

Колледж космического машиностроения и технологий

**ОТЧЕТ**

**по УП.02.01 «Разработка и администрирование баз данных»**

**специальности 09.02.03 «Программирование в компьютерных системах»**

Выполнил студент

Волков М.А.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

**Королев 2021 г.**

**Содержание**

[1. Проектирование базы данных 3](#_Toc87901478)

[1.1. Построение ER-модель базы данных 3](#_Toc87901479)

[2. Разработка приложения 4](#_Toc87901480)

[2.1. Разработка таблицы и связи 4](#_Toc87901481)

[2.2. Разработка хранимые процедуры 5](#_Toc87901482)

[2.3. Разработка клиентское приложение 6](#_Toc87901483)

[Приложение 10](#_Toc87901484)

# Проектирование базы данных

## Построение ER-модель базы данных

Рисунок 1.

ER диаграмма №1

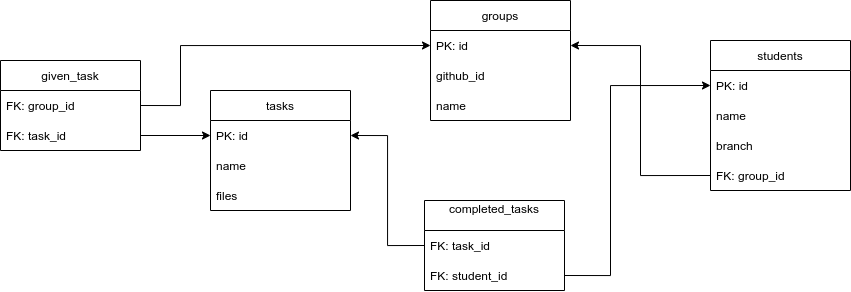
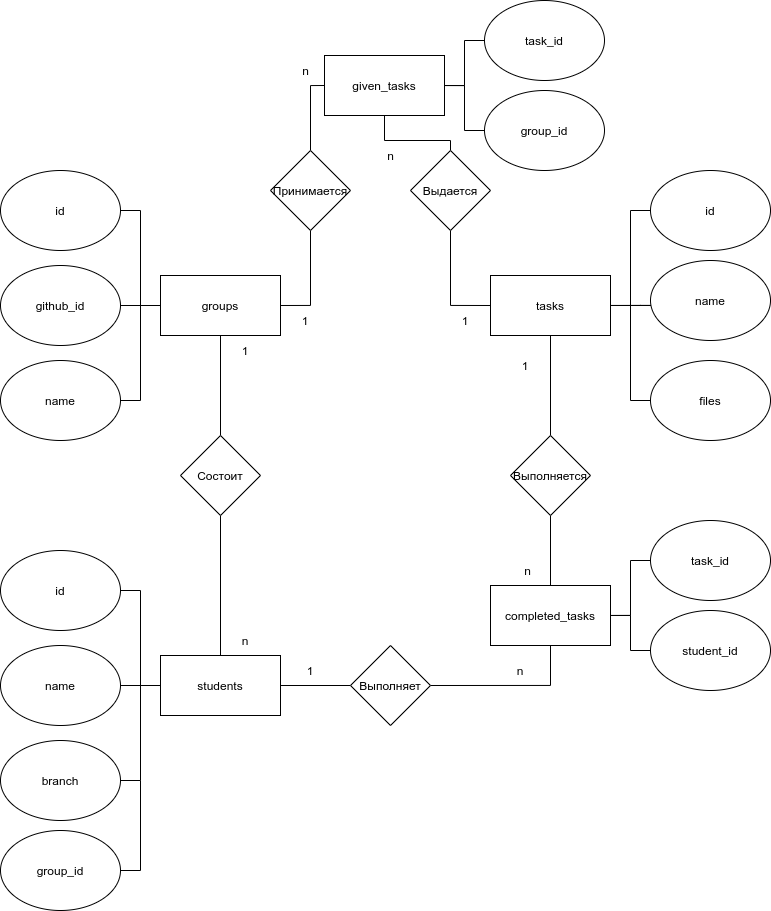


Рисунок 2.

