Střední průmyslová škola Třebíč
Manželů Curieových 734, 674 01 Třebíč

Maturitní práce

EDITOR JAZYKA HTML 5

Profilová část maturitní zkoušky

Studijní obor: počítačové systémy

Třída: PSA4

Školní rok: 2018/2019

Tomáš Milostný

ABSTRAKT

Cílem projektu je vytvoření programu HTML5 Editor s použitím Windows Forms a programovacího jazyka C#. Editor umožňuje snadné vytváření a práci s HTML soubory pro následné použití na webu. Je možné pracovat s jednotlivým souborem nebo otevřít složku a pracovat s více soubory současně, pohodlně pomocí záložek. Práce je také usnadněna vyznačením HTML syntaxe, kontextovou nápovědou, automatickým doplněním koncové značky a pravidelnou kontrolou validity kódu. Nakonec je možné ověřit kód zobrazením v náhledu.

KLÍČOVÁ SLOVA

HTML 5, editor, Windows Forms, C#, vývoj pro web

ABSTRACT

The goal of the project is to create an HTML5 Editor using Windows Forms and the C # programming language. The editor makes it easy to create and work with HTML files for subsequent use on the web. It is possible to work with both a single file or open a folder and work with multiple files simultaneously, conveniently using tabs. The work is also facilitated by marking the HTML syntax, context help, automatic supplementing of the end tag, and regular checking of the validity of the code. Finally, it is possible to verify the code by displaying in the preview.

KEYWORDS

HTML 5, editor, Windows Forms, C#, web development

PODĚKOVÁNÍ

Děkuji Mgr. Andree Odehnalové za předané znalosti a vedení projektu a Mgr. Petře Pacalové za kontrolu abstraktu v anglickém jazyce.

V Třebíči dne 28. března 2019 .....

podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval/a samostatně a uvedl/a v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil/a.

V Třebíči dne 28. března 2019 .....

podpis autora

OBSAH

ÚVOD

Zadání tématu mého ročníkového projektu, vytvoření HTML5 Editoru, jsem si vybral, protože mám zkušenosti s různými vývojovými prostředími pro web a přišlo mi velice zajímavé vytvořit si vlastní editor. Aplikace je vytvořena pomocí platformy Windows Forms, v programovacím jazyce C#, .NET Frameworku verze 4.7.2, ve vývojovém prostředí Visual Studio 2017/2019, se kterými jsem již byl seznámen v hodinách programového vybavení. V průběhu vývoje jsem se však několikrát obrátil k internetovým zdrojům pro rozšíření znalostí, které byly následně aplikovány při tvorbě algoritmů použitých v programu. HTML5 Editor splňuje všechny body zadání a umožňuje práci s HTML soubory.

1 TEORETICKÁ ČÁST

1.1 Jazyk a vývojové prostředí

Projekt byl vytvořen pomocí vývojového prostředí Visual Studio 2017 a běží na platformě Windows Forms v programovacím jazyce C#. Během vývoje došlo k aktualizaci na Microsoft Visual Studio 2019. Pro vytvoření prostředí vhodného pro práci s HTML soubory byla nutná znalost syntaxe jazyka HTML 5.

1.1.1 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s aplikacemi Windows Forms, webovými stránkami, webovými aplikacemi a webovými službami jak ve strojovém kódu, tak v řízeném kódu na platformách Microsoft Windows a dalších.

Visual Studio obsahuje editor kódu podporující IntelliSense a refaktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu aplikací s GUI, designer webu, tříd a databázových schémat. Je možné přidávat rozšíření, což vylepšuje funkčnost na téměř každé úrovni – od doplnění podpory pro verzovací systémy (jako Subversion a Microsoft Team Foundation Server) po nové nástroje jako editory a vizuální designery pro doménově specifické jazyky nebo nástroje pro další aspekty návrhu programu.

Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++ (použitím Visual C++), VB.NET (použitím Visual Basic .NET) a C# (použitím Visual C#). Také je podporováno XML/XSLT, HTML/XHTML, JavaScript a CSS. [1]

1.1.2 Windows Forms

Windows Forms (WinForms) je v informatice knihovna tříd pro tvorbu grafického rozhraní, která je součástí .NET frameworku firmy Microsoft. Poskytuje platformu pro vytváření aplikací pro desktop a tablet PC. Přestože je považován za nástupce staršího a komplexnějšího MFC založeného na C++, neposkytuje Windows Forms porovnatelná paradigmata a slouží pouze jako platforma pro tvorbu uživatelského rozhraní při řešení vícevrstvé architektury. [2]

1.1.3 C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270). Microsoft založil C# na jazycích C++ a Java (a je tedy nepřímým potomkem jazyka C, ze kterého čerpá syntaxi).

C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. [3]

V projektu je použit C# verze 7.3 a .NET Framework verze 4.7.2.

1.2 HTML

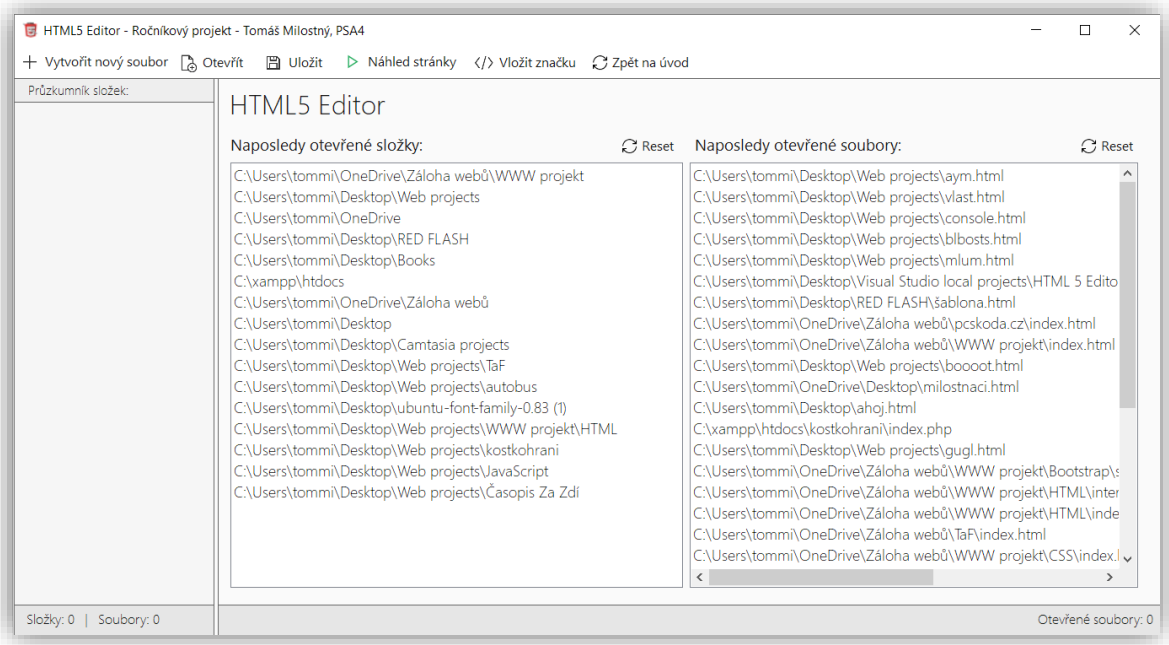
Hypertext Markup Language (zkratka HTML) je v informatice název značkovacího jazyka používaného pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy. HTML je hlavním z jazyků pro vytváření stránek v systému World Wide Web (www), který umožňuje publikaci dokumentů na internetu.

Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. [4]

2 PRAKTICKÁ ČÁST

2.1 Grafické uživatelské rozhraní (GUI)

Uživatelské rozhraní je rozděleno na 4 základní části – horní panel s funkčními tlačítky pro ovládání aplikace, „Průzkumník složek“ na levé straně, textový editor na pravé straně (v případě, že není otevřen žádný soubor, se zobrazí úvodní obrazovka s naposledy otevřenými soubory a složkami) a spodní lišta s informacemi pro uživatele, zobrazuje se počet načtených souborů a složek v „Průzkumníkovi složek“, orientace v souboru (zobrazení aktuálně vybraného řádku a celkový počet řádků), stav kontroly validity aktivního souboru a počet právě otevřených souborů.



