

Licenciatura em Sistemas Multimédia

**Estruturas de Dados e Algoritmia II**

*CODr – Organizador de código*

*Daniel jose Guedes carneiro*

# Índice

[Índice 3](#_Toc45573773)

[1. Apresentacao do sistema 5](#_Toc45573774)

[2. Estrutura do Projeto 5](#_Toc45573775)

[2.1. Login 5](#_Toc45573776)

[2.2. Snippets 6](#_Toc45573777)

[2.3. New Snippets 7](#_Toc45573778)

[2.4. Editar 7](#_Toc45573779)

[2.5. Novo 8](#_Toc45573780)

[2.6. Adicionar Extensões 8](#_Toc45573781)

[2.7. Pesquisar no Código 9](#_Toc45573782)

[2.8. Extensoes 9](#_Toc45573783)

[2.9. Editar e Remover 9](#_Toc45573784)

[2.10. Sobre 10](#_Toc45573785)

[3. Conclusão 11](#_Toc45573786)

[4. Codificação C# 12](#_Toc45573787)

[4.1 ClassCryptography.cs 12](#_Toc45573788)

[4.2 ClassDatabase.cs 12](#_Toc45573789)

[4.3 ClassExtensions.cs 13](#_Toc45573790)

[4.4 ClassFile.cs 16](#_Toc45573791)

[4.5 ClassLogin.cs 17](#_Toc45573792)

[4.6 ClassSnippets.cs 18](#_Toc45573793)

[4.7 CodeEditorHotKeys.cs 21](#_Toc45573794)

[4.8 CodeEditorSearchManager.cs 22](#_Toc45573795)

[4.9 Login.cs 24](#_Toc45573796)

[4.10 Main.cs 26](#_Toc45573797)

[4.11 About.cs 30](#_Toc45573798)

[4.12 AllSnippets.cs 32](#_Toc45573799)

[4.13 Extensions.cs 33](#_Toc45573800)

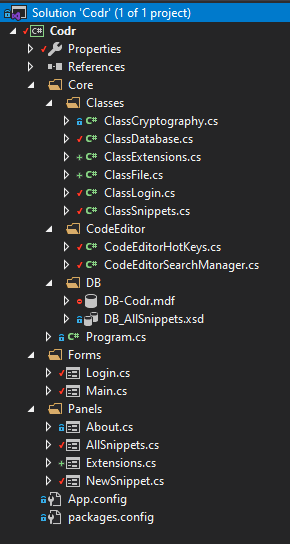
[4.14 NewSnippet.cs 34](#_Toc45573801)

[5. Codificação SQL 62](#_Toc45573802)

# 

# 1. Apresentacao do sistema

No âmbito da disciplina de Estruturas de Dados e Algoritmia I foram dadas algumas sugestões para a elaboração deste trabalho. O exemplo escolhido foi a elaboração de um gestor de codigo. Este software foi desenvolvido em C# no *Microsoft Visual Studio*.

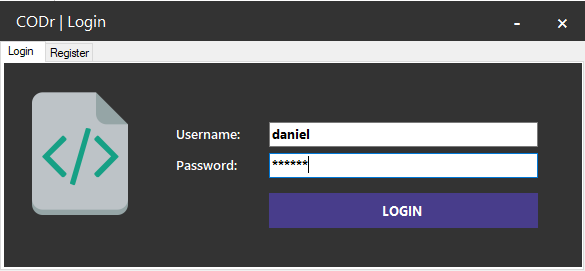


# 2. Estrutura do Projeto

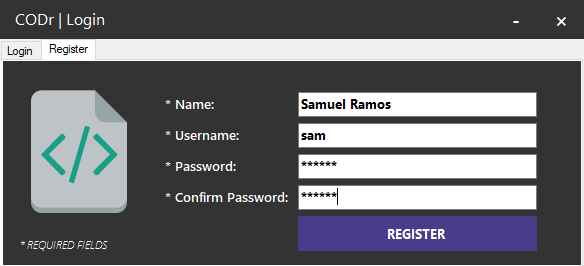
Entende-se por objetivo deste projeto a aplicação dos conhecimentos até a data adquiridos nas aulas, tendo isso em conta, a estrutura do projeto compreende-se pela criação de *snippets* (código fonte) e a criação de ficheiros.

## 2.1. Login

O utilizador pode aqui tem a possibilidade de se registar ou entrar com as suas credenciais. As *passwords* são encriptadas em código e enviadas para a base de dados.

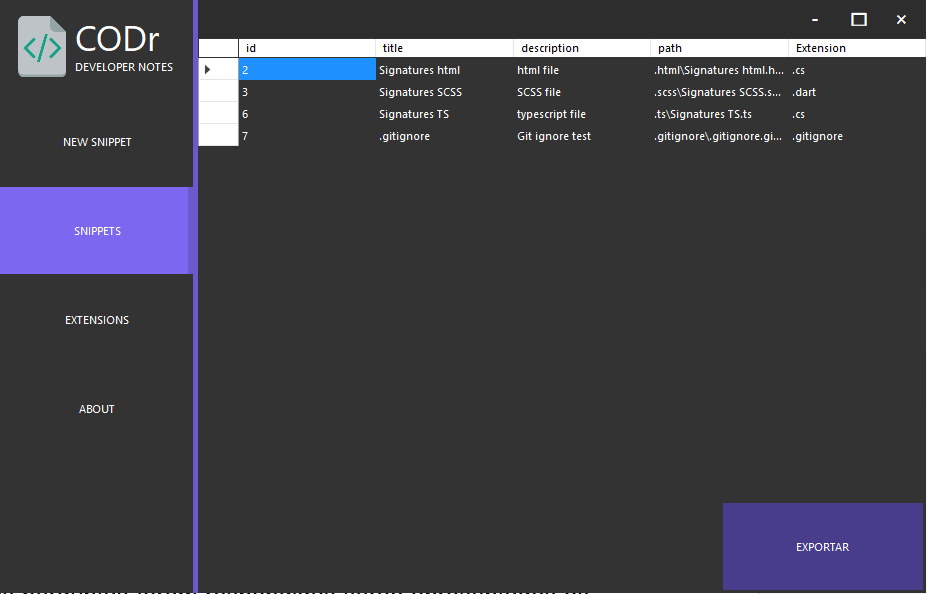


Para o utilizador se registar terá de aceder a aba *Register* e preencher os campos. Apos preencher os campos será direcionado para a aba de *Login* para entrar na sua conta.

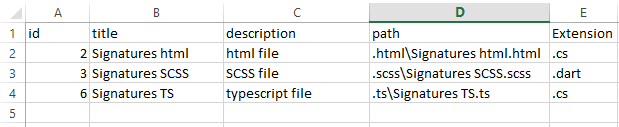


## 2.2. Snippets

Depois de efetuar o login a primeira pagina que vai ser redirecionado será a pagina de *snippets*. Aqui poderá ver todos os *snippets* guardados e também exportar para CSV a tabela apresentada.