ER диаграмма №2



# Разработка приложения

## Разработка таблицы и связи

Для приложения было разработано 5 таблиц. Их описания вы можете прочитать ниже, в таблице 1

Таблица 1.

Описание таблиц.

|  |  |
| --- | --- |
| **Имя таблицы** | **Описание** |
| tasks | Все задания системы |
| groups | Группы |
| students | Студенты групп |
| completed\_tasks | Завершенные студентами задания |
| given\_tasks | Выданные группам задания |

Таблица на листинге 1 хранит в себе все задания, которые выдал преподаватель. В поле «name» лежит имя задания, а в «files» лежат пути для проверки наличия файлов.

Листинг 1. Создание таблицы заданий

|  |
| --- |
| CREATE TABLE "tasks"  (  "id" INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  "name" varchar(64) NOT NULL,  "files" varchar(128) NOT NULL  ); |

Таблица на листинге 2 хранит в себе группы, с которыми работает преподаватель. Группы имеют айди Гитхаба где они выкладывают файлы и имя.

Листинг 2. Создание таблицы групп

|  |
| --- |
| CREATE TABLE "groups"  (  "id" INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  "github\_id" INTEGER,  "name" varchar(64) NOT NULL  ); |

Таблица на листинге 3 хранит в себе группы с которыми работает преподаватель. Группы имеют айди Гитхаба где они выкладывают файлы и имя.

Листинг 3. Создание таблицы студентов

|  |
| --- |
| CREATE TABLE "students"  (  "id" INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  "name" varchar(64) NOT NULL,  "branch" varchar(128) NOT NULL,  "group\_id" INTEGER,  FOREIGN KEY ("group\_id") REFERENCES "groups" ("id") ON DELETE CASCADE  ); |

Листинг 4 показывает нам создание таблицы выполненных заданий с айди выполнившего студента и айди выполненного задания.

Листинг 4. Создание таблицы выполненных заданий

|  |
| --- |
| CREATE TABLE "completed\_tasks"  (  "task\_id" INTEGER,  "student\_id" INTEGER,  FOREIGN KEY ("task\_id") REFERENCES "tasks" ("id") ON DELETE CASCADE,  FOREIGN KEY ("student\_id") REFERENCES "students" ("id") ON DELETE CASCADE  ); |

На последнем листинге находится запрос создания таблицы выданных заданий группам, с двумя внешними ключами, ссылающимися на группу и задание.

Листинг 5. Создание таблицы выданных заданий

|  |
| --- |
| CREATE TABLE "given\_tasks"  (  "task\_id" INTEGER,  "group\_id" INTEGER,  FOREIGN KEY ("task\_id") REFERENCES "tasks" ("id") ON DELETE CASCADE,  FOREIGN KEY ("group\_id") REFERENCES "groups" ("id") ON DELETE CASCADE  ); |

## Разработка хранимые процедуры

Мы разработали хранимую в БД процедуру CompleteTask, чтобы ускорить выполнение частых запросов к БД. На листинге 5 вы можете видеть её код.

Листинг 6. Процедура, хранимая в БД.

|  |
| --- |
| CREATE PROCEDURE CompleteTask(StudentId INTEGER, TaskId INTEGER)  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO "completed\_tasks"  ("task\_id", "student\_id")  VALUES  (@TaskId, @StudentId);  end; $$ |

Эта процедура упрощает внос данных в БД о выполнении задания студентом. Она принимает первичный ключ (айди) студента и первичный ключ (айди) выполненного задания. После этого она вставляет эти два айди в таблицу «completed\_tasks».

Также есть аналогичная функция, но для выдачи заданий группам на работу (Листинг 7).

Листинг 7. Процедура выдачи заданий.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GiveTask(GroupId INTEGER, TaskId INTEGER)  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO "given\_tasks"  ("task\_id", "group\_id")  VALUES  (@TaskId, @GroupId);  end; $$ |

Всего для приложения было разработано 2 процедуры. Их описания вы можете прочитать ниже, в таблице 2.