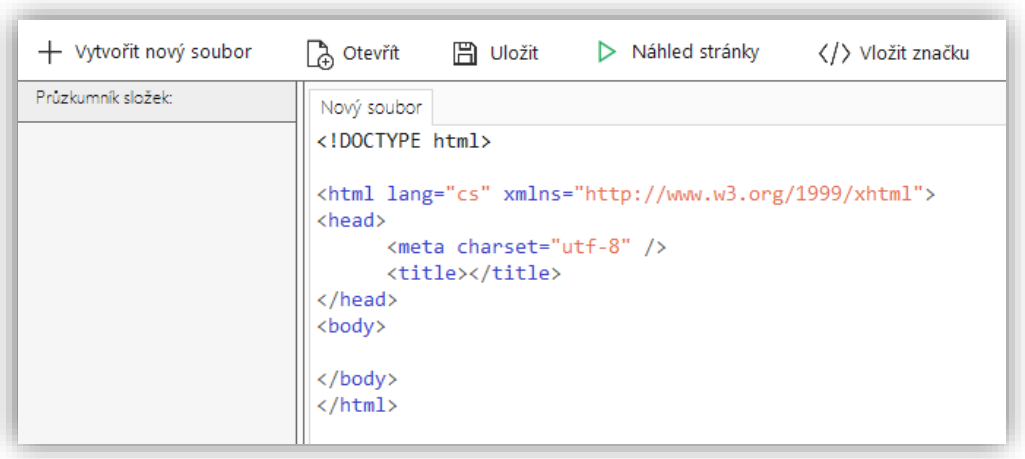
Obrázek 1 Základní uživatelské rozhraní

2.2 Práce se soubory

HTML5 Editor umožňuje vytváření, otevírání i ukládání souborů.

2.2.1 Vytvoření nového souboru

Pro vytvoření nového souboru je možné kliknout na tlačítko „Vytvořit nový soubor“ nebo pomocí klávesové zkratky Ctrl + N. Vytvoří se nová záložka a nový RichTextBox (více v kapitole „Textový editor kódu“), do kterého se načte připravená šablona HTML dokumentu, do které již uživatel může psát vlastní kód.



Obrázek 2 Vytvoření nového souboru

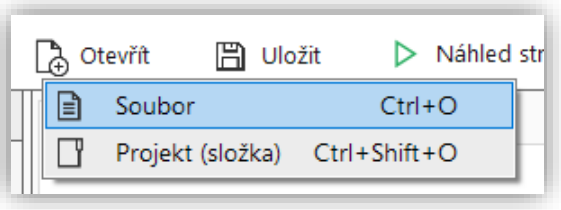
Dále v metodě `NewFile()` probíhá aktualizace počítadla řádků, reset časovače pro validaci řádku (více v kapitole „Validace kódu“), skryje se úvodní obrazovka, pokud předtím nebyl načten jiný soubor a také se zobrazí flowLayoutPanel, který obsahuje labely pro orientaci v textovém editoru a stav validity souboru. Nový soubor se rovněž přidává do Listu otevřených souborů, aby pozice souborů v záložkách odpovídaly pozicím v Listu.

```
private void NewFile()
{
    validationTimer.Stop();
    tabControl1.TabPages.Add("Nový soubor");
    openList.Add("Nový soubor");
    RichTextBox rtb = NewRichTextBox("template.html", "Nový soubor");
    tabControl1.TabPages[tabControl1.TabCount - 1].Controls.Add(rtb);
    tabControl1.SelectTab(tabControl1.TabCount - 1);
    rtb.SelectionStart = 137;
    LineCounter();
    if (panelStart.Visible) panelStart.Visible = false;
    if (!flowLayoutPanel1.Visible) flowLayoutPanel1.Visible = true;
    rtb.Focus();
    validationTimer.Start();
}
```

Obrázek 3 Metoda pro vytvoření nového souboru

2.2.2 Otevření souboru

Otevřít již existující soubor je možné rozkliknutím menu pod tlačítkem „Otevřít“ a výběrem „Soubor“ nebo pomocí klávesové zkratky Ctrl + O.



Obrázek 4 Menu otevření souboru

Poté se zobrazí dialogové okno pro výběr souboru, pokud vybraný soubor již není otevřen se následně načte do nového RichTextBoxu na nové záložce, provede se zvýraznění HTML syntaxe a provede se kontrola validity daného souboru. Cesta k souboru se nyní zapíše do Listu otevřených souborů (není možné otevřít jeden soubor ve více záložkách) a také se zapíše do historie otevřených souborů, která se zobrazuje na úvodní stránce.

```
private void OpenFile()
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        panelStart.Visible = false;
        LoadFile(openFileDialog1.FileName, "");
        tabControl1.SelectTab(tabControl1.TabCount - 1);
    }
}
```

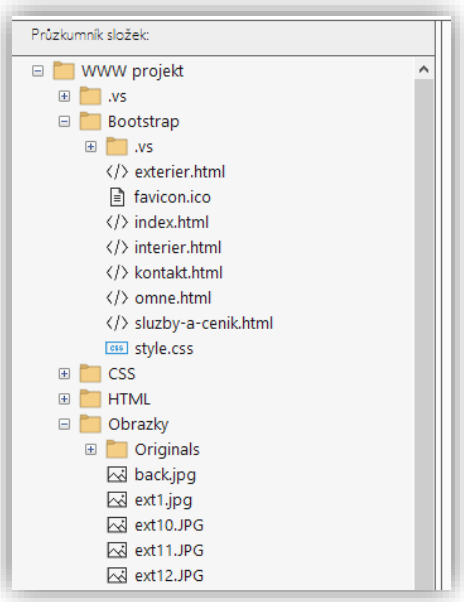
Obrázek 5 Metoda pro otevření souboru

```
private void LoadFile(string cesta, string parent) //načtení souboru a vytvoření RichTextBoxu
{
    bool is_open = false;
    foreach (string item in openList)
    {
        if (item == cesta) { is_open = true; break; }
    }
    if (!is_open)
    {
        Cursor.Current = Cursors.WaitCursor;
        validationTimer.Stop();
        if (parent == "") tabControl1.TabPages.Add(NodeName(cesta));
        else tabControl1.TabPages.Add(parent + "/" + NodeName(cesta));
        RichTextBox richTextBox = NewRichTextBox(cesta, parent + "/" + NodeName(cesta));
        tabControl1.TabPages[tabControl1.TabCount - 1].Controls.Add(richTextBox);
        openList.Add(cesta);
        LineCounter();
        UpdateLatestFiles(cesta);
        ValidateRichTextBox();
        if (!flowLayoutPanel1.Visible) flowLayoutPanel1.Visible = true;
        htmlRTBs[tabControl1.SelectedIndex].SelectionStart = 0;
        htmlRTBs[tabControl1.SelectedIndex].ScrollToCaret();
        tabControl1.SelectTab(tabControl1.TabCount - 1);
        richTextBox.Focus();
        validationTimer.Start();
        Cursor.Current = Cursors.Default;
    }
}
```

Obrázek 6 Metoda pro načtení souboru do editoru kódu

2.2.3 Otevření složky projektu

Otevřít složku pro následné načtení do stromové struktury (byla použita Windows Forms komponenta TreeView) v „Průzkumníku složek“ je možné přes tlačítko „Otevřít“ a výběrem položky „Projekt (složka)“ nebo pomocí klávesové zkratky Ctrl + Shift + O. Otevře se dialogové okno pro výběr složky a následně se spustí rekurzivní metoda pro načítání složek (v případě, že daná složka obsahuje nějaké podsložky, volá sebe sama pro jednotlivé podsložky a podle toho se skládají ve stromové struktuře) a metoda pro načítání souborů, které jsou navzájem provázané.