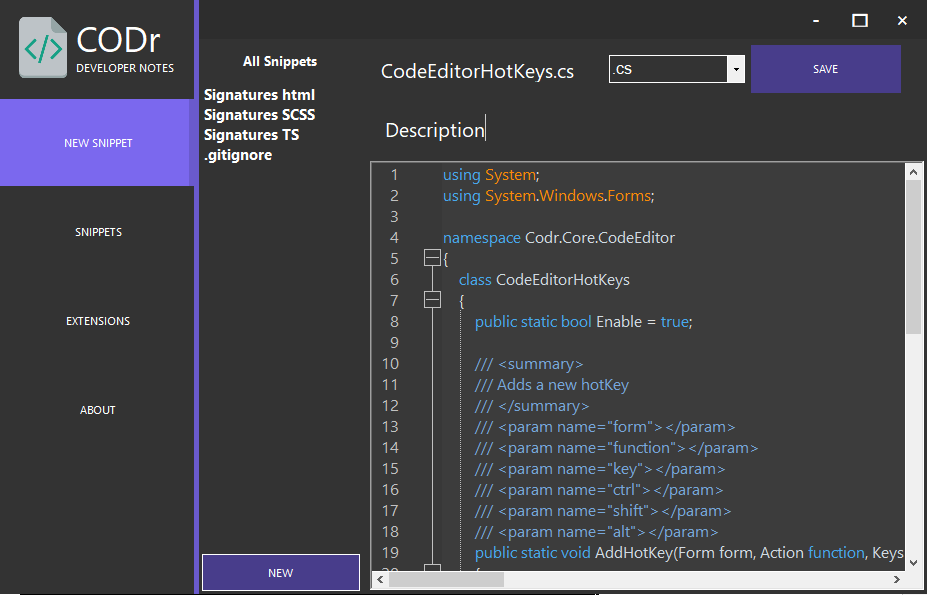


Neste exemplo em específico seria exportado o seguinte ficheiro CSV:



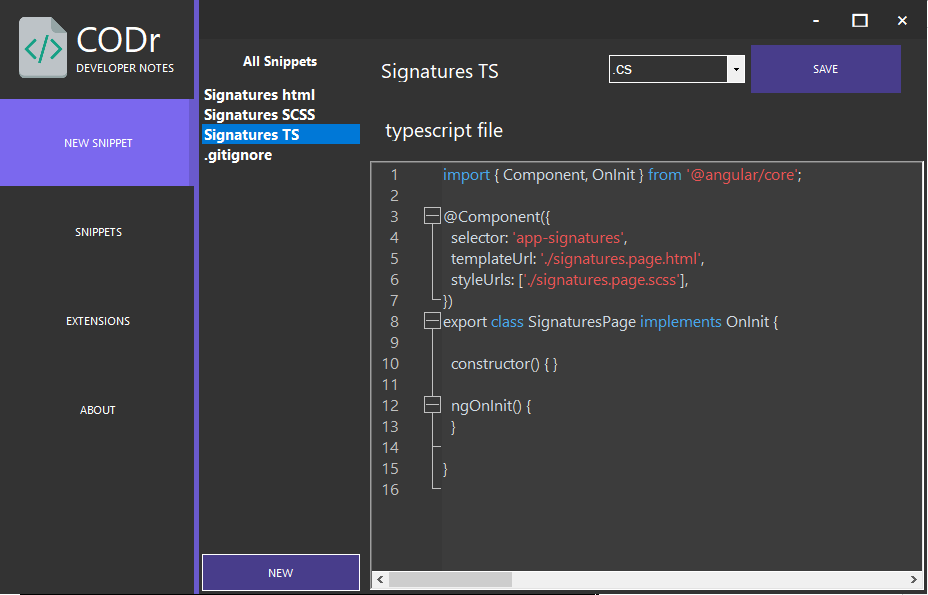
## 2.3. New Snippets

Nesta página será possível visualizar e editar todos os snippets.



## 2.4. Editar

Para editar um *snippet* basta selecionar através da lista do lado direito e os valores do mesmo serão demonstrados no lado direito. Ao selecionar uma extensão o programa vai guardar este codigo dentro de uma pasta com o mesmo nome.

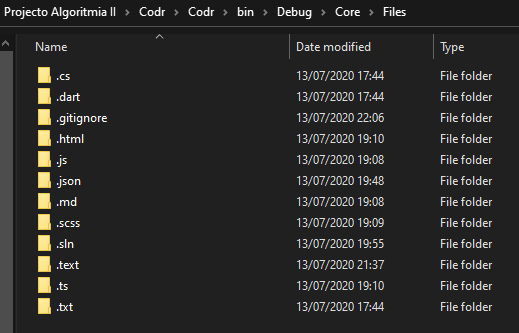


## 2.5. Novo

Para criar um novo *snippet* basta clicar no Butão *NEW e preencher todos os campos necessários para a criação do novo ficheiro.*

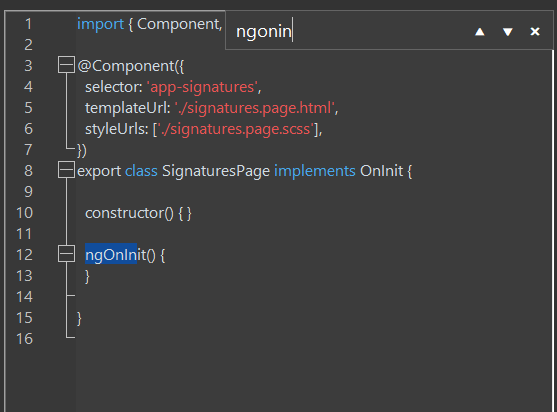
## 2.6. Adicionar Extensões

No caso de não haver extensões que o utilizador procure basta escrever uma extensão começada por “.” seguida pelo nome da extensão esta será adicionada automaticamente. Também será criada uma nova pasta na raiz da aplicação com esse nome onde ira conter todos os ficheiros com essa extensão.



## 2.7. Pesquisar no Código

Para pesquisar no código, comandos que são usados universalmente em outras aplicações como “CTRL + F” mostra uma caixa de pesquisa que permite navegar pelo código e encontrar o que o utilizador necessita.

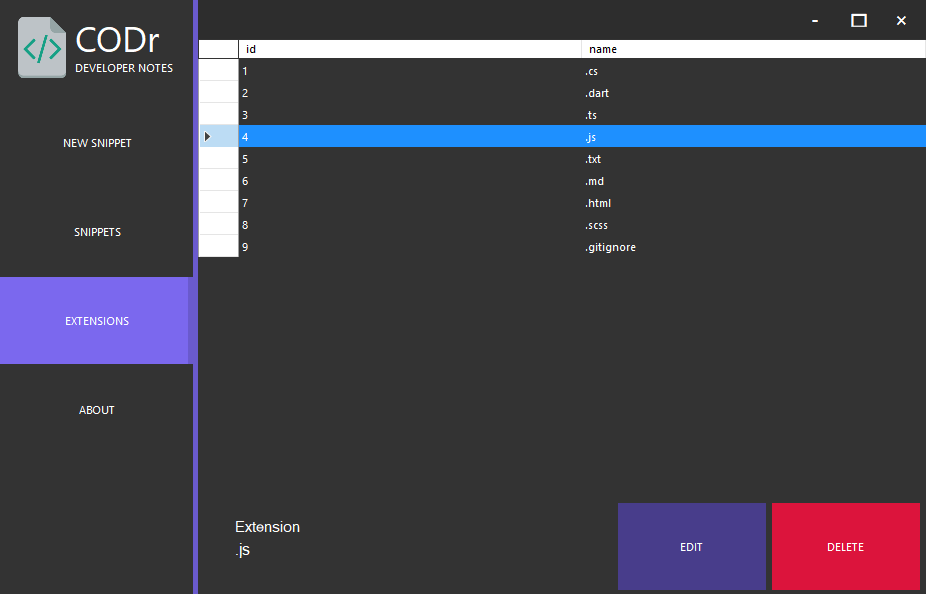


## 2.8. Extensoes

Aqui poderá verificar todas as extensões registadas, com a opcao de remover e editar.