Таблица 2.

Описание процедур.

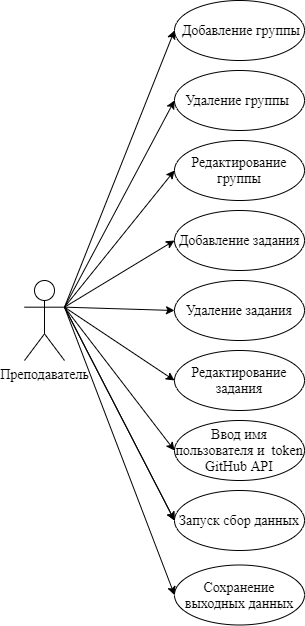
|  |  |
| --- | --- |
| **Имя таблицы** | **Описание** |
| CompleteTask(StudentId INTEGER, TaskId INTEGER) | Ставит отметку о выполнении задания с номером TaskId студентом с номером StudentId. |
| GiveTask(GroupId INTEGER, TaskId INTEGER) | Выдает задание c номером TaskId группе с номером GroupId. |

## Разработка клиентское приложение

Здесь мы можем видеть диаграмму прецедентов для приложения, на которой показаны возможные функциональные отношения (Рисунок 3).

Рисунок 3.

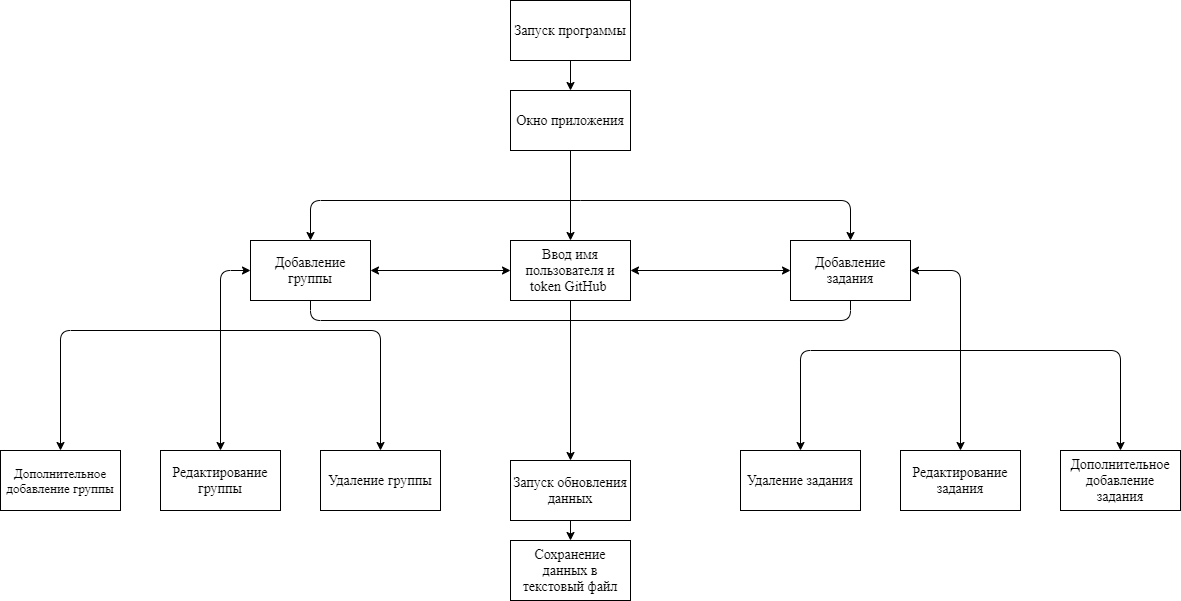
Диаграмма прецедентов приложения



Дальше приведен сценарий использования приложения пользователем (Рисунок 4).

Рисунок 4.

Сценарий использования приложения.