Obrázek 7 Načtená složka s projektem ve stromové struktuře

```
private void GetTreeNodeFolders(TreeNode treeNode, string item) //Načíst složky do stromové struktury
{
    var dir = Directory.GetDirectories(item);
    int i = 0;
    foreach (var subdir in dir)
    {
        try
        {
            TreeNode tree = new TreeNode(NodeName(subdir));
            treeNode.Nodes.Add(tree);
            foldercount++;
            var dir2 = Directory.GetDirectories(subdir);
            if (dir2.Count() > 0) GetTreeNodeFolders(treeNode.Nodes[i], subdir);
            GetTreeNodeFiles(treeNode.Nodes[i], subdir);
            i++;
        }
        catch (UnauthorizedAccessException)
        {
            MessageBox.Show("Nepovolený přístup do složky \"" + subdir + "\"", "Chyba při načítační", M
        }
    }
}
```

Obrázek 8 Metoda pro načítání složek do stromové struktury

V metodě pro otevírání souborů rovněž probíhá rozlišení typu souboru (HTML, CSS, JavaScript, obrázky a ostatní) a výběru příslušné ikony pro položku stromové struktury z předdefinovaného ImageListu.

```
private void GetTreeNodeFiles(TreeNode treeNode, string cesta) //Načíst soubory do stromové struktury
{
    var files = Directory.GetFiles(cesta, "*.*");
    foreach (string file in files)
    {
        fileList.Add(file);
        filecount++;
        string filename = NodeName(file).ToLower();
        TreeNode tree = new TreeNode(NodeName(file));
        if (filename.Contains(".html"))
        {
            tree.ImageIndex = 1;
            tree.SelectedImageIndex = 1;
        }
        else if (filename.Contains(".css")) { tree.ImageIndex = 2; tree.SelectedImageIndex = 2; }
        else if (filename.Contains(".js")) { tree.ImageIndex = 3; tree.SelectedImageIndex = 3; }
        else if (filename.Contains(".png") || filename.Contains(".jpg") || filename.Contains(".gif") || f
        { tree.ImageIndex = 4; tree.SelectedImageIndex = 4; }
        else { tree.ImageIndex = 5; tree.SelectedImageIndex = 5; }
        treeNode.Nodes.Add(tree);
    }
}
```

Obrázek 9 Metoda pro načítání souborů do stromové struktury

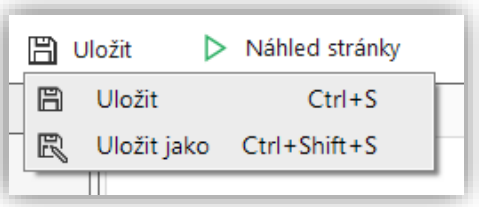
Dvojklikem na větev stromové struktury se soubor otevře. Pokud se jedná o soubor vhodný pro zobrazení v editoru kódu (HTML, CSS, PHP soubor), načte se v aplikaci do RichTextBoxu. V ostatních případech je pomocí metody `Process.Start()` soubor otevřen v jiné výchozí aplikaci systému Windows (např. obrázek se otevře v systémové aplikaci pro zobrazování obrázků).

```
private void OpenNode() //Otevření souboru ze stromové struktury
{
    if (treeView1.SelectedNode != null)
    {
        foreach (string item in fileList)
        {
            if (item.Contains(treeView1.SelectedNode.Parent.Text + "\\" + treeView1.SelectedNode.Text))
            {
                if (treeView1.SelectedNode.ImageIndex >= 1 && treeView1.SelectedNode.ImageIndex <= 3)
                {
                    if (treeView1.SelectedNode.Parent != treeView1.Nodes[0])
                    {
                        LoadFile(item, treeView1.SelectedNode.Parent.Text);
                    }
                    else LoadFile(item, "");
                }
                else if (treeView1.SelectedNode.ImageIndex == 4 || treeView1.SelectedNode.ImageIndex == 5)
                {
                    if (tabControl1.TabCount == 0) panelStart.Visible = true;
                    try { Process.Start(item); }
                    catch (System.ComponentModel.Win32Exception)
                    {
                        MessageBox.Show("Chyba při otevírání souboru " + item, "",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
                break;
            }
        }
    }
}
```

Obrázek 10 Metoda pro otevření souboru ze stromové struktury

2.2.4 Ukládání souboru

Pomocí menu pod tlačítkem „Uložit“ lze vybrat jednu ze dvou možností uložení – „Uložit“ a „Uložit jako“. Funkce je také možné aktivovat pomocí klávesových zkratek Ctrl + S a Ctrl + Shift + S. Zároveň je v programu ošetřeno, že když je ukládán nový soubor, vždy se použije metoda „Uložit jako“.



Obrázek 11 Menu ukládání souboru

Před uložením souboru je také provedena validita celého souboru k uložení, pokud je nevalidní, zobrazí se dialogové okno s upozorněním pro uživatele. Následně probíhá zápis souboru pomocí třídy StreamWriter.

```
private void Save()
{
    if (tabControl1.TabCount > 0)
    {
        ValidateRichTextBox();
        if (!labelShowErrors.Text.Contains("žádné")) //Validace při ukládání
        {
            if (MessageBox.Show("Soubor \"" + tabControl1.SelectedTab.Text + "\" není validní.\nPřesto uložit?", "",
                MessageBoxButtons.YesNo, MessageBoxIcon.Error) == DialogResult.No)
            {
                contextMenuSave.Items[0].Enabled = true;
                return;
            }
        }
        StreamWriter sw = new StreamWriter(openList[tabControl1.SelectedIndex], false);
        RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
        foreach (string line in rtb.Lines) sw.WriteLine(line);
        sw.Close();
        if (tabControl1.SelectedTab.Text.Contains('*'))
        {
            tabControl1.SelectedTab.Text = tabControl1.SelectedTab.Text.Remove(tabControl1.SelectedTab.Text.Length - 1);
        }
        contextMenuSave.Items[0].Enabled = false;
    }
}
```

Obrázek 12 Metoda pro uložení souboru

Metoda `SaveAs()` (Uložit jako) také kontroluje validitu napsaného kódu, následně zobrazí dialogové okno pro výběr umístění nového souboru a provede se zápis. Název záložky je pak změněn podle názvu nového souboru a aktualizuje se výpis otevřeného projektu v „Průzkumníku složek“.

```
private void SaveAs()
{
    if (tabControl1.TabCount > 0)
    {
        ValidateRichTextBox();
        if (!labelShowErrors.Text.Contains("žádné")) //Validace při ukládání
        {
            if (MessageBox.Show("Soubor \"" + tabControl1.SelectedTab.Text + "\" není validní.\nPřesto uložit?", "",
                MessageBoxButtons.YesNo, MessageBoxIcon.Error) == DialogResult.No)
                return;
        }
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            StreamWriter sw = new StreamWriter(saveFileDialog1.FileName, false);
            RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
            foreach (string line in rtb.Lines) sw.WriteLine(line);
            sw.Close();
            if (tabControl1.SelectedTab.Text.Contains('*'))
            {
                tabControl1.SelectedTab.Text = tabControl1.SelectedTab.Text.Remove(tabControl1.SelectedTab.Text.Length - 1);
            }
            tabControl1.SelectedTab.Text = NodeName(saveFileDialog1.FileName);
            openList.RemoveAt(tabControl1.SelectedIndex);
            openList.Insert(tabControl1.SelectedIndex, saveFileDialog1.FileName);
            UpdateLatestFiles(saveFileDialog1.FileName);
```

Obrázek 13 Metoda funkce Uložit jako...