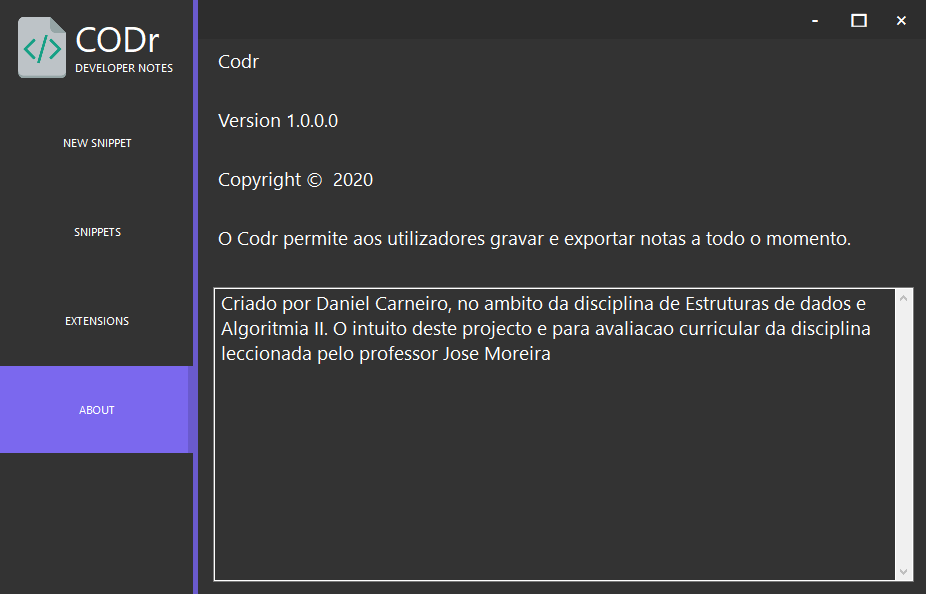
## 2.9. Editar e Remover

Para editar ou remover será primeiro preciso selecionar a linha que deseja e depois alterar o nome ou remover permanente.



## 2.10. Sobre

Aqui constara informação sobre o criador da aplicação, estes dados serão gerados automaticamente através da informação registada nas configurações do projecto.



# 3. Conclusão

Este trabalho foi essencialmente importante não pela demonstração que e possível fazer um software completo mas pelas técnicas que foram usadas, desde a painéis dinâmicos ate a diferentes logicas de codigo para obter resultados semelhantes. Pode-se afirmar que este projecto tem um pouco de tudo, ligação a uma base de dados em que permite adicionar, alterar e eliminar, permite também exportar dados e criar e editar ficheiros com *syntax highlight*.

# 4. Codificação C#

## 4.1 ClassCryptography.cs

using System.Security.Cryptography;

using System.Text;

namespace Codr.Core.Classes

{

public static class ClassCryptography

{

/// <summary>

/// Generates a SHA512 string for passwords

/// </summary>

/// <param name="pwd"></param>

/// <returns></returns>

public static string GenerateSHA512String(string pwd)

{

SHA512 sha512 = SHA512Managed.Create();

byte[] bytes = Encoding.UTF8.GetBytes(pwd);

byte[] computeHash = sha512.ComputeHash(bytes);

StringBuilder stringBuilder = new StringBuilder();

foreach (var hash in computeHash)

{

stringBuilder.Append(hash.ToString("X2"));

}

return stringBuilder.ToString();

}

}

}

## 4.2 ClassDatabase.cs

using System;

using System.Data;

using System.Data.SqlClient;

namespace Codr.Core.Classes

{

public class ClassBD

{

public class ClassSnippet

{

public string id { get; set; }

public string title { get; set; }

public string description { get; set; }

public int id\_extension { get; set; }

public string path { get; set; }

}

/// <summary>

/// Open the sql database connection

/// </summary>

/// <returns></returns>

public static SqlConnection OpenDatabase()

{

// reads the projects saved connection string

string StringConn = Properties.Settings.Default.DB\_CodrConnectionString;

// defines the sql connection

SqlConnection sqlConnection = new SqlConnection(StringConn);

// opens db with params

try

{

sqlConnection.Open();

}

catch (Exception)

{

return null;

}

return sqlConnection;

}

/// <summary>

/// Dynamic function that requires the sql command to be passed into

/// </summary>

/// <param name="sql"></param>

/// <returns></returns>

public static DataTable GetData(string sql)

{

// opens connection

SqlConnection sqlConnection = OpenDatabase();

// creates the sql command from params

string sqlCmd = sql;

// sets the command

SqlCommand cmd = new SqlCommand();

cmd = new SqlCommand(sqlCmd, sqlConnection);

DataTable table = new DataTable();

table.Load(cmd.ExecuteReader()); // loads data to datatable

return table;

}

}

}

## 4.3 ClassExtensions.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Codr.Core.Classes

{

public class ClassExtensions

{

/// <summary>

/// Gets the extension name by checking the extension id

/// </summary>

/// <param name="id\_extension"></param>

/// <returns></returns>

public static string GetExtensionName(int id\_extension)

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string SQL;

SQL = "SELECT name FROM extensions where id=@id\_extension";

SqlCommand sqlExtCommand = new SqlCommand(SQL, sqlConnection);

sqlExtCommand.Parameters.AddWithValue("id\_extension", id\_extension);

SqlDataReader sqlDataReader = sqlExtCommand.ExecuteReader();

string extName = "";

while (sqlDataReader.Read())

{

extName = sqlDataReader["name"].ToString();

}

sqlConnection.Close();

if (extName == "")

{

extName = AddExtension(extName);

}

return extName;

}

/// <summary>

/// Searches an extension name and returns the respective ID

/// </summary>

/// <param name="extension"></param>

/// <returns></returns>

public static string SearchExtensions(string extension)

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

if (sqlConnection == null)

{

return "";

}

string SQL;

SQL = "SELECT \* FROM extensions WHERE extensions.name = @extension";

SqlCommand sqlCommand = new SqlCommand(SQL, sqlConnection);

sqlCommand.Parameters.AddWithValue("extension", extension.Trim());

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

if (sqlDataReader.HasRows)

{

DataTable table = new DataTable();

table.Load(sqlDataReader); // loads data to the datatable

var id\_extension = table.Rows[0].ItemArray[0].ToString();

return id\_extension;

}

return "";

}

/// <summary>

/// Creates a new extension

/// </summary>

/// <param name="extension"></param>

/// <returns></returns>

public static string AddExtension(string extension)

{

try

{

// verifica se ja existe uma categoria com o mesmo nome

string extId = SearchExtensions(extension);

if (extId != "")

{

return extId;

}

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sqlCmd = "INSERT INTO extensions (name) VALUES (@name)";

if (sqlConnection == null)

{

MessageBox.Show("Unable to open extensions table",

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

SqlCommand cmd = new SqlCommand(sqlCmd, sqlConnection);

cmd.Parameters.AddWithValue("name", extension);

cmd.ExecuteNonQuery();

sqlConnection.Close();

return SearchExtensions(extension);

}

catch (Exception err)

{

MessageBox.Show("Unable to create extension \n" + err,

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

}

/// <summary>

/// Deletes an extension by id

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static bool DeleteExtension(int id)

{

try

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sql = "DELETE FROM extensions WHERE id=@id";

SqlCommand cmd = new SqlCommand();

cmd = new SqlCommand(sql, sqlConnection);

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

sqlConnection.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.ToString(), "CODr",

MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

}

/// <summary>

/// Edits an existing extension and assigns a new name

/// </summary>

/// <param name="id"></param>

/// <param name="newName"></param>

/// <returns></returns>

public static bool EditExtension(int id, string newName)

{

try

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sql = "UPDATE extensions SET name=@name WHERE id=@id";

SqlCommand cmd = new SqlCommand();

cmd = new SqlCommand(sql, sqlConnection);

cmd.Parameters.AddWithValue("id", id);

cmd.Parameters.AddWithValue("name", newName);

cmd.ExecuteNonQuery();

sqlConnection.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.ToString(), "CODr",

MessageBoxButtons.OK, MessageBoxIcon.Information);

return false;

}

}

}

}

## 4.4 ClassFile.cs

using System;

namespace Codr.Core.Classes

{

public class ClassFile

{

public static string rootPath = AppDomain.CurrentDomain.BaseDirectory;

public static string filesFolder = @rootPath + "Core\\Files\\";