При запуске программы открывается окно приложения и у пользователя будет три действия на выбор, это: добавление группы, ввод имя пользователя и токен GitHub, добавление задания. Неважно, в каком порядке выполнять, главное сделать эти три действия, чтобы запустить обновление данных. Когда пользователь добавляет группу, появится 3 действия, это: редактирование группы, удаление группы и дополнительное добавление группы.

Когда пользователь добавляет задание, появится 3 действия, это: редактирование задания, удаление задания и дополнительное добавление задания.

После того, как пользователь обновил данные, их можно будет сохранить в текстовый файл.

Ниже представлена таблица с всеми файлами проекта и что они делают.

Таблица 3.

Файлы проекта

|  |  |
| --- | --- |
| **Файл** | **Описание** |
| appdatabase.pas | Модуль базы данных приложения |
| config.pas | Модуль работы с конфигурацией |
| groupformunit.lfm | Дизайн формы редактирования группы |
| groupformunit.pas | Модуль формы редактирования группы |
| icon.ico | Иконка приложения |
| KkmtTesting.ico | Иконка приложения |
| KkmtTesting.lpi | Файл Lazarus-проекта |
| KkmtTesting.lpr | Главный файл кода проекта |
| KkmtTesting.lps | Файл Lazarus-окружения проекта |
| KkmtTesting.res | Файл ресурсов проекта |
| lib | Третье-сторонние библиотеки и модули |
| mainformunit.lfm | Дизайн главной формы |
| mainformunit.pas | Модуль главной формы |
| README.md | Краткое описание проекта |
| resources | Папка с ресурсами в проекте |
| settingsformunit.lfm | Дизайн формы редактирования настроек |
| settingsformunit.pas | Модуль формы редактирования настроек |
| taskformunit.lfm | Дизайн формы редактирования задания |
| taskformunit.pas | Модуль формы редактирования задания |
| utils.pas | Модуль утилитарных функций проекта |
| 'Руководство по использованию программы KkmtTesting.docx' | Пользовательское руководство к программе |

# Приложение

**Приложение 1. MainFormUnit.pas**

unit MainFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, CheckLst,

ExtCtrls, Grids, ValEdit, ComCtrls, SQLite3, jsonparser, fpjson, fphttpclient,

Types, LCLIntf, Buttons,

Utils, AppDatabase,

GroupFormUnit, TaskFormUnit, SettingsFormUnit;

type

{ TMainForm }

// Класс главной формы

TMainForm = class(TForm)

// Кнопка добавления группы

AddGroupButton: TBitBtn;

// Кнопока добавления группы

AddTaskList: TBitBtn;

// Кнопка удаления группы

DeleteGroupButton: TBitBtn;

// Кнопка удаления таска

DeleteTaskButton: TBitBtn;

// Кнопка редактирования группы

EditGroupButton: TBitBtn;

// Группа редактирования таска

EditTaskButton: TBitBtn;

IconList: TImageList;

Panel3: TPanel;

// Кнопка сохранения репорта

SaveStatusReportButton: TBitBtn;

Panel1: TPanel;

Panel2: TPanel;

ProgressBar1: TProgressBar;

// Кнопка настроек

SettingsButton: TBitBtn;

Splitter1: TSplitter;

Splitter2: TSplitter;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

// Лист групп

GroupsList: TListBox;

StaticText1: TStaticText;

StaticText2: TStaticText;

// Лист тасков

TasksList: TListBox;

// Главный лист с статусом выполнения заданий

ListView1: TListView;

TasksPanel: TPanel;

GroupsPanel: TPanel;

// Кнопка обновления репорта

UpdateStatusButton: TBitBtn;

// Нажатие на кнопку добавления группы

procedure AddGroupButtonClick(Sender: TObject);

// Нажатие на кнопку добавления таска

procedure AddTaskListClick(Sender: TObject);

// Нажатие лист групп

procedure GroupsListClick(Sender: TObject);

// Двойное нажатие на лист с статусами заданий

procedure ListView1DblClick(Sender: TObject);