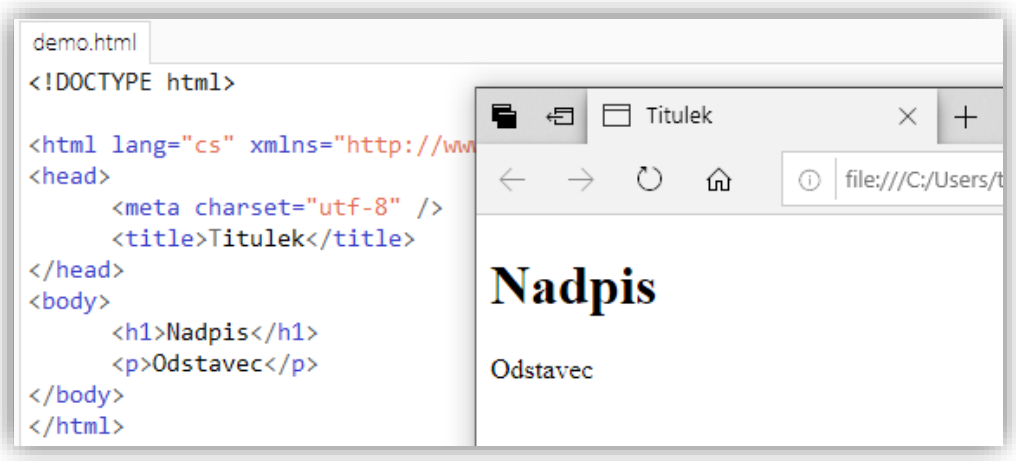
Během práce se souborem je k názvu souboru v záložce přidán symbol „*“, díky čemuž Editor i uživatel poznají, že soubor byl modifikován. Toto umožňuje snadné detekování neuložené práce v situaci, kdy chce uživatel zavřít soubor (pravé kliknutí na záložku a vybrat „Zavřít aktuální záložku“, kliknutím prostředního tlačítka myši nebo klávesové zkratky Ctrl + W) se provádí kontrola uložení (při zavření aplikace se kontrolují jednotlivé otevřené soubory).

```
private void CloseFileCheck(string tab) //Kontrola uložení souboru; zavření souboru
{
    validationTimer.Stop();
    if (tab.Contains('*'))
    {
        tab = tab.Remove(tab.Length - 1);
        if (MessageBox.Show("Soubor \"" + tab + "\" nebyl uložen.\nUložit nyní?", "Uložit...",
            MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
        {
            if (tab.Contains("Nový soubor")) SaveAs();
            else Save();
        }
    }
    openList.RemoveAt(tabControl1.SelectedIndex);
    htmlRTBs.RemoveAt(tabControl1.SelectedIndex);
    tabControl1.TabPages.Remove(tabControl1.SelectedTab);
    labelTabCount.Text = "Otevřené soubory: " + tabControl1.TabCount.ToString();
    LineCounter();
    if (tabControl1.TabCount == 0)
    {
        panelStart.Visible = true;
        flowLayoutPanel1.Visible = false;
        LoadStartUpListBoxes();
    }
    else validationTimer.Start();
}
```

Obrázek 14 Metoda pro kontrolu uložení při zavření souboru

2.2.5 Náhled souboru

Kliknutím na tlačítko „Náhled stránky“ nebo pomocí klávesy F5 se soubor v právě vybrané záložce otevře ve výchozím internetovém prohlížeči (zajišťuje správnost zobrazení z pohledu běžného uživatele a aktuální podporu HTML 5).



Obrázek 15 Zobrazení souboru v náhledu

V případě, že soubor nebyl uložen nebo se jedná o nový soubor, bude zobrazeno dialogové okno s upozorněním, že se nezobrazí neuložené úpravy v kódu souboru.

```
private void ShowPreview()
{
    if (tabControl1.TabCount > 0)
    {
        if (tabControl1.SelectedTab.Text.Contains('*') || tabControl1.SelectedTab.Text.Contains("Nový soubor"))
        {
            if (MessageBox.Show("Soubor \"" + tabControl1.SelectedTab.Text + "\" nebyl uložen a nezobrazí se poslední úpravy."
                + "\nUložit nyní?", "Uložit...", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
            {
                if (tabControl1.SelectedTab.Text.Contains("Nový soubor")) SaveAs();
                else Save();
            }
        }
        if (File.Exists(openList[tabControl1.SelectedIndex])) Process.Start(openList[tabControl1.SelectedIndex]);
    }
}
```

Obrázek 16 Metoda pro zobrazení náhledu souboru

2.3 Textový editor kódu

Textový editor HTML kódu je hlavní součástí celé aplikace. Je tvořen Windows Forms komponentou RichTextBox, který byl doplněn o další funkce vyžadované pro práci s HTML kódem.

2.3.1 Vytvoření nového RichTextBoxu

Při vytváření nebo načítání souboru je volána metoda `NewRichTextBox()`, kde se definují vlastnosti a události vytvářeného RichTextBoxu a provádí se zde také úkony společné oběma procesům. Načítá se soubor pomocí třídy StreamReader a provádí se zvýraznění HTML syntaxe. Nakonec se RichTextBox přidá do Listu htmlRTBs, který se následně používá při manipulaci s konkrétním RichTextBoxem podle indexu vybrané záložky.

```
private RichTextBox NewRichTextBox(string cesta, string tabName)
{
    RichTextBox rtb = new RichTextBox
    {
        Width = tabControl1.SelectedTab.Width,
        Height = tabControl1.SelectedTab.Height,
        ForeColor = Color.FromArgb(23, 23, 23),
        Font = new Font("Consolas", 11),
        BorderStyle = BorderStyle.None,
        BackColor = Color.White,
        Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right,
        WordWrap = false,
        AcceptsTab = true,
        ShortcutsEnabled = true
    };

    StreamReader sr = new StreamReader(cesta);
    while (!sr.EndOfStream) rtb.Text += sr.ReadLine() + "\n";
    sr.Close();

    if (cesta.Contains(".htm") || cesta.Contains(".php") || tabName == "Nový soubor")
        MatchAllHtml(rtb);
    rtb.Click += HtmlRichTextBox_Click;
    rtb.MouseDown += HtmlRichTextBox_MouseDown;
    rtb.MouseUp += HtmlRichTextBox_MouseUp;
    rtb.KeyUp += HtmlRichTextBox_KeyUp;
    rtb.KeyDown += HtmlRichTextBox_KeyDown;
    rtb.TextChanged += RichTextBox1_TextChanged;
    rtb.VScroll += HtmlRichTextBox_VScroll;

    htmlRTBs.Add(rtb);
    labelTabCount.Text = "Otevřené soubory: " + tabControl1.TabCount.ToString();
    return rtb;
}
```

Obrázek 17 Metoda pro vytvoření RichTextBoxu

2.3.2 Zvýraznění HTML syntaxe

Zvýraznění syntaxe jakéhokoliv jazyka velice usnadňuje práci a zvyšuje přehlednost napsaného kódu. V programu je odlišeno zvýraznění HTML značek, dat atributů mezi uvozovkami, čistého textu a poznámek. Při vytváření nebo otevírání souboru je volána metoda `MatchAllHtml()`, která vyhledá a zvýrazní všechny HTML značky v načteném souboru.

Obrázek 18 Ukázka zvýraznění HTML syntaxe

Při vytváření algoritmu pro zvýraznění HTML syntaxe byl použit internetový zdroj [5] pro inspiraci a získání nových poznatků o způsobu vyhledávání v textu pomocí regulárních výrazů a možnosti využití třídy Regex. Použitím MatchCollection, která obsahuje všechny nalezené shody podle regulárního výrazu, byly nalezeny veškeré HTML značky v dokumentu a následuje cyklické barevné zvýraznění jednotlivých shod (HTML tagů). Poté se v textu hledají HTML poznámky.

```
private void MatchAllHtml(RichTextBox rtb)
{
    int selectionAt = rtb.SelectionStart;
    int selectionRange = rtb.SelectionLength;
    LockWindowUpdate(rtb.Handle.ToInt32());

    //Označení HTML tagů
    MatchCollection m = Regex.Matches(rtb.Text, @"(?></?\w+)(?>(?:[^>'""]+|'[^']*'|""[^""]*"")*)>",
        RegexOptions.IgnoreCase);
    foreach (Match match in m) HighLightTags(match, 0, rtb);
    rtb.SelectionStart = selectionAt;
    rtb.SelectionLength = selectionRange;
    rtb.SelectionColor = Color.Black;

    try //Označení poznámky nakonec
    {
        string text = rtb.Text;
        bool poznamka_nalezena = false;
        for (int i = 0; i < text.Length; i++)
        {
            if (text[i] == '-' && text[i - 1] == '!' && text[i - 2] == '<')
            {
                poznamka_nalezena = true;
                rtb.SelectionStart = i - 3;
            }
            if (poznamka_nalezena && text[i] == '>' && text[i - 1] == '-' && text[i - 2] == '-')
            {
                rtb.SelectionLength = i - rtb.SelectionStart + 1;
                rtb.SelectionColor = Color.Green;
                poznamka_nalezena = false;
            }
        }
    }
    catch { }
    LockWindowUpdate(0);
}
```