}

}

## 4.5 ClassLogin.cs

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Codr.Core.Classes

{

public static class ClassLogin

{

/// <summary>

/// Logs the user into the software

/// </summary>

/// <param name="user"></param>

/// <param name="pwd"></param>

/// <returns></returns>

public static bool SignIn(string user, string pwd)

{

try

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sqlCmd = "SELECT \* FROM users WHERE username=@username AND password=@password";

if (sqlConnection == null)

{

MessageBox.Show("Unable to open users",

"CODr | Login", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

string hasPassw = ClassCryptography.GenerateSHA512String(pwd.Trim());

SqlCommand sqlCommand = new SqlCommand(sqlCmd, sqlConnection);

sqlCommand.Parameters.AddWithValue("username", user.Trim());

sqlCommand.Parameters.AddWithValue("password", hasPassw);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

if (sqlDataReader.Read() == true)

{

// Logged in successfully

sqlDataReader.Close();

return true;

}

sqlDataReader.Close();

return false;

}

catch (Exception err)

{

MessageBox.Show("Unable to login \n" + err,

"CODr | Login", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

/// <summary>

/// Allows users to register in the app

/// </summary>

/// <param name="name"></param>

/// <param name="user"></param>

/// <param name="pwd"></param>

/// <returns></returns>

public static bool Register(string name, string user, string pwd)

{

try

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sqlCmd = "INSERT INTO users (name, userName, password) VALUES (@name, @userName, @password)";

if (sqlConnection == null)

{

MessageBox.Show("Unable to open users",

"CODr | Login", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

string hasPassw = ClassCryptography.GenerateSHA512String(pwd.Trim());

SqlCommand sqlCommand = new SqlCommand(sqlCmd, sqlConnection);

sqlCommand.Parameters.AddWithValue("name", name.Trim());

sqlCommand.Parameters.AddWithValue("userName", user.Trim());

sqlCommand.Parameters.AddWithValue("password", hasPassw);

int execute = sqlCommand.ExecuteNonQuery();

if (execute > 0)

{

// Logged in successfully

sqlConnection.Close();

return true;

}

sqlConnection.Close();

return false;

}

catch (Exception err)

{

MessageBox.Show("Unable to register \n" + err,

"CODr | Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

}

}

## 4.6 ClassSnippets.cs

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using static Codr.Core.Classes.ClassBD;

namespace Codr.Core.Classes

{

class ClassSnippets

{

/// <summary>

/// Creates a new code snippet and also adds a new file on the main proj folder

/// </summary>

/// <param name="title"></param>

/// <param name="description"></param>

/// <param name="id\_extension"></param>

/// <returns></returns>

public static string AddSnippet(string title, string description, string id\_extension)

{

try

{

string extName = ClassExtensions.GetExtensionName(int.Parse(id\_extension));

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sqlCmd = "INSERT INTO snippets (title, description, id\_extension, path) VALUES " +

"(@title, @description, @id\_extension, @path)";

if (sqlConnection == null)

{

MessageBox.Show("Unable to open snippets",

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

string filePath = extName + "\\" + title + extName;

SqlCommand comando = new SqlCommand(sqlCmd, sqlConnection);

comando.Parameters.AddWithValue("title", title.Trim());

comando.Parameters.AddWithValue("description", description.Trim());

comando.Parameters.AddWithValue("id\_extension", id\_extension);

comando.Parameters.AddWithValue("path", filePath);

comando.ExecuteNonQuery();

sqlConnection.Close();

return filePath;

}

catch (Exception err)

{

MessageBox.Show("Nao foi possivel adicionar a extension \n" + err,

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

}

/// <summary>

/// Function that edits a particular snippet, creates a new file

/// </summary>

/// <param name="id"></param>

/// <param name="title"></param>

/// <param name="description"></param>

/// <param name="id\_extension"></param>

/// <returns></returns>

public static string EditSnippet(int id, string title, string description, string id\_extension)

{

try

{

string extName = ClassExtensions.GetExtensionName(int.Parse(id\_extension));

SqlConnection sqlConnection = ClassBD.OpenDatabase();

string sqlCmd = "UPDATE snippets " +

"SET title=@title, description=@description, id\_extension=@id\_extension, path=@path " +

"WHERE id=@id";

if (sqlConnection == null)

{

MessageBox.Show("Unable to open snippets",

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

string filePath = extName + "\\" + title + extName;

SqlCommand comando = new SqlCommand(sqlCmd, sqlConnection);

comando.Parameters.AddWithValue("id", id);

comando.Parameters.AddWithValue("title", title.Trim());

comando.Parameters.AddWithValue("description", description.Trim());

comando.Parameters.AddWithValue("id\_extension", id\_extension);

comando.Parameters.AddWithValue("path", filePath);

comando.ExecuteNonQuery();

sqlConnection.Close();

return filePath;

}

catch (Exception err)

{

MessageBox.Show("Unable to edit this snippet \n" + err,

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return "";

}

}

/// <summary>

/// Gets a specific snippet by ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static ClassSnippet OpenSnippet(string id)

{

// Opens DB

SqlConnection sqlConnection = ClassBD.OpenDatabase();

// checks if sql connection is successfull

if (sqlConnection == null)

{

return null;

}

// creates the sql select command

string SQL;

SQL = "SELECT \* FROM snippets WHERE snippets.id = @id";

SqlCommand sqlCommand = new SqlCommand(SQL, sqlConnection);

sqlCommand.Parameters.AddWithValue("id", id.Trim());

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

// checks if there is data

if (sqlDataReader.Read())

{

// creates a new snippet and assigns data to it

ClassSnippet snippet = new ClassSnippet();

snippet.id = id;

snippet.title = sqlDataReader["title"].ToString();

snippet.description = sqlDataReader["description"].ToString();

snippet.id\_extension = int.Parse(sqlDataReader["id\_extension"].ToString());

snippet.path = sqlDataReader["path"].ToString();

return snippet;

}

else

{

// Does not exist

return null;

}

}

}

}

## 4.7 CodeEditorHotKeys.cs

using System;

using System.Windows.Forms;

namespace Codr.Core.CodeEditor

{

class CodeEditorHotKeys

{

public static bool Enable = true;

/// <summary>

/// Adds a new hotKey

/// </summary>

/// <param name="form"></param>

/// <param name="function"></param>

/// <param name="key"></param>

/// <param name="ctrl"></param>

/// <param name="shift"></param>

/// <param name="alt"></param>

public static void AddHotKey(Form form, Action function, Keys key, bool ctrl = false, bool shift = false, bool alt = false)

{

form.KeyPreview = true;

form.KeyDown += delegate (object sender, KeyEventArgs e) {

if (IsHotkey(e, key, ctrl, shift, alt))

{

function();

}

};

}

/// <summary>

/// Checks if it is a hotKey

/// </summary>

/// <param name="eventData"></param>

/// <param name="key"></param>

/// <param name="ctrl"></param>

/// <param name="shift"></param>

/// <param name="alt"></param>

/// <returns></returns>

public static bool IsHotkey(KeyEventArgs eventData, Keys key, bool ctrl = false, bool shift = false, bool alt = false)

{

return eventData.KeyCode == key && eventData.Control == ctrl && eventData.Shift == shift && eventData.Alt == alt;

}

}

}

## 4.8 CodeEditorSearchManager.cs

using ScintillaNET;

using System.Windows.Forms;

namespace Codr.Core.CodeEditor

{

class CodeEditorSearchManager

{

public static ScintillaNET.Scintilla TextArea;

public static TextBox SearchBox;

public static string LastSearch = "";

public static int LastSearchIndex;

/// <summary>

/// Function that deals with code search

/// </summary>

/// <param name="next"></param>

/// <param name="incremental"></param>

public static void Find(bool next, bool incremental)

{

bool first = LastSearch != SearchBox.Text;

LastSearch = SearchBox.Text;

if (LastSearch.Length > 0)

{

if (next)