// Нажатие на кнопку сохранения репорта

procedure SaveStatusReportButtonClick(Sender: TObject);

// Нажатие на кнопку настроек

procedure SettingsButtonClick(Sender: TObject);

// Нажатие на лист тасков

procedure TasksListClick(Sender: TObject);

// Нажатие на кнопку обновления статусов заданий

procedure UpdateStatusButtonClick(Sender: TObject);

// Нажатие на кнопку удаления группы

procedure DeleteGroupButtonClick(Sender: TObject);

// Нажатие на кнопку удаления таска

procedure DeleteTaskButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования группы

procedure EditGroupButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования таска

procedure EditTaskButtonClick(Sender: TObject);

// Создание формы

procedure FormCreate(Sender: TObject);

// Функция обновления листа

procedure UpdateList(TableName: String; UpdatingList: TListBox);

//

procedure UpdateCurrentGroup;

//

procedure UpdateCurrentTask;

// Toggle блокировку формы

procedure ChangeFormBlock(AEnabled: Boolean);

private

// Айди выбранной группы

SelectedGroup: Integer;

// Айди выбранного таска

SelectedTask: Integer;

// Айди загруженной группы

LoadedGroup: Integer;

// Айди загруженного таска

LoadedTask: Integer;

public

// Обновляется ли сейчас статус задания

NowUpdating: Boolean;

end;

var

MainForm: TMainForm;

implementation

{$R \*.lfm}

type

{ TReportUpdateThread }

TReportUpdateThread = class(TThread)

private

// Нужна ли остановка обновления

FStop: Boolean;

// Количество веток для проверки

FMax: Integer;

// Уже проверено веток

FProgress: Integer;

// Айди выбранной группы

FSelectedGroup: Integer;

// Айди выбранного таска

FSelectedTask: Integer;

// Лист итемов для добавления в вывод

FListItemToAdd: TListItem;

// Изменить прогресс бар

procedure ChangeProgressBar;

// Обновить прогресс

procedure UpdateProgress;

// Проверка была выполнена успешно

procedure SuccessfulExecute;

protected  
 // Главная функция потока

procedure Execute; override;

// Вызовится в любом случае после завершения работы

procedure AfterExecute(Sender: TObject);

public

constructor Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

// Запуск проверки

procedure Start;

// Остановка проверки

procedure StopUpdating;

end;

var

// Поток проверки заданий

Thread: TReportUpdateThread;

{ TMainForm }

procedure TMainForm.FormCreate(Sender: TObject);

begin

UpdateList('groups', GroupsList);

UpdateList('tasks', TasksList);

SelectedGroup := -1;

SelectedTask := -1;

LoadedGroup := -1;

LoadedTask := -1;

end;

procedure TMainForm.UpdateStatusButtonClick(Sender: TObject);

begin

if not NowUpdating then

begin

if (SelectedGroup = -1) and (SelectedTask = -1) then exit;

Label2.Caption := StaticText2.Caption;

Label4.Caption := StaticText1.Caption;

Thread := TReportUpdateThread.Create(True, SelectedGroup, SelectedTask);

Thread.Start;

end

else

begin

Thread.StopUpdating;

end;

end;

procedure TMainForm.SaveStatusReportButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

ResultTable: TTable;

GroupName, TaskName: String;

ReportContents: TStringList;

FileName: String;

ListItem: TListItem;

begin

if (LoadedGroup = -1) or (LoadedTask = -1) then

exit;

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT name FROM groups WHERE id=:id', [par('id', LoadedGroup)]);

GroupName := ResultTable[0].KeyData['name'];

ResultTable := AppDB.ExecOut('SELECT name FROM tasks WHERE id=:id', [par('id', LoadedTask)]);

TaskName := ResultTable[0].KeyData['name'];

ResultTable.Free;

AppDB.Free;

ReportContents := TStringList.Create;

FileName := Format('%s\_%s\_%s',

[GroupName, TaskName, FormatDateTime('DD-MM-YYYY\_hh-nn-ss', Now)]);