Obrázek 19 Metoda pro zvýraznění HTML syntaxe celého souboru

Během psaní se pak volá metoda `MatchPartHtml()`, která hledá a zvýrazňuje HTML tagy pouze v poslední zapsané oblasti, aktivuje se v události TextChanged právě aktivního RichTextBoxu. Tato metoda funguje podobně jen je uzpůsobena rychlému zvýraznění pouze jedné HTML značky . Následuje rovněž obdobný způsob nalezení a vyznačení poznámky.

```
private void MatchPartHtml(RichTextBox rtb) //Označení HTMl tagu při psaní
{
    bool tag = false;
    int startSelectionAt = rtb.SelectionStart;
    int StartSelctionRange = rtb.SelectionLength;
    int searchSelectionStart = startSelectionAt;

    for (int i = startSelectionAt - 1; i >= 0; i--) //Nalezení začátku HTML tagu, start indexu
    {
        if (rtb.Text[i] == '<')
        {
            searchSelectionStart = i;
            if (rtb.Text[i + 1] != '!') tag = true;
            break;
        }
    }
    int searchSelectionRange = rtb.Text.Length - searchSelectionStart;
    for (int i = Math.Max(0, startSelectionAt - 1); i < rtb.Text.Length; i++) //Nalezení konce HTML tagu, délky tagu
    {
        if (rtb.Text[i] == '>')
        {
            searchSelectionRange = i - searchSelectionStart + 1;
            break;
        }
    }
    LockWindowUpdate(rtb.Handle.ToInt32());
    rtb.SelectionStart = searchSelectionStart;
    rtb.SelectionLength = searchSelectionRange;

    if (tag) //Označení HTML značky
    {
        Match match = Regex.Match(rtb.SelectedText,
            @"(?><//?\w+)(?>(?:[^>'""]+|'[^']*'|""[^""]*"")*)>",
            RegexOptions.IgnoreCase);
        HighLightTags(match, rtb.SelectionStart, rtb);
    }
    else //Označení poznámky
    {
        try
        {
            string text = rtb.Text;
            bool poznamka_nalezena = false;
            for (int i = 2; i < text.Length; i++)
            {
                if (text[i] == '-' && text[i - 1] == '!' && text[i - 2] == '<')
                {
                    poznamka_nalezena = true;
                    rtb.SelectionStart = i - 3;
                }
                if (poznamka_nalezena && text[i] == '>' && text[i - 1] == '-' && text[i - 2] == '-')
                {
                    rtb.SelectionLength = i - rtb.SelectionStart + 1;
                    rtb.SelectionColor = Color.Green;
                    poznamka_nalezena = false;
                }
            }
```

Obrázek 20 Metoda pro zvýraznění HTML syntaxe při psaní

2.3.3 Kontrola validity HTML kódu

Validní HTML kód musí splňovat několik pravidel, které se kontrolují při otevření souboru, během psaní i před uložením souboru. Kontroluje se, zda není značka zapsána v nesprávném formátu (je uzavřená mezi < a >, u atributů nechybí uvozovky, …), zda se zapsaná značka nachází v předdefinované databázi (není zastaralá a je podporovaná v HTML 5 nebo není nesmyslná) a zda soubor obsahuje <!DOCTYPE> a další kód je uzavřen mezi <html> a </html>.

Validace při psaní (metoda `ValidatePartRTB()`) probíhá v pravidelných 1sekundových intervalech pomocí Windows Forms komponenty Timer v Tick událostech nebo při odřádkování či použití šipek k navigaci v RichTextBoxu.

```
private void ValidateRichTextBox() //Validace celého RichTextBoxu
{
    if (tabControl1.TabCount > 0)
    {
        Cursor.Current = Cursors.WaitCursor;
        RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
        LockWindowUpdate(rtb.Handle.ToInt32());
        rtb.TextChanged -= RichTextBox1_TextChanged;
        int startStart = rtb.SelectionStart;
        rtb.SelectionStart = 0;
        pocet_chyb = 0;
        errorForm.ClearListBox();

        for (int i = 0; i < rtb.Lines.Count(); i++)
        {
            ValidateLine(i);
        }
        ValidateRequiredTags();
        UpdateValidationStatus(startStart);
        rtb.TextChanged += RichTextBox1_TextChanged;
        LockWindowUpdate(0);
        Cursor.Current = Cursors.Default;
    }
}

Počet odkazů: 2
private void ValidatePartRTB() //Validace jednoho řádku při psaní - validationTimer tick
{
    RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
    int firstcharindex = rtb.GetFirstCharIndexOfCurrentLine();
    int currentline = rtb.GetLineFromCharIndex(firstcharindex);
    int start = rtb.SelectionStart;
    ValidateLine(currentline);
    UpdateValidationStatus(start);
}
```

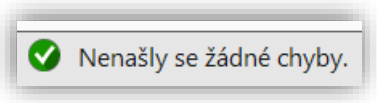
Obrázek 21 Metody pro validaci celého souboru a validaci při psaní

Metoda validace jednoho řádku, která je volána v ostatních metodách, postupně prochází jednotlivé znaky řádku a při nalezení značky zkontroluje jeho validitu. Pokud daná značka neodpovídá definovaným předpisům, zapíše se chybové hlášení do ListBoxu, který je v dalším formuláři „ErrorListForm“ a přičte se počet chyb.

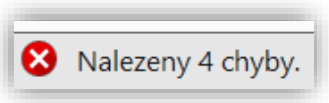
```
public int pocet_chyb;
private readonly ErrorListForm errorForm = new ErrorListForm();

Počet odkazů: 2
private void ValidateLine(int i) //Validace jednoho řádku
{
    RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
    string tag_ke_kontrole = "";
    bool zapisovat = false;
    errorForm.ClearLineLogs(i + 1, out int errorCount);
    pocet_chyb -= errorCount;

    for (int y = 0; y < rtb.Lines[i].Length; y++)
    {
        if (zapisovat) tag_ke_kontrole += rtb.Lines[i][y];
        if (rtb.Lines[i][y] == '>' || (zapisovat && y + 1 < rtb.Lines[i].Length && rtb.Lines[i][y + 1] == '<')
            || (tag_ke_kontrole != "" && y + 1 == rtb.Lines[i].Length))
        {
            Regex regex = new Regex(@"(?></?\w+)(?>(?:[^>'""]+|'[^']*'|""[^""]*"")*)>", RegexOptions.IgnoreCase);
            if (!regex.IsMatch(tag_ke_kontrole) && !tag_ke_kontrole.Contains("<!DOCTYPE") && tag_ke_kontrole.Contains("<")
                && !tag_ke_kontrole.Contains("<!--") && !tag_ke_kontrole.Contains("-->"))
            {
                errorForm.ErrorLog((i + 1).ToString() + ". řádek: chyba zápisu " + tag_ke_kontrole);
                pocet_chyb++;
            }
            else if (!tag_ke_kontrole.Contains("<!--") && !tag_ke_kontrole.Contains("-->"))
            {
                string nazev = "";
                for (int j = 1; j < tag_ke_kontrole.Length; j++)
                {
                    if (tag_ke_kontrole[j] == '/') continue;
                    if (tag_ke_kontrole[j] == ' ' || tag_ke_kontrole[j] == '>') break;
                    else nazev += tag_ke_kontrole[j];
                }
                bool valid = false;
                foreach (HTMLtag item in htmlTags)
                {
                    if (nazev == item.Nazev) { valid = true; break; }
                }
                if (!valid) errorForm.ErrorLog((i + 1).ToString() + ". řádek: nevalidní značka " + tag_ke_kontrole);
            }
            rtb.SelectionLength = 0;
            zapisovat = false;
        }
        if (rtb.Lines[i][y] == '<')
        {
            zapisovat = true;
            tag_ke_kontrole = "<";
        }
    }
```

Obrázek 22 Metoda pro validaci jednoho řádku RichTextBoxu

V případě, že byly nalezeny nějaké chyby, vypíše se počet chyb a zobrazí se chybová ikona ve spodním panelu s informacemi. Pokud je soubor v pořádku bude se zobrazovat text „Nenalezeny žádné chyby“ a zelená ikona označující, že je vše v pořádku.

