований, например, касающихся безопасности. Эти данные заносятся в ТЗ.

Консультации с экспертами. Сложные моменты обсуждаются с техническими специалистами, такими как системные архитекторы и аналитики.

Формирование ТЗ. Документ оформляется так, чтобы его легко понял любой участник проекта, включая клиента. Используются графики и пояснения для упрощения восприятия.

Утверждение. Подготовленное ТЗ предоставляется клиенту для ознакомления и утверждения. Обязательно присутствие представителя заказчика, который будет отвечать за приемку проекта. Все замечания вносятся в документ.

Подписание ТЗ. После внесения изменений финальная версия утверждается и подписывается обеими сторонами. Если ТЗ включено в контракт, его подписывают представители обеих сторон.

После утверждения ТЗ проводится расчет бюджета, учитывающий все этапы работы, включая разработку, проектирование, тестирование и сопровождение.

Согласно ГОСТ 34, ТЗ должно включать следующие разделы:

Общая информация;

Цель и задачи создания (модернизации) системы;

Описание объектов автоматизации;

Требования к системе;

Объем и содержание работ по созданию системы;

Процедура контроля и сдачи системы;

Требования к подготовке объекта автоматизации к эксплуатации;

Требования к документации;

Источники разработки.

Таким образом, каждое из перечисленных разделов играет важную роль в формировании полного и качественного технического задания. Правильное оформление и соблюдение стандартов ГОСТ 34 позволяют гарантировать соответствие проекта ожиданиям заказчика и минимизировать риски на всех этапах его реализации.

Разработанное техническое задание для приложениях о профессях на «АПЗ» представлено ниже.

Общие сведения. Приложение представляет собой информационный ресурс,

посвященный профессиям в области программирования. Он будет содержать разделы, описывающие основные аспекты профессии, включая обязанности, требования, обзор рынка труда и карьерные пути. Приложение будет доступно для широкой аудитории, включая студентов, соискателей и специалистов в области информационных технологий.

Назначение:

Создание сайта для предоставления актуальной информации о профессиях

на АПЗ, что поможет пользователям лучше понять специфику

работы, требования и перспективы в данной области.

Цели:

- Обеспечить доступ к информации о профессии для студентов и

специалистов.

- Повысить осведомленность о карьерных возможностях в области

программирования.

Характеристика объектов автоматизации:

- Информационные разделы о профессии (обязанности, требования,

рынок труда и карьерные пути).

- Интерактивные элементы (кнопки и динамическое отображение

информации).

- Пользовательский интерфейс, обеспечивающий удобное

взаимодействие с приложением.

Требования к системе:

- Возможность отображения информации по выбранным разделам.

- Интерактивные кнопки для навигации.

- Высокая производительность и быстродействие.

- Удобство и интуитивность интерфейса.

Состав и содержание работ по созданию системы:

- Разработка структуры и дизайна приложения.

- Создание XAML-разметки и написание cs-кода.

- Наполнение приложения контентом

- Тестирование функциональности и исправление ошибок.

Порядок контроля и приемки системы. Контроль за выполнением работ будет осуществляться на каждом этапе разработки:

- Проверка соответствия дизайна и функционала техническому заданию.

- Тестирование всех интерактивных элементов.

- Приемка готового приложения после завершения всех работ и исправления

ошибок.

Требования к составу и содержанию работ по подготовке объекта

автоматизации к вводу системы в действие:

- Проведение тестирования приложения на различных устройствах

- Подготовка инструкций по использованию приложения для конечных

пользователей.

Требования к документированию. Документация по проекту должна

включать:

- Описание архитектуры приложения.

- Инструкции по установке и настройке.

- Руководство пользователя.

- Техническая документация по коду (комментарии, описание функций).

Источники разработки:

Данное техническое задание описывает все ключевые аспекты разработки

сайта о профессиях в области программирования. Выполнение этого задания позволит

создать качественный и информативный ресурс, который будет полезен для

пользователей, интересующихся данной областью.

2.РАЗРАБОТКА И ИНТЕГРАЦИЯ МОДУЛЯ «ПРОФЕСИИ В АПЗ»

Модуль «Профессии в АПЗ» представляет собой интерактивное WPF-приложение, разработанное на языке C#, которое предоставляет пользователям информацию о профессиях в области программирования. Это приложение включает в себя различные аспекты профессии, такие как обязанности, навыки, обзор рынка труда и карьерный путь. Основная цель данного модуля — помочь пользователям лучше понять, что включает в себя работа инженера-программиста, и какие навыки им потребуются для успешной карьеры.