{

// SEARCH FOR THE NEXT OCCURANCE

// Search the document at the last search index

TextArea.TargetStart = LastSearchIndex - 1;

TextArea.TargetEnd = LastSearchIndex + (LastSearch.Length + 1);

TextArea.SearchFlags = SearchFlags.None;

// Search, and if not found..

if (!incremental || TextArea.SearchInTarget(LastSearch) == -1)

{

// Search the document from the caret onwards

TextArea.TargetStart = TextArea.CurrentPosition;

TextArea.TargetEnd = TextArea.TextLength;

TextArea.SearchFlags = SearchFlags.None;

// Search, and if not found..

if (TextArea.SearchInTarget(LastSearch) == -1)

{

// Search again from top

TextArea.TargetStart = 0;

TextArea.TargetEnd = TextArea.TextLength;

// Search, and if not found..

if (TextArea.SearchInTarget(LastSearch) == -1)

{

// clear selection and exit

TextArea.ClearSelections();

return;

}

}

}

}

else

{

// SEARCH FOR THE PREVIOUS OCCURANCE

// Search the document from the beginning to the caret

TextArea.TargetStart = 0;

TextArea.TargetEnd = TextArea.CurrentPosition;

TextArea.SearchFlags = SearchFlags.None;

// Search, and if not found..

if (TextArea.SearchInTarget(LastSearch) == -1)

{

// Search again from the caret onwards

TextArea.TargetStart = TextArea.CurrentPosition;

TextArea.TargetEnd = TextArea.TextLength;

// Search, and if not found..

if (TextArea.SearchInTarget(LastSearch) == -1)

{

// clear selection and exit

TextArea.ClearSelections();

return;

}

}

}

// Select the occurance

LastSearchIndex = TextArea.TargetStart;

TextArea.SetSelection(TextArea.TargetEnd, TextArea.TargetStart);

TextArea.ScrollCaret();

}

SearchBox.Focus();

}

}

}

## 4.9 Login.cs

using Codr.Core.Classes;

using System;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace Codr.Forms

{

public partial class Login : Form

{

public const int WM\_NCLBUTTONDOWN = 0xA1;

public const int HT\_CAPTION = 0x2;

[DllImportAttribute("user32.dll")]

public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImportAttribute("user32.dll")]

public static extern bool ReleaseCapture();

public Login()

{

InitializeComponent();

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HT\_CAPTION, 0);

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void button1\_Click(object sender, EventArgs e)

{

Close();

}

private void btnRegister\_Click(object sender, EventArgs e)

{

// all fields are required

if (textBoxRegName.Text == "" || textBoxRegUsername.Text == "" || TextBoxRegPassw.Text == "" || TextBoxRegPasswConf.Text == "")

{

MessageBox.Show("All fields are required!", "CODr | Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// in case passwords do not match throw error

if (TextBoxRegPassw.Text != TextBoxRegPasswConf.Text)

{

MessageBox.Show("The passwords do not match!", "CODr | Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

bool userLogin = ClassLogin.Register(textBoxRegName.Text, textBoxRegUsername.Text, TextBoxRegPassw.Text);

if (userLogin)

{

tabControl1.SelectedTab = loginTab; // redirects user to login with new credentials

}

else

{

MessageBox.Show("Unable to create an account!", "CODr | Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

ClearRegisterFields(); // clears fields

}

private void btnLogin\_Click(object sender, EventArgs e)

{

// username and password are required

if (TextBoxLoginPassword.Text == "" || textBoxLoginUsername.Text == "")

{

MessageBox.Show("Username and password fields are required!", "CODr | Login", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

bool userLogin = ClassLogin.SignIn(textBoxLoginUsername.Text, TextBoxLoginPassword.Text);

if (userLogin)

{

this.Hide();

Main main = new Main();

main.Show();

}

else

{

MessageBox.Show("The username or password are incorrect!", "CODr | Login", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

ClearLoginFields();

}

/// <summary>

/// Clears all login tab fields

/// </summary>

private void ClearLoginFields()

{

TextBoxLoginPassword.Clear();

textBoxLoginUsername.Clear();

}

/// <summary>

/// Clears all register tab fields

/// </summary>

private void ClearRegisterFields()

{

textBoxRegName.Clear();

textBoxRegUsername.Clear();

TextBoxRegPassw.Clear();

TextBoxRegPasswConf.Clear();

}

private void TextBoxLoginPassword\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

btnLogin\_Click(sender, e);

}

}

}

}

## 4.10 Main.cs

using Codr.Core.Classes;

using CodrApp;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace Codr.Forms

{

public partial class Main : Form

{

public const int WM\_NCLBUTTONDOWN = 0xA1;

public const int HT\_CAPTION = 0x2;

[DllImportAttribute("user32.dll")]

public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImportAttribute("user32.dll")]

public static extern bool ReleaseCapture();

public Main()

{

InitializeComponent();

}

private void Main\_Load(object sender, EventArgs e)

{

buttonNotas.PerformClick();

createFolders();

}

/// <summary>

/// Creates all folders in Bin/Debug/Core/Files

/// </summary>

private void createFolders()

{

SqlConnection sqlConnection = ClassBD.OpenDatabase();

//2. Criar comando SQL SELECT para submeter à BD

string SQL;

SQL = "SELECT name FROM extensions";

SqlCommand sqlCommand = new SqlCommand(SQL, sqlConnection);

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

List<string> folderNames = new List<string>();

while (sqlDataReader.Read())

{

folderNames.Add(sqlDataReader["name"].ToString());

}

foreach (var folder in folderNames)

{

var path = ClassFile.filesFolder + folder;

if (!Directory.Exists(path))

{

Directory.CreateDirectory(path);

}

}

}

private void buttonTarefas\_Click(object sender, EventArgs e)

{

panelMain.Controls.Clear();

panelActivo.Height = buttonNotas.Height;

panelActivo.Top = buttonNotas.Top;

buttonNotas.BackColor = Color.MediumSlateBlue;

buttonNovaTarefa.BackColor = Color.FromArgb(51, 51, 51);

buttonSobre.BackColor = Color.FromArgb(51, 51, 51);

buttonCategorias.BackColor = Color.FromArgb(51, 51, 51);

AllSnippets objForm = new AllSnippets();

objForm.TopLevel = false;

objForm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

objForm.Dock = DockStyle.Fill;

objForm.Show();

panelMain.Controls.Add(objForm);

}

private void buttonSobre\_Click(object sender, EventArgs e)

{

panelMain.Controls.Clear();

panelActivo.Height = buttonSobre.Height;

panelActivo.Top = buttonSobre.Top;

buttonNotas.BackColor = Color.FromArgb(51, 51, 51);

buttonNovaTarefa.BackColor = Color.FromArgb(51, 51, 51);

buttonSobre.BackColor = Color.MediumSlateBlue;

buttonCategorias.BackColor = Color.FromArgb(51, 51, 51);

Sobre objForm = new Sobre();

objForm.TopLevel = false;

objForm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

objForm.Dock = DockStyle.Fill;

objForm.Show();

panelMain.Controls.Add(objForm);

}

private void buttonNovaTarefa\_Click(object sender, EventArgs e)

{

panelMain.Controls.Clear();

panelActivo.Height = buttonNovaTarefa.Height;

panelActivo.Top = buttonNovaTarefa.Top;

buttonNotas.BackColor = Color.FromArgb(51, 51, 51);

buttonNovaTarefa.BackColor = Color.MediumSlateBlue;

buttonSobre.BackColor = Color.FromArgb(51, 51, 51);

buttonCategorias.BackColor = Color.FromArgb(51, 51, 51);

Snippets objForm = new Snippets();

objForm.TopLevel = false;

objForm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

objForm.Dock = DockStyle.Fill;

objForm.Show();

panelMain.Controls.Add(objForm);

}

private void buttonCategorias\_Click(object sender, EventArgs e)