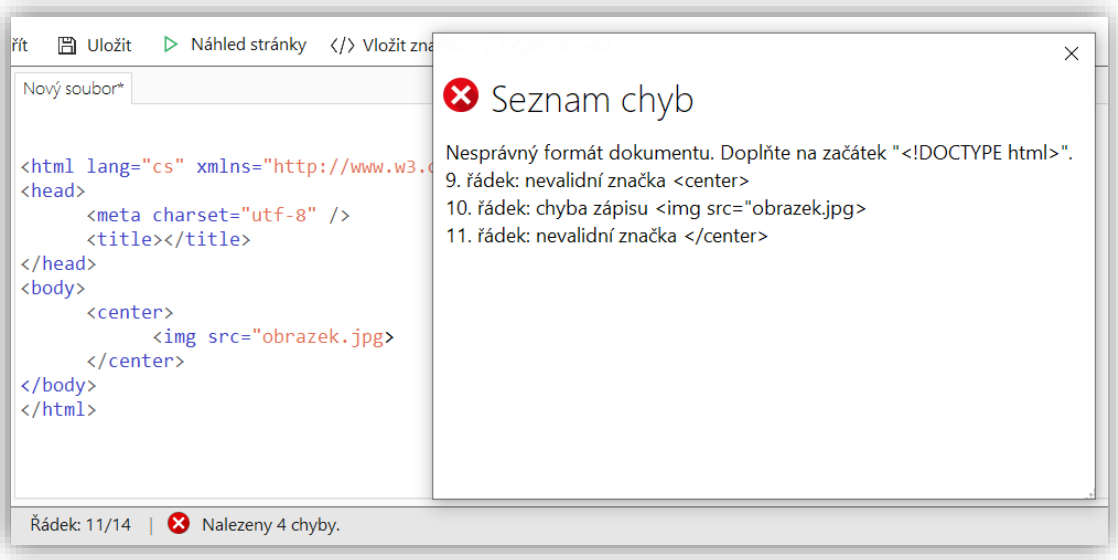


Obrázek 23 Ukázka výsledku validace bez chyb



Obrázek 24 Ukázka výsledku validace s chybami

Kliknutím na text „Nalezeno x chyb.“ se zobrazí dialogové okno s výpisem konkrétních chyb. Dvojitým kliknutím na chybu se kurzor RichTextBoxu přesune na řádek, ve kterém se chyba vyskytuje. V případě chyby s <!DOCTYPE> nebo značkou <html> se kurzor přesune na začátek dokumentu.



Obrázek 25 Ukázka výpisu chyb validace v chybovém dialogu

2.4 Práce s HTML značkami

Pro vkládání HTML značek, kontextovou nápovědu a kontrolu validity souboru byla vytvořena databáze HTML značek podle abecedního seznamu validních tagů pro HTML 5 na W3Schools [6]. Jedná se o neměnný seznam HTML značek, jejich atributů, zda jsou párové a jejich krátký popisek. Nejjednodušším řešením bylo vytvoření textového souboru, který má na řádcích jednotlivá data o HTML značkách, která jsou následně rozdělena středníkem.

Bylo zde využito objektově orientované programování a byla vytvořena třída „HTMLtag“, která má jednotlivá data rozděleny do vlastností. Při inicializaci hlavního formuláře se pomocí `List<HTMLtag> htmlTags = new HTMLtag().CreateDatabase();` zavolá metoda `CreateDatabase()`, která naplní nový List objekty třídy HTMLtag. Následně je již velmi rychlé a snadné tuto kolekci použít pro vyhledávání i výpis.

```
public List<HTMLtag> CreateDatabase()
{
    StreamReader sr = new StreamReader("html5data.txt");
    List<HTMLtag> htmlTags = new List<HTMLtag>();
    while (!sr.EndOfStream)
    {
        string[] data = sr.ReadLine().Split(';'); //rozdělení dat z jednoho řádku souboru

        bool parovost;
        if (data[2] == "ano") parovost = true;
        else parovost = false;

        string[] atributy_tagu = data[3].Split(','); //obsahuje seznam atributů pro jednolivé tagy
        string[] vsechny_atributy = atributy_tagu.Union(data[4].Split(',')).ToArray();
        Array.Sort(vsechny_atributy);

        htmlTags.Add(new HTMLtag(data[0], parovost, data[2], atributy_tagu, data[1], vsechny_atributy, data[5]));
    }
    sr.Close();
    return htmlTags;
}
```

Obrázek 26 Metoda pro načtení databáze HTML značek

Atributy jsou rozděleny na atributy specifické pro danou značku, obecné atributy společné většině značek (např. class, style, …) a atributy událostí (např. onclick, onmousedown, …). Obecné a atributy událostí jsou rozděleny do vlastních polí a načítají se také z vlastních souborů. V třídě HTMLtag se rozlišuje, zda daná značka podporuje pouze obecné atributy nebo atributy událostí nebo obojí.

```
class HTMLtag
{
    Počet odkazů: 9
    public string Nazev { get; }
    Počet odkazů: 2
    public bool Parovy { get; }
    Počet odkazů: 2
    public string ParovyToString { get; }
    Počet odkazů: 2
    public string[] Atributy { get; }
    Počet odkazů: 2
    public string Popisek { get; }
    Počet odkazů: 2
    public string[] VsechnyAtributy { get; }
    Počet odkazů: 5
    public string Global_Event { get; }

    Počet odkazů: 1
    public HTMLtag(string nazev, bool parovy, string parovy_slovy, string[] atributy, string popisek,
        string[] vsechny_atributy, string global_event)
    {
        Nazev = nazev;
        Parovy = parovy;
        Atributy = atributy;
        Popisek = popisek;
        ParovyToString = parovy_slovy;
        VsechnyAtributy = vsechny_atributy;
        Global_Event = global_event;
    }

    Počet odkazů: 2
    public HTMLtag() { }

    Počet odkazů: 2
    public List<HTMLtag> CreateDatabase()
```

Obrázek 27 Třída HTMLtag

2.4.1 Automatické vkládání koncové značky při psaní

Během zápisu do RichTextBoxu se vyvolává událost TextChanged. Pokud poslední napsaný znak byl konec HTML tagu „<“ spustí se automatické vkládání koncové značky. Prvním krokem je rozlišení, zda obsahuje nějaké atributy a zda se náhodou nejedná o koncovou značku (v tomto případě se doplnění nevykoná).

```
//Automatické vkládání koncových značek
bool koncovyTag = false;
bool tag_s_atributem = false;
if (rtb.Text[rtb.SelectionStart - 1] == '>' && !backspacePressed)
{
    listBoxIntellisense.Items.Clear();
    int start = rtb.SelectionStart;
    int start1 = start;
    int length = 0;
    for (int i = rtb.SelectionStart; rtb.Text[i] != '<'; i--)
    {
        if (rtb.Text[i] == ' ')
        {
            start1 = i;
            length = 0;
            tag_s_atributem = true;
        }
        if (rtb.Text[i] == '/' && rtb.Text[i - 1] == '<')
        {
            koncovyTag = true;
            break;
        }
        length++;
    }
    if (!koncovyTag)
```