ReportContents.Add('KKMT TASKS REPORT');

ReportContents.Add(Format('%s', [FileName]));

ReportContents.Add('');

for ListItem in ListView1.Items do

ReportContents.Add(Format('%s , %s , %s',

[ListItem.Caption, ListItem.SubItems[0], ListItem.SubItems[1]]));

ForceDirectories(Application.Location+'/reports');

ReportContents.SaveToFile(Application.Location+'/reports/'+FileName+'.txt');

end;

procedure TMainForm.SettingsButtonClick(Sender: TObject);

var

ModalForm: TSettingsForm;

begin

ModalForm := TSettingsForm.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

procedure TMainForm.ListView1DblClick(Sender: TObject);

begin

if ListView1.ItemIndex < 0 then

exit;

OpenURL(ListView1.Items[ListView1.ItemIndex].SubItems.Strings[0]);

end;

{ === GroupsPanel === }

procedure TMainForm.GroupsListClick(Sender: TObject);

begin

if GroupsList.ItemIndex < 0 then

exit;

SelectedGroup := TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I;

StaticText2.Caption := GroupsList.GetSelectedText;

end;

procedure TMainForm.AddGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

ModalForm := TGroupForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

//OpenDialog(TGroupForm);

UpdateList('groups', GroupsList);

end;

procedure TMainForm.DeleteGroupButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if GroupsList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM groups WHERE id=:id',

[par('id', TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I)]);

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

end;

procedure TMainForm.EditGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

if GroupsList.ItemIndex >= 0 then

begin

ModalForm := TGroupForm.Create(MainForm, TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOK then

begin

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

ModalForm.Free;

end;

end;

{ === TasksPanel === }

procedure TMainForm.TasksListClick(Sender: TObject);

begin

if TasksList.ItemIndex < 0 then

exit;

SelectedTask := TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I;

StaticText1.Caption := TasksList.GetSelectedText;

end;

procedure TMainForm.AddTaskListClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

ModalForm := TTaskForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

UpdateList('tasks', TasksList);

end;

procedure TMainForm.DeleteTaskButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if TasksList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM tasks WHERE id=:id',

[par('id', TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I)]);

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

end;

procedure TMainForm.EditTaskButtonClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

if TasksList.ItemIndex >= 0 then

begin

ModalForm := TTaskForm.Create(MainForm, TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOk then

begin

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

ModalForm.Free;

end;

end;

procedure TMainForm.UpdateList(TableName: String; UpdatingList: TListBox);

var

AppDB: TAppDatabase;

Table: TTable;

Row: TRow;

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM '+TableName+';');

UpdatingList.Clear;

for Row in Table do

begin

UpdatingList.Items.AddObject(Row.KeyData['name'], TIntObj.Create(Row.KeyData['id']));

end;

end;

procedure TMainForm.UpdateCurrentGroup;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedGroup <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', SelectedGroup)]);

if Table.Count = 0 then

begin

SelectedGroup := -1;

Label2.Caption := '';

end

else

begin

Label2.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedGroup <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', LoadedGroup)]);

if Table.Count = 0 then

begin

LoadedGroup := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.UpdateCurrentTask;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedTask <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', SelectedTask)]);

if Table.Count = 0 then

begin

SelectedTask := -1;

Label4.Caption := '';

end

else

begin

Label4.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedTask <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', LoadedTask)]);

if Table.Count = 0 then

begin

LoadedTask := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.ChangeFormBlock(AEnabled: Boolean);

begin

GroupsList.Enabled := AEnabled;

AddGroupButton.Enabled := AEnabled;

DeleteGroupButton.Enabled := AEnabled;

EditGroupButton.Enabled := AEnabled;

TasksList.Enabled := AEnabled;

AddTaskList.Enabled := AEnabled;

DeleteTaskButton.Enabled := AEnabled;

EditTaskButton.Enabled := AEnabled;

SaveStatusReportButton.Enabled := AEnabled;

SettingsButton.Enabled := AEnabled;

end;