Идея создания модуля возникла из необходимости предоставить исчерпывающую информацию о профессии инженера-программиста тем, кто только начинает свой путь в IT-сфере или рассматривает возможность смены карьеры. Основной задачей было разработать простой, но информативный ресурс, который поможет пользователям разобраться в ключевых аспектаска профессии, таких как обязанности, требования к специалистам, состояние рынка труда и карьерный путь.

Структура приложения представляет собой:

Главная форма (MainWindow.xaml). Форма содержит заголовок, боковую панель с кнопками и область контента, где отображаются данные по выбранной теме. Боковая панель имеет кнопки, которые вызывают метод ShowInfo(), передавая соответствующий параметр для отображения нужной информации.

Класс MainWindowViewModel.cs. Класс содержит методы, управляющие отображением информации в зависимости от выбранной кнопки. Внутри метода происходит переключение между различными темами, связанными с профессией инженера-программиста.

Файл ресурсов (App.xaml). Файл стилей, который отвечает за визуальное оформление приложения, включая шрифты, цвета, отступы и другие элементы дизайна.

Первым этапом было определение структуры будущего приложения. Было решено использовать простую и интуитивно понятную навигацию, состоящую из четырех основных разделов: описание профессии, должностные обязанности, обзор рынка труда и путь становления инженера-программиста. Каждый раздел должен был содержать краткую, но емкую информацию, которая поможет читателю понять суть вопроса.

Далее приступили к созданию WPF-кода. Задача состояла в разработке каркаса XAML-файлов, которые служат основой для последующего наполнения приложения. Базовый шаблон был создан, включающий заголовок, боковую панель с кнопками и область для отображения основного контента. Главная форма (MainWindow.xaml) была разработана с использованием XAML и класса MainWindowViewModel.

Затем был написан код на C# для управления поведением кнопок и отображением информации. Методы класса MainWindowViewModel управляют загрузкой и показом информации в зависимости от выбранного раздела. Классы модели и представления были созданы для отделения логики от внешнего вида.

Наконец, были добавлены ресурсы (изображения и тексты) и проведены тесты, чтобы убедиться в корректном функционировании приложения. Это включало проверку правильной работы кнопок, отображения информации и возвращения на начальный экран.

Этот подход позволил создать полноценное WPF-приложение на языке C#, которое предоставляет пользователям глубокое понимание профессии в области программирования.

Давайте посмотрим на код нашего WPF-приложения на языке C#. XAML код MainWindow можно увидеть на (листинг 1)

Листинг 1 – XAML код MainWindow

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

<Label FontSize="25" Content="Программисты в АПЗ" Grid.ColumnSpan="2" Margin="271,10,129,207" Grid.RowSpan="2"/>

<Image Source="/APZ.png" Margin="200,160,200,160" Grid.ColumnSpan="2" Grid.RowSpan="2" />

<Button Content="Инженер-программист" Width="200" Height="40" Click="Button\_Click"/>

<Button Content="Инженер-технолог" Grid.Column="1" Width="200" Height="40" Click="Button\_Click\_1" />

<Button Content="Инженер-электроник" Grid.Row="1" Grid.Column="0" Width="200" Height="40" Click="Button\_Click\_2" />

<Button Content="Инженер-конструктор" Grid.Row="1" Grid.Column="1" Width="200" Height="40" Click="Button\_Click\_3" />

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

</Grid>

</Window>

Данный код представляет собой XAML разметку главного окна приложения.

Увидеть C# код основного окна можно на (листинг 2)

Листинг 2 – C# код MainWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp1

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

Window1 window1 = new Window1();

this.Close();

window1.Show();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

Window2 window2 = new Window2();

this.Close();

window2.Show();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

Window3 window3 = new Window3();

this.Close();

window3.Show();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

Window4 window4 = new Window4();

this.Close();

window4.Show();

}

}

}

Данный код отвечает за кнопки которые закрывают главное окно и открывают окно с выбранной профессией.

Давайте посмотрим на созданное главное окно приложения, а увидеть его можно на рисунке 1.



Рисунок 1 – Главное окно приложения

Здесь можно увидеть главное приложение с 4 кнопками и значком АПЗ.

Теперь давайте посмотрим на окно с профессией Инженер-программист на рисунке 2.