{

panelMain.Controls.Clear();

panelActivo.Height = buttonCategorias.Height;

panelActivo.Top = buttonCategorias.Top;

buttonNotas.BackColor = Color.FromArgb(51, 51, 51);

buttonNovaTarefa.BackColor = Color.FromArgb(51, 51, 51);

buttonSobre.BackColor = Color.FromArgb(51, 51, 51);

buttonCategorias.BackColor = Color.MediumSlateBlue;

Extensions objForm = new Extensions();

objForm.TopLevel = false;

objForm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

objForm.Dock = DockStyle.Fill;

objForm.Show();

panelMain.Controls.Add(objForm);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

formMove(e);

}

private void panel4\_MouseMove(object sender, MouseEventArgs e)

{

formMove(e);

}

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

formMove(e);

}

private void formMove(MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HT\_CAPTION, 0);

}

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button2\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void buttonMaximizar\_Click(object sender, EventArgs e)

{

MaxMinJanela();

}

private void MaxMinJanela()

{

this.WindowState = this.WindowState == FormWindowState.Maximized ? FormWindowState.Normal : FormWindowState.Maximized;

buttonMaximizar.Text = this.WindowState == FormWindowState.Maximized ? "🗗" : "🗖";

}

private void panel2\_DoubleClick(object sender, EventArgs e)

{

MaxMinJanela();

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

}

}

}

## 4.11 About.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CodrApp

{

partial class Sobre : Form

{

public Sobre()

{

InitializeComponent();

this.Text = String.Format("About {0}", AssemblyTitle);

this.labelProductName.Text = AssemblyProduct;

this.labelVersion.Text = String.Format("Version {0}", AssemblyVersion);

this.labelCopyright.Text = AssemblyCopyright;

this.labelCompanyName.Text = AssemblyCompany;

this.textBoxDescription.Text = AssemblyDescription;

}

#region Assembly Attribute Accessors

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyCompany

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

}

}

## 4.12 AllSnippets.cs

using Codr.Core;

using Codr.Core.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

namespace CodrApp

{

public partial class AllSnippets : Form

{

public AllSnippets()

{

InitializeComponent();

}

private void buttonNovaTarefa\_Click(object sender, EventArgs e)

{

string filename = "";

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "CSV (\*.csv)|\*.csv";

saveFileDialog.FileName = "Output.csv";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

if (File.Exists(filename))

{

try

{

File.Delete(filename);

}

catch (IOException ex)

{

MessageBox.Show("It wasn't possible to write the data to the disk." + ex.Message, "CODr", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

int columnCount = dataGridView1.ColumnCount;

string columnNames = "";

string[] output = new string[dataGridView1.RowCount + 1];

for (int i = 0; i < columnCount; i++)

{

columnNames += dataGridView1.Columns[i].Name.ToString() + ",";

}

output[0] += columnNames;

for (int i = 1; (i - 1) < dataGridView1.RowCount; i++)

{

for (int j = 0; j < columnCount; j++)

{

output[i] += dataGridView1.Rows[i - 1].Cells[j].Value.ToString() + ",";

}

}

System.IO.File.WriteAllLines(saveFileDialog.FileName, output, System.Text.Encoding.UTF8);

}

}

private void AllSnippets\_Load(object sender, EventArgs e)

{

string comandoSQL = "SELECT snippets.id, snippets.title, snippets.description, snippets.path, extensions.name AS Extension " +

"FROM snippets INNER JOIN " +

"extensions ON snippets.id\_extension = extensions.id";

DataTable table = ClassBD.GetData(comandoSQL); // recebe os dados

dataGridView1.DataSource = table; // Mostra valores na grid

}

}

}

## 4.13 Extensions.cs

using Codr.Core.Classes;

using System;

using System.Data;

using System.Windows.Forms;

namespace Codr

{

public partial class Extensions : Form

{

public Extensions()

{

InitializeComponent();

}

private void Extensions\_Load(object sender, EventArgs e)

{

FillDatagrid();

ShowHideControls();

}

private void FillDatagrid()

{

string comandoSQL = "SELECT id, name from extensions ";

DataTable table = ClassBD.GetData(comandoSQL); // recebe os dados

dataGridView1.DataSource = table; // Mostra valores na grid

}

private void ShowHideControls(bool visible = false)

{

buttonDelete.Visible = visible ? true : false;

buttonEdit.Visible = visible ? true : false;

panel1.Visible = visible ? true : false;

if (!visible) dataGridView1.ClearSelection();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

Int32 selectedRowCount =

dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

if (selectedRowCount > 0)

{

ShowHideControls(true);

int selectedrowindex = dataGridView1.SelectedCells[0].RowIndex;

DataGridViewRow selectedRow = dataGridView1.Rows[selectedrowindex];

string id = Convert.ToString(selectedRow.Cells["id"].Value);

string name = Convert.ToString(selectedRow.Cells["name"].Value);

textBox1.Text = name;

}

}

private void buttonEdit\_Click(object sender, EventArgs e)

{

int selectedrowindex = dataGridView1.SelectedCells[0].RowIndex;

DataGridViewRow selectedRow = dataGridView1.Rows[selectedrowindex];

string id = Convert.ToString(selectedRow.Cells["id"].Value);

ClassExtensions.EditExtension(int.Parse(id), textBox1.Text);

FillDatagrid();

ShowHideControls();

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

int selectedrowindex = dataGridView1.SelectedCells[0].RowIndex;

DataGridViewRow selectedRow = dataGridView1.Rows[selectedrowindex];

string id = Convert.ToString(selectedRow.Cells["id"].Value);

ClassExtensions.DeleteExtension(int.Parse(id));

FillDatagrid();

ShowHideControls();

}

}

}

## 4.14 NewSnippet.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using Codr.Core.Classes;

using Codr.Core.CodeEditor;

using ScintillaNET;

namespace CodrApp

{

public partial class Snippets : Form

{

public Snippets()

{

InitializeComponent();

}

#region FormLoad

ScintillaNET.Scintilla TextArea;

private void Main\_Load(object sender, EventArgs e)

{

// CREATE CONTROL

TextArea = new ScintillaNET.Scintilla();

TextPanel.Controls.Add(TextArea);

// BASIC CONFIG

TextArea.Dock = System.Windows.Forms.DockStyle.Fill;

TextArea.TextChanged += this.OnTextChanged;

// INITIAL VIEW CONFIG

TextArea.WrapMode = WrapMode.None;

TextArea.IndentationGuides = IndentView.LookBoth;

// STYLING

InitColors();

InitSyntaxColoring();

// NUMBER MARGIN

InitNumberMargin();

// BOOKMARK MARGIN

InitBookmarkMargin();

// CODE FOLDING MARGIN

InitCodeFolding();

// DRAG DROP

InitDragDropFile();

// INIT HOTKEYS

InitHotkeys();

loadAllExtensions();

loadAllSnippets();

ClearFields();

textBoxId.Visible = false;

}

/// <summary>

/// Loads all extensions into the comboBox

/// </summary>

void loadAllExtensions()

{

string comandoSQL = "SELECT \* from extensions";

DataTable table = ClassBD.GetData(comandoSQL); // recebe os dados

comboboxExtensions.ValueMember = "id";

comboboxExtensions.DisplayMember = "name";

comboboxExtensions.DataSource = table;

comboboxExtensions.SelectedItem = null;

}

/// <summary>

/// Loads all snippets into the comboBox

/// </summary>

void loadAllSnippets()

{

string comandoSQL = "SELECT \* from snippets";

DataTable table = ClassBD.GetData(comandoSQL); // recebe os dados

listOfSnippets.ValueMember = "id";

listOfSnippets.DisplayMember = "title";

listOfSnippets.DataSource = table;

listOfSnippets.SelectedItem = null;

}

#endregion

private void listOfSnippets\_SelectedIndexChanged(object sender, EventArgs e)

{

object id = listOfSnippets.SelectedValue;

if (id != null)