{ TReportUpdateThread }

constructor TReportUpdateThread.Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

begin

inherited Create(CreateSuspended);

FreeOnTerminate := True;

FStop := False;

FProgress := 0;

FSelectedGroup := SelectedGroup;

FSelectedTask := SelectedTask;

FListItemToAdd := nil;

OnTerminate := @AfterExecute;

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstMarquee;

MainForm.ProgressBar1.Position := 0;

MainForm.ListView1.Clear;

end;

procedure TReportUpdateThread.Start;

begin

MainForm.NowUpdating := True;

MainForm.UpdateStatusButton.ImageIndex := 1;

MainForm.ChangeFormBlock(False);

inherited Start;

end;

procedure TReportUpdateThread.StopUpdating;

begin

FStop := True;

end;

procedure TReportUpdateThread.AfterExecute(Sender: TObject);

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Position := 0;

MainForm.NowUpdating := False;

MainForm.UpdateStatusButton.ImageIndex := 3;

MainForm.ChangeFormBlock(True);

end;

procedure TReportUpdateThread.ChangeProgressBar;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Max := FMax;

end;

procedure TReportUpdateThread.UpdateProgress;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.StepIt;

MainForm.ListView1.Items.AddItem(FListItemToAdd);

end;

procedure TReportUpdateThread.SuccessfulExecute;

begin

MainForm.LoadedGroup := FSelectedGroup;

MainForm.LoadedTask := FSelectedTask;

end;

procedure TReportUpdateThread.Execute;

var

HTTP: TFPHttpClient;

AppDB: TAppDatabase;

Table: TTable;

GitHubId: String;

HttpResult: String;

BranchesJSON: TJSONArray;

Enum, Enum2: TJSONEnum;

Files: TStringList;

Counter: Integer;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT github\_id FROM groups WHERE id=:id;', [par('id', FSelectedGroup)]);

if Table.Count <= 0 then

exit;

GitHubId := Table[0].KeyData['github\_id'];

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', FSelectedTask)]);

if Table.Count <= 0 then

exit;

Files := TStringList.Create;

for Enum in TJSONArray(GetJSON(Table[0].KeyData['files'])) do

Files.Add(Enum.Value.AsString);

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/branches', [GitHubId] ));

except

MessageDlg('Ошибка получения данных о ветках репозитория. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

exit;

end;

BranchesJSON := TJSONArray(GetJSON(HttpResult));

FMax := BranchesJSON.Count;

Synchronize(@ChangeProgressBar);

for Enum in BranchesJSON do

begin

if FStop then break;

FListItemToAdd := TListItem.Create(MainForm.ListView1.Items);

FListItemToAdd.Caption := TJSONObject(enum.Value)['name'].AsString;

FListItemToAdd.SubItems.Add(Format('https://github.com/%s/tree/%s',

[GitHubId, TJSONObject(enum.Value)['name'].AsString]));

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/git/trees/%s?recursive=1',

[GitHubId, TJSONObject(enum.Value).GetPath('commit.sha').AsString]));

Counter := Files.Count;

for Enum2 in TJSONObject(GetJSON(HttpResult)).Arrays['tree'] do

begin

if Files.IndexOf(TJSONObject(Enum2.Value).Strings['path']) <> -1 then

Dec(Counter);

end;

if Counter = 0 then

// something good

FListItemToAdd.SubItems.Add('DA')

else

// somethinf bad

FListItemToAdd.SubItems.Add('NYET');

Synchronize(@UpdateProgress);

end;

BranchesJSON.Free;

AppDB.Free;

HTTP.Free;

Table.Free;

Files.Free;

Synchronize(@SuccessfulExecute);

end;

end.