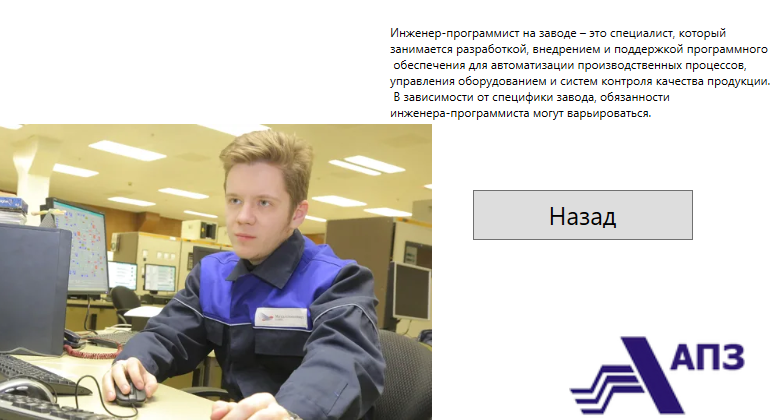


Рисунок 2 – Окно о профессии Инженер-программист

Здесь рассказывается о профессии «Инженер-программист». Давайте посмотрим на XAML разметку и C# код этого окна.

XAML разметку можно увидеть на (листинг 3)

Листинг 3 – XAML разметка Window1

<Window x:Class="WpfApp1.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Window1" Height="450" Width="800">

<Grid>

<Label Content="Инженер-программист на заводе – это специалист, который &#xA;занимается разработкой, внедрением и поддержкой программного&#xA; обеспечения для автоматизации производственных процессов, &#xA;управления оборудованием и систем контроля качества продукции.&#xA; В зависимости от специфики завода, обязанности &#xA;инженера-программиста могут варьироваться." Margin="0,10,0,-10" Grid.Column="1"/>

<Image Source="/i1`23.PNG" Margin="-29,115,327,0" Grid.ColumnSpan="2" />

<Image Source="/APZ.png" Margin="184,284,29,-35" Grid.Column="1" />

<Button Grid.Column="1" FontSize="25" Content="Назад" Height="50" Width="220" Click="Button\_Click" />

<Grid.ColumnDefinitions>

<ColumnDefinition Width="397\*"/>

<ColumnDefinition Width="403\*"/>

</Grid.ColumnDefinitions>

</Grid>

</Window>

Это была XAML разметка, с помощью которой был создан дизайн с профессией «Инженер-программист». C# код на этой странице нужен только для кнопки «Назад». Увидеть код можно увидеть на (листинг 4)

Листинг 4 – C# код Window1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfApp1

{

/// <summary>

/// Логика взаимодействия для Window1.xaml

/// </summary>

public partial class Window1 : Window

{

public Window1()

{

InitializeComponent();

}

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

{

MainWindow main = new MainWindow();

this.Close();

main.Show();

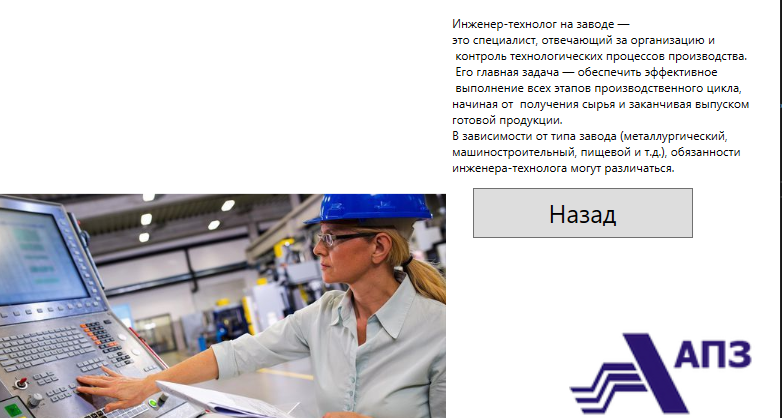
}

}

}

Это был C# код с помощью которого была реализована кнопка «Назад». В каждом окне с профессией C# код ничем не отличается, кроме тайтла, поэтому больше рассматривать его не будем, а продолжим рассматривать наши окна с профессиями и их XAML разметками.

Следующее окно с профессией «Инженер-технолог». Давайте посмотрим как он выглядит, а сделать это можно на рисунке 3.



Здесь можно увидеть