{

var snippet = ClassSnippets.OpenSnippet(id.ToString());

textBoxId.Text = snippet.id;

textBoxTitulo.Text = snippet.title;

textBoxDescricao.Text = snippet.description;

comboboxExtensions.SelectedItem = snippet.id\_extension;

comboboxExtensions.Text = ClassExtensions.GetExtensionName(snippet.id\_extension);

LoadDataFromFile(ClassFile.filesFolder + snippet.path, true);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

try

{

if(textBoxTitulo.Text == "" || textBoxDescricao.Text == "" || TextArea.Text == "" || comboboxExtensions.Text == "")

{

MessageBox.Show("All Fields are required", "CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string idExt = ClassExtensions.AddExtension(comboboxExtensions.Text);

if (idExt != "")

{

string filePath;

if (listOfSnippets.SelectedIndex == -1)

{

filePath = ClassSnippets.AddSnippet(textBoxTitulo.Text, textBoxDescricao.Text, idExt);

}

else

{

filePath = ClassSnippets.EditSnippet(int.Parse(textBoxId.Text), textBoxTitulo.Text, textBoxDescricao.Text, idExt);

}

filePath = ClassFile.filesFolder + filePath;

if (filePath != "")

{

if(!Directory.Exists(ClassFile.filesFolder + comboboxExtensions.Text))

{

Directory.CreateDirectory(ClassFile.filesFolder + comboboxExtensions.Text);

}

File.WriteAllText(filePath, "");

using (StreamWriter strwriter = System.IO.File.AppendText(filePath))

{

strwriter.Write(this.TextArea.Text);

}

loadAllExtensions();

loadAllSnippets();

}

}

}

catch (Exception err)

{

MessageBox.Show("Unable to create the snippet, \n" + err,

"CODr", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonNew\_Click(object sender, EventArgs e)

{

ClearFields();

}

private void ClearFields()

{

textBoxId.Text = "";

textBoxTitulo.Text = "Title";

textBoxDescricao.Text = "Description";

TextArea.Text = "";

comboboxExtensions.SelectedIndex = -1;

listOfSnippets.SelectedIndex = -1;

}

#region Numbers, Bookmarks, Code Folding

/// <summary>

/// the background color of the text area

/// </summary>

private const int BACK\_COLOR = 0x383838;

/// <summary>

/// default text color of the text area

/// </summary>

private const int FORE\_COLOR = 0xc1c1c1;

/// <summary>

/// change this to whatever margin you want the line numbers to show in

/// </summary>

private const int NUMBER\_MARGIN = 1;

/// <summary>

/// change this to whatever margin you want the bookmarks/breakpoints to show in

/// </summary>

private const int BOOKMARK\_MARGIN = 2;

private const int BOOKMARK\_MARKER = 2;

/// <summary>

/// change this to whatever margin you want the code folding tree (+/-) to show in

/// </summary>

private const int FOLDING\_MARGIN = 3;

/// <summary>

/// set this true to show circular buttons for code folding (the [+] and [-] buttons on the margin)

/// </summary>

private const bool CODEFOLDING\_CIRCULAR = false;

private void InitNumberMargin()

{

TextArea.Styles[Style.LineNumber].BackColor = IntToColor(BACK\_COLOR);

TextArea.Styles[Style.LineNumber].ForeColor = IntToColor(FORE\_COLOR);

TextArea.Styles[Style.IndentGuide].ForeColor = IntToColor(FORE\_COLOR);

TextArea.Styles[Style.IndentGuide].BackColor = IntToColor(BACK\_COLOR);

var nums = TextArea.Margins[NUMBER\_MARGIN];

nums.Width = 30;

nums.Type = MarginType.Number;

nums.Sensitive = true;

nums.Mask = 0;

TextArea.MarginClick += TextArea\_MarginClick;

}

private void InitBookmarkMargin()

{

//TextArea.SetFoldMarginColor(true, IntToColor(BACK\_COLOR));

var margin = TextArea.Margins[BOOKMARK\_MARGIN];

margin.Width = 20;

margin.Sensitive = true;

margin.Type = MarginType.Symbol;

margin.Mask = (1 << BOOKMARK\_MARKER);

//margin.Cursor = MarginCursor.Arrow;

var marker = TextArea.Markers[BOOKMARK\_MARKER];

marker.Symbol = MarkerSymbol.Circle;

marker.SetBackColor(IntToColor(0xFF003B));

marker.SetForeColor(IntToColor(0x000000));

marker.SetAlpha(100);

}

private void InitCodeFolding()

{

TextArea.SetFoldMarginColor(true, IntToColor(BACK\_COLOR));

TextArea.SetFoldMarginHighlightColor(true, IntToColor(BACK\_COLOR));

// Enable code folding

TextArea.SetProperty("fold", "1");

TextArea.SetProperty("fold.compact", "1");

// Configure a margin to display folding symbols

TextArea.Margins[FOLDING\_MARGIN].Type = MarginType.Symbol;

TextArea.Margins[FOLDING\_MARGIN].Mask = Marker.MaskFolders;

TextArea.Margins[FOLDING\_MARGIN].Sensitive = true;

TextArea.Margins[FOLDING\_MARGIN].Width = 20;

// Set colors for all folding markers

for (int i = 25; i <= 31; i++)

{

TextArea.Markers[i].SetForeColor(IntToColor(BACK\_COLOR)); // styles for [+] and [-]

TextArea.Markers[i].SetBackColor(IntToColor(FORE\_COLOR)); // styles for [+] and [-]

}

// Configure folding markers with respective symbols

TextArea.Markers[Marker.Folder].Symbol = CODEFOLDING\_CIRCULAR ? MarkerSymbol.CirclePlus : MarkerSymbol.BoxPlus;

TextArea.Markers[Marker.FolderOpen].Symbol = CODEFOLDING\_CIRCULAR ? MarkerSymbol.CircleMinus : MarkerSymbol.BoxMinus;

TextArea.Markers[Marker.FolderEnd].Symbol = CODEFOLDING\_CIRCULAR ? MarkerSymbol.CirclePlusConnected : MarkerSymbol.BoxPlusConnected;

TextArea.Markers[Marker.FolderMidTail].Symbol = MarkerSymbol.TCorner;

TextArea.Markers[Marker.FolderOpenMid].Symbol = CODEFOLDING\_CIRCULAR ? MarkerSymbol.CircleMinusConnected : MarkerSymbol.BoxMinusConnected;

TextArea.Markers[Marker.FolderSub].Symbol = MarkerSymbol.VLine;

TextArea.Markers[Marker.FolderTail].Symbol = MarkerSymbol.LCorner;

// Enable automatic folding

TextArea.AutomaticFold = (AutomaticFold.Show | AutomaticFold.Click | AutomaticFold.Change);

}

private void TextArea\_MarginClick(object sender, MarginClickEventArgs e)

{

if (e.Margin == BOOKMARK\_MARGIN)

{

// Do we have a marker for this line?

const uint mask = (1 << BOOKMARK\_MARKER);

var line = TextArea.Lines[TextArea.LineFromPosition(e.Position)];

if ((line.MarkerGet() & mask) > 0)

{

// Remove existing bookmark

line.MarkerDelete(BOOKMARK\_MARKER);

}

else

{

// Add bookmark

line.MarkerAdd(BOOKMARK\_MARKER);

}

}

}

#endregion

#region HotKeys

private void OnTextChanged(object sender, EventArgs e)

{

}

private void InitColors()

{

TextArea.SetSelectionBackColor(true, IntToColor(0x114D9C));

}

public static Color IntToColor(int rgb)

{

return Color.FromArgb(255, (byte)(rgb >> 16), (byte)(rgb >> 8), (byte)rgb);

}

public void InvokeIfNeeded(Action action)

{

if (this.InvokeRequired)

{

this.BeginInvoke(action);