Obrázek 28 Automatické vložení koncové značky 1/2

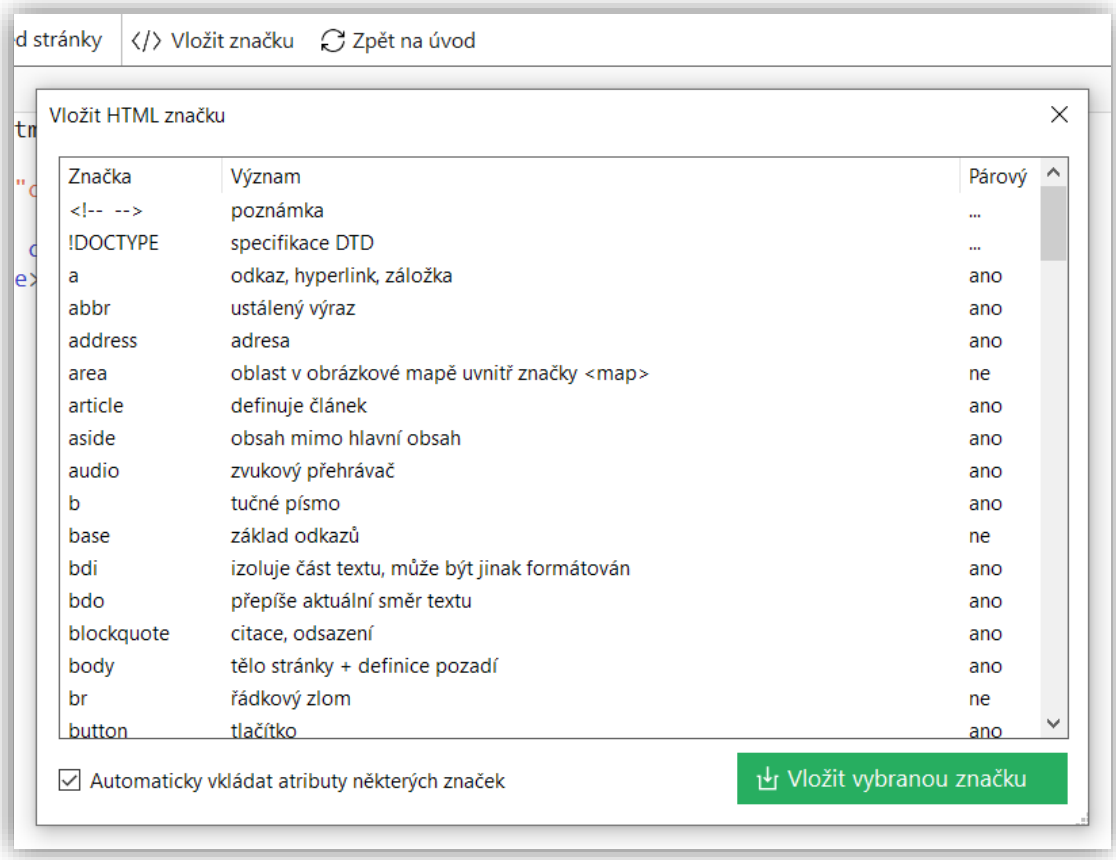
Pokud již předtím nebyla vložena koncová značka, probíhá nalezení názvu značky a vyhledání v databázi HTML značek. Jakmile je příslušná značka nalezena, zjistí se, zda se jedná o párovou značku. Do textu se vloží začátek koncové značky „</“ a nalezený název značky a proběhne zvýraznění HTML syntaxe.

```
if (!koncovyTag)
{
    rtb.SelectionStart = start1 - length;
    rtb.SelectionLength = length;

    string tag = rtb.SelectedText;
    if (tag_s_atributem) tag += ">";
    tag = tag.Remove(0, 1);

    foreach (HTMLtag item in htmlTags)
    {
        if (tag.Remove(tag.Length - 1, 1) == item.Nazev)
        {
            if (item.Parovy)
            {
                rtb.SelectionStart = start;
                rtb.SelectionLength = 0;
                rtb.SelectedText += "</" + tag;
                MatchPartHtml(rtb);
            }
            break;
        }
    }
    rtb.SelectionLength = 0;
    rtb.SelectionStart = start;
}
kontextovaNapoveda = false;
```

Obrázek 29 Automatické vložení koncové značky 2/2

2.4.2 Vkládání HTML značek z dialogového okna

Dialogové okno pro výběr značky pro vložení se zobrazí kliknutím na tlačítko „Vložit značku“ nebo pomocí klávesové zkratky Ctrl + Shift + V. HTML značky se vypisují do Windows Forms komponenty ListView a výpis je rozdělen do třech sloupců „Značka“, „Význam“ (krátký popisek, co daná značka dělá) a „Párový“ (ano nebo ne, podle toho, zda se jedná o párovou značku). Pod výpisem databáze je zaškrtávací políčko, kterým se povoluje možnost automatického vkládání některý atributů dané značky. Kliknutím na tlačítko „Vložit vybranou značku“ nebo stisknutím klávesy Enter potvrdíte výběr značky a vloží se na pozici kurzoru v aktivním RichTextBoxu.



Obrázek 30 Dialogové okno pro vložení HTML značky

Vybráním značky ze seznamu a potvrzením výběru nastaví se v instanci formuláře „insertTagForm“ vlastnost „SelectedTagIndex“, který odpovídá pořadí značky v databázi HTML značek. Poté jsou dostupné název značky i její atributy k vložení.

Během vkládání textu do RichTextBoxu se vykonává událost TextChanged, ve které je poté automaticky doplněna koncová značka (pokud je značka párová) a zvýrazní se syntaxe jako při běžném psaní.

```
private void InsertTag()
{
    if (tabControl1.TabCount > 0)
    {
        if (tabControl1.SelectedTab.Text.Contains(".htm") || tabControl1.SelectedTab.Text.Contains(".php")
            || tabControl1.SelectedTab.Text.Contains("Nový soubor"))
        {
            if (insertTagForm1.ShowDialog() == DialogResult.OK) //Otevření okna pro vybrání a vložení tagu
            {
                RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
                rtb.KeyUp -= HtmlRichTextBox_KeyUp;

                string html_tag = htmlTags[insertTagForm1.SelectedTagIndex].Nazev;
                string[] atributy = htmlTags[insertTagForm1.SelectedTagIndex].Atributy;

                int select = rtb.SelectionStart;
                if (html_tag != "<!--  -->")
                {
                    rtb.SelectedText += '<' + html_tag;
                    if (insertTagForm1.VlozitAtributy() && atributy[0] != "nn")
                    {
                        foreach (string item in atributy)
                        {
                            rtb.SelectedText += ' ';
                            if (item.EndsWith("-")) rtb.SelectedText += item.Remove(item.Length - 1);
                            else rtb.SelectedText += item + "=\"\"";
                        }
                    }
                    rtb.SelectedText += '>';
                }
                else //Vložení poznámky
                {
                    rtb.SelectedText += html_tag;
                    rtb.SelectionStart = select + 5;
                    rtb.SelectionLength = 0;
```

Obrázek 31 Metoda pro vložení HTML značky vybrané v dialogu

2.4.3 Kontextová nápověda

Kontextová nápověda je rychlá nápověda, která se filtruje podle kontextu zadaného textu. Aktivuje se zapsáním začátku nové značky „<“ a zobrazí se seznam dostupných značek do ListBoxu na pravé straně.

Obrázek 32 Ukázka kontextové nápovědy

Po aktivaci kontextové nápovědy probíhá sestavení hledaného výrazu procházením textu od aktuální pozice kurzoru. Poté probíhá rozlišení, zda se bude načítat seznam HTML značek nebo atributů a ty se následně filtrují podle hledaného výrazu.

```
//Kontextová nápověda (Intellisense)
if (rtb.Text[rtb.SelectionStart - 1] == '<' && texChanCount < 1)
{
    kontextovaNapoveda = true;
    start_nap = rtb.SelectionStart;
}

if (kontextovaNapoveda && texChanCount < 1)
{
    listBoxIntellisense.Items.Clear();
    int start = rtb.SelectionStart;
    int length = 0;
    for (int i = rtb.SelectionStart; rtb.Text[i] != '<'
        && rtb.SelectionStart >= start_nap; i--)
    {
        length++;
    }

    rtb.SelectionStart = start - length + 1;
    rtb.SelectionLength = 0;
    if (length > 0) rtb.SelectionLength = length - 1;
    hledano = rtb.SelectedText;
    rtb.SelectionStart = start;
    sem_vlozit = start;
    rtb.SelectionLength = 0;
```

Obrázek 33 Aktivace kontextové nápovědy a sestavení hledaného výrazu

```
napoveda_atribut = false;
foreach (HTMLtag item in htmlTags)
{
    if (item.Nazev.StartsWith(hledano) && !hledano.Contains(' '))
        listBoxIntellisense.Items.Add(item.Nazev);
    else
    {
        string[] split = hledano.Split(' ');
        if (item.Nazev == split[0])
        {
            hledano = split.Last();
            foreach (string atribut in item.VsechnyAtributy) //speciální atributy tagu
            {
                if (atribut.StartsWith(hledano) && atribut != "nn")
                {
                    napoveda_atribut = true;
                    listBoxIntellisense.Items.Add(atribut);
                }
            }
            if (item.Global_Event == "global" || item.Global_Event == "both") //obecné atributy nebo obojí
            {
                foreach (string atribut in global_atributy)
                {
                    if (atribut.StartsWith(hledano))
                    {
                        napoveda_atribut = true;
                        listBoxIntellisense.Items.Add(atribut);
                    }
                }
            }
            if (item.Global_Event == "event" || item.Global_Event == "both") //událostné atributy nebo obojí
            {
                foreach (string atribut in event_atributy)
                {
                    if (atribut.StartsWith(hledano))
```

Obrázek 34 Načítání a filtrování kontextové nápovědy

Pokud je kontextová nápověda aktivní, šipky nahoru a dolů slouží pro přesun do ListBoxu a výběr se potvrdí stisknutím klávesy tabulátoru a do aktivního RichTextBoxu se doplní zbytek hledaného výrazu. Stisknutím klávesy mezerníku lze pokračovat ke kontextové nápovědě atributů nebo lze kontextovou nápovědu ukončit zapsáním ukončení značky „>“.

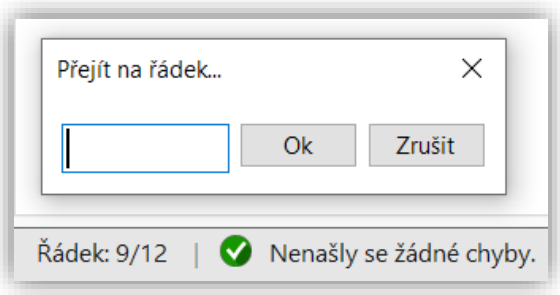
```
//Přístup pomocní šipek v RTB keydown
Počet odkazů: 1
private void ListBoxIntellisense_Leave(object sender, EventArgs e)
{
    if (tabControl1.TabCount > 0)
    {
        RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
        rtb.SelectionStart = sem_vlozit;
        bool doplnit_rovnitko = true;

        if (listBoxIntellisense.SelectedItem != null)
        {
            string selected = listBoxIntellisense.SelectedItem.ToString();
            if (selected.StartsWith("<"))
            {
                rtb.SelectedText += selected.Remove(0, 1);
                rtb.SelectionStart -= 4;
            }
            else if (selected.EndsWith("-"))
            {
                rtb.SelectedText += selected.Remove(0, hledano.Length).Remove(selected.Length - 1, 1);
                doplnit_rovnitko = false;
            }
            else rtb.SelectedText += selected.Remove(0, hledano.Length);
        }
        if (napoveda_atribut && doplnit_rovnitko)
        {
            rtb.SelectedText += "=\"\"";
            rtb.SelectionStart--;
        }
        rtb.Focus();
        listBoxIntellisense.Items.Clear();
    }
}
```

Obrázek 35 Doplnění HTML značky nebo atributu v kontextové nápovědě

2.5 Další funkce

Během vývoje došlo i k přidání dalších funkcí, které nebyly specifikovány v původním zadání projektu, ale dále vylepšují využití HTML5 Editoru.

2.5.1 Orientace v řádcích souboru



Obrázek 36 Orientace v řádcích souboru

Při práci s HTML kódem je pro uživatele užitečnou informací, když ví, na kterém řádku se právě nachází a také kolik řádků má soubor celkem. Informace se zobrazuje ve spodní části uživatelského rozhraní a je ve formátu: Řádek: „aktuální řádek“ / „celkový počet řádků“. Kliknutím na text nebo pomocí klávesové zkratky Ctrl + G se zobrazí dialogové okno pro přesun kurzoru na konkrétní řádek, toto je velmi užitečné zejména v rozsáhlých souborech. Přesun na řádek se rovněž používá v dialogu s výpisem chyb validity.

```
private void LineCounter()
{
    try
    {
        RichTextBox rtb = htmlRTBs[tabControl1.SelectedIndex];
        int firstcharindex = rtb.GetFirstCharIndexOfCurrentLine();
        int currentline = rtb.GetLineFromCharIndex(firstcharindex) + 1;
        labelLineCounter.Text = "Řádek: " + currentline.ToString() + "/";
        if (rtb.Lines.Count() > 0)
            labelLineCounter.Text += rtb.Lines.Count().ToString();
        else labelLineCounter.Text += (rtb.Lines.Count() + 1).ToString();
    }
    catch { labelLineCounter.Text = "Řádek: 0/0"; }
}
```

Obrázek 37 Metody pro počítadlo řádků a přesun na řádek

2.5.2 Úvodní obrazovka programu

Při spuštění aplikace, nebo, když nejsou otevřené žádné soubory v záložkách, se načtou naposledy otevřené složky a soubory do dvou ListBoxů (Obrázek 2). Cesty k složkám a souborům jsou uloženy (ukládání probíhá při otevření souboru nebo složky) ve dvou textových souborech. Tato funkce aplikace umožňuje snadné pokračování s dříve rozdělanou prací.

```
private void UpdateLatestFiles(string cesta)
{
    List<string> soubory = new List<string> { cesta };
    StreamReader sr = new StreamReader("lastfiles.txt");
    for (int i = 0; i < 31 && !sr.EndOfStream; i++)
    {
        string line = sr.ReadLine();
        bool zapsat = true;
        foreach (string item in soubory)
        {
            if (item == line) { zapsat = false; break; }
        }
        if (zapsat) soubory.Add(line);
    }
    sr.Close();

    StreamWriter sw = new StreamWriter("lastfiles.txt");
    foreach (string item in soubory)
    {
        sw.WriteLine(item);
    }
    sw.Close();
}
```

Obrázek 38 Metoda pro uložení historie posledních souborů

Vrátit se na úvodní obrazovku je možné kliknutím na tlačítko „Zpět na úvod“ a je nabídnuta možnost zavřít aktuální projekt (proběhne zavření jednotlivých záložek a kontrola uložení souborů).

ZÁVĚR

Díky práci na tomto projektu jsem určitě rozšířil své znalosti v programovacím jazyce C# a naučil jsem se čelit výzvám, se kterými jsem doposud neměl zkušenosti, ale všechny úkoly stanovené v zadání projektu byly splněny. V průběhu vývoje došlo k několika změnám a vylepšením (např. forma zobrazení náhledu HTML souboru nebo přidání dalších funkcí jako orientace v řádcích a úvodní obrazovka).

Výsledný software by byl jistě použitelný pro začínající webové vývojáře pracující s jazykem HTML 5. Pro vývoj webové stránky v reálném světě by však bylo zapotřebí rozšířit podporu pro další jazyky (např. CSS, PHP, JavaScript, …), ale toto nebylo cílem projektu.

SEZNAM LITERATURY

[1] Microsoft Visual Studio. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-03-20]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio

[2] Windows Forms. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-03-20]. Dostupné z: https://cs.wikipedia.org/wiki/Windows_Forms

[3] C Sharp. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-03-20]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

[4] Hypertext Markup Language. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-03-20]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language

[5] Zvýrazňovač HTML systaxe v C#. In: ITnetwork [online]. Praha: David Čápka, 2014 [cit. 2019-03-24]. Dostupné z: https://www.itnetwork.cz/csharp/formulare/winforms/program-zvyraznovac-syntaxe

[6] HTML Element Reference. In: *W3Schools* [online]. W3Schools, 2019 [cit. 2019-03-24]. Dostupné z: https://www.w3schools.com/tags/

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

A ZADÁNÍ MATURITNÍ PRÁCE

Střední průmyslová škola Třebíč
Manželů Curieových 734, 674 01 Třebíč

Téma č.: **5**

Název tématu: **Editor jazyka HTML5**

Konkrétní úlohy, které jsou v práci řešeny:

Vytvořit program HTML editor usnadňující práci při tvorbě webových stránek. Editor bude podporovat syntaxi jazyka HTML5. Tagy lze zapisovat v textovém režimu i pomocí zkratkových kláves nebo vkládat pomocí myši z vhodného úložiště tagů. Bude nabídnuta i možnost vložení atributů tagu. U párových značek bude nabídnuta i ukončovací značka. Editor bude obsahovat kontextovou nápovědu, která se vyvolá stisknutím zkratkové klávesy pro tag, na němž stojí kurzor. Zadaný kód bude možno zobrazit v náhledu (WYSIWYG).

Editor bude obsahovat funkce pro pohodlnou práci se soubory. Program bude vytvořen v programovacím jazyku C#.

vedoucí práce: Mgr. Andrea Odehnalová třída: PSA4 žák: Tomáš Milostný