}

else

{

action.Invoke();

}

}

private void ZoomIn()

{

TextArea.ZoomIn();

}

private void ZoomOut()

{

TextArea.ZoomOut();

}

private void InitHotkeys()

{

// register the hotkeys with the form

CodeEditorHotKeys.AddHotKey(this, OpenSearch, Keys.F, true);

CodeEditorHotKeys.AddHotKey(this, ZoomIn, Keys.Oemplus, true);

CodeEditorHotKeys.AddHotKey(this, ZoomOut, Keys.OemMinus, true);

CodeEditorHotKeys.AddHotKey(this, CloseSearch, Keys.Escape);

// allows user to use ctrl + f to search

TextArea.ClearCmdKey(Keys.Control | Keys.F);

}

private void InitSyntaxColoring()

{

// Configure the default style

TextArea.StyleResetDefault();

TextArea.Styles[Style.Default].Font = "Segoe UI";

TextArea.Styles[Style.Default].Size = 12;

TextArea.Styles[Style.Default].BackColor = IntToColor(0x3C3C3C);

TextArea.Styles[Style.Default].ForeColor = IntToColor(0xFFFFFF);

TextArea.StyleClearAll();

// Configure the CPP (C#) lexer styles

TextArea.Styles[Style.Cpp.Identifier].ForeColor = IntToColor(0xD0DAE2);

TextArea.Styles[Style.Cpp.Comment].ForeColor = IntToColor(0xBD758B);

TextArea.Styles[Style.Cpp.CommentLine].ForeColor = IntToColor(0x40BF57);

TextArea.Styles[Style.Cpp.CommentDoc].ForeColor = IntToColor(0x2FAE35);

TextArea.Styles[Style.Cpp.Number].ForeColor = IntToColor(0xFFFF00);

TextArea.Styles[Style.Cpp.String].ForeColor = IntToColor(0xFFFF00);

TextArea.Styles[Style.Cpp.Character].ForeColor = IntToColor(0xE95454);

TextArea.Styles[Style.Cpp.Preprocessor].ForeColor = IntToColor(0x8AAFEE);

TextArea.Styles[Style.Cpp.Operator].ForeColor = IntToColor(0xE0E0E0);

TextArea.Styles[Style.Cpp.Regex].ForeColor = IntToColor(0xff00ff);

TextArea.Styles[Style.Cpp.CommentLineDoc].ForeColor = IntToColor(0x77A7DB);

TextArea.Styles[Style.Cpp.Word].ForeColor = IntToColor(0x48A8EE);

TextArea.Styles[Style.Cpp.Word2].ForeColor = IntToColor(0xF98906);

TextArea.Styles[Style.Cpp.CommentDocKeyword].ForeColor = IntToColor(0xB3D991);

TextArea.Styles[Style.Cpp.CommentDocKeywordError].ForeColor = IntToColor(0xFF0000);

TextArea.Styles[Style.Cpp.GlobalClass].ForeColor = IntToColor(0x48A8EE);

TextArea.Lexer = Lexer.Cpp;

TextArea.SetKeywords(0, "class extends implements import interface new case do while else if for in switch throw get set function var try catch finally while with default break continue delete return each const namespace package include use is as instanceof typeof author copy default deprecated eventType example exampleText exception haxe inheritDoc internal link mtasc mxmlc param private return see serial serialData serialField since throws usage version langversion playerversion productversion dynamic private public partial static intrinsic internal native override protected AS3 final super this arguments null Infinity NaN undefined true false abstract as base bool break by byte case catch char checked class const continue decimal default delegate do double descending explicit event extern else enum false finally fixed float for foreach from goto group if implicit in int interface internal into is lock long new null namespace object operator out override orderby params private protected public readonly ref return switch struct sbyte sealed short sizeof stackalloc static string select this throw true try typeof uint ulong unchecked unsafe ushort using var virtual volatile void while where yield");

TextArea.SetKeywords(1, "void Null ArgumentError arguments Array Boolean Class Date DefinitionError Error EvalError Function int Math Namespace Number Object RangeError ReferenceError RegExp SecurityError String SyntaxError TypeError uint XML XMLList Boolean Byte Char DateTime Decimal Double Int16 Int32 Int64 IntPtr SByte Single UInt16 UInt32 UInt64 UIntPtr Void Path File System Windows Forms ScintillaNET");

}

#endregion

#region Quick Search Bar

bool SearchIsOpen = false;

private void OpenSearch()

{

CodeEditorSearchManager.SearchBox = TxtSearch;

CodeEditorSearchManager.TextArea = TextArea;

if (!SearchIsOpen)

{

SearchIsOpen = true;

InvokeIfNeeded(delegate ()

{

PanelSearch.Visible = true;

TxtSearch.Text = CodeEditorSearchManager.LastSearch;

TxtSearch.Focus();

TxtSearch.SelectAll();

});

}

else

{

InvokeIfNeeded(delegate ()

{

TxtSearch.Focus();

TxtSearch.SelectAll();

});

}

}

private void CloseSearch()

{

if (SearchIsOpen)

{

SearchIsOpen = false;

InvokeIfNeeded(delegate ()

{

PanelSearch.Visible = false;

//CurBrowser.GetBrowser().StopFinding(true);

});

}

}

private void BtnClearSearch\_Click(object sender, EventArgs e)

{

CloseSearch();

}

private void BtnPrevSearch\_Click(object sender, EventArgs e)

{

CodeEditorSearchManager.Find(false, false);

}

private void BtnNextSearch\_Click(object sender, EventArgs e)

{

CodeEditorSearchManager.Find(true, false);

}

private void TxtSearch\_TextChanged(object sender, EventArgs e)

{

CodeEditorSearchManager.Find(true, true);

}

private void TxtSearch\_KeyDown(object sender, KeyEventArgs e)

{

if (CodeEditorHotKeys.IsHotkey(e, Keys.Enter))

{

CodeEditorSearchManager.Find(true, false);

}

if (CodeEditorHotKeys.IsHotkey(e, Keys.Enter, true) || CodeEditorHotKeys.IsHotkey(e, Keys.Enter, false, true))

{

CodeEditorSearchManager.Find(false, false);

}

}

#endregion

#region Drag & Drop File

public void InitDragDropFile()

{

TextArea.AllowDrop = true;

TextArea.DragEnter += delegate (object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.FileDrop))

e.Effect = DragDropEffects.Copy;

else

e.Effect = DragDropEffects.None;

};

TextArea.DragDrop += delegate (object sender, DragEventArgs e)

{

// get file drop

if (e.Data.GetDataPresent(DataFormats.FileDrop))

{

Array a = (Array)e.Data.GetData(DataFormats.FileDrop);

if (a != null)

{

string path = a.GetValue(0).ToString();

LoadDataFromFile(path);

}

}

};

}

private void LoadDataFromFile(string path, bool hasTitle = false)

{

if (File.Exists(path))

{

if (!hasTitle)

{

textBoxTitulo.Text = Path.GetFileName(path);

}

TextArea.Text = File.ReadAllText(path);

}

}

#endregion

}

}

# 5. Codificação SQL

CREATE TABLE [dbo].[users] (

[id] INT IDENTITY (1, 1) NOT NULL,

[userName] NVARCHAR(150) NULL,

[password] NVARCHAR(150) NULL,

[name] NVARCHAR(200) NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

CREATE TABLE [dbo].[extensions] (

[id] INT IDENTITY (1, 1) NOT NULL,

[name] NVARCHAR(10) NOT NULL,

[details] NVARCHAR(500) NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

CREATE TABLE [dbo].[snippets] (

[id] INT IDENTITY (1, 1) NOT NULL,

[title] NVARCHAR(500) NOT NULL,

[description] NVARCHAR(500) NULL,

[id\_extension] INT NULL,

[path] NVARCHAR(1000) NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([id\_extension]) REFERENCES [dbo].[extensions] ([id])

);