**Приложение 2. GroupFormUnit.pas**

unit GroupFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

fphttpclient,

AppDatabase, Utils;

type

{ TGroupForm }

TGroupForm = class(TForm)

// OK кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле имени группы

LabeledEdit1: TLabeledEdit;

// Поле айди репозитория группы

LabeledEdit2: TLabeledEdit;

// Нажатие на ОК кнопку

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

private

// Айди группы для режима реадктирования

EditId: Integer;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; GroupForEditing: Integer); overload;

// Изменить дизайн формы для редактирования группы

procedure MakeForEdit(GroupId: Integer);

end;

implementation

{$R \*.lfm}

{ TGroupForm }

procedure TGroupForm.Button1Click(Sender: TObject);

var

HTTP: TFPHttpClient;

HttpResult: String;

AppDB: TAppDatabase;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

//ShowMessage(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

if String(LabeledEdit1.Text).IsEmpty or String(LabeledEdit1.Text).Contains(' ') then

begin

MessageDlg('Имя группы не должно быть пустым или иметь пробелы', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

except

if HTTP.ResponseStatusCode = 404 then

begin

MessageDlg('Репозиторий не найден', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end

else

begin

MessageDlg('Ошибка получения данных о репозитории. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

try

if EditId = -1 then

AppDB.Exec('INSERT INTO groups (name, github\_id) VALUES (:name, :rep\_id);',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text)])

else

AppDB.Exec('UPDATE groups SET name=:name, github\_id=:rep\_id WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TGroupForm.Button2Click(Sender: TObject);

begin

end;

constructor TGroupForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TGroupForm.Create(TheOwner: TComponent; GroupForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(GroupForEditing);

end;

procedure TGroupForm.MakeForEdit(GroupId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', GroupId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Self.LabeledEdit2.Text := ResultTable[0].KeyData['github\_id'];

Self.Caption := 'Изменение группы';

Self.EditId := GroupId;

end;

end.

**Приложение 3. TaskFormUnit.pas**

unit TaskFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

CheckLst, ColorBox, ShellCtrls, ComCtrls, Buttons, AppDatabase, fpjson;

type

{ TTaskForm }

TTaskForm = class(TForm)

// ОК кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Кнопка добавления дочерней ноды имени файла

Button3: TBitBtn;

// Кнопка удаления ноды имени файла

Button4: TBitBtn;

Label1: TLabel;

// Имя таска

LabeledEdit1: TLabeledEdit;

Panel1: TPanel;

// Дерево файлов

TreeView1: TTreeView;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Нажатие на кнопку добавления ноды

procedure Button3Click(Sender: TObject);

// Нажатие на кнопку удаления ноды

procedure Button4Click(Sender: TObject);

// Выделение ноды

procedure TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

private

// Айди таска для режима редактирования

EditId: Integer;

// Создание массива путей для каждого файла

function TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

// Создание дерева из массива путей

function TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; TaskForEditing: Integer); overload;

// Изменить дизайн формы для режима редактирования

procedure MakeForEdit(TaskId: Integer);

end;

implementation

{$R \*.lfm}

{ TTaskForm }

procedure TTaskForm.Button3Click(Sender: TObject);

var

Filename: String;

begin

Filename := InputBox('Введите имя файла',

'Введите имя файла для проверки', '');

if not Filename.IsEmpty then

TreeView1.Items.AddChild(TreeView1.Selected, Filename);

end;

procedure TTaskForm.Button4Click(Sender: TObject);

begin

if TreeView1.Selected <> nil then

TreeView1.Items.Delete(TreeView1.Selected);

end;

procedure TTaskForm.TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if not Assigned(TreeView1.GetNodeAt(X, Y)) then

for i := 0 to TreeView1.Items.Count - 1 do

TreeView1.Items[i].Selected := false;

end;

procedure TTaskForm.Button1Click(Sender: TObject);

var

AppDB: TAppDatabase;

FilePath: String;

json: TJSONArray;

begin

if Length(LabeledEdit1.Text) <= 0 then

begin

MessageDlg('Введите имя задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

if TreeView1.Items.Count <= 0 then

begin

MessageDlg('Задайте структуру задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

json := TJSONArray.Create;

AppDB := TAppDatabase.Create;

for FilePath in TreeToPathsGenerator(TreeView1.Items) do

json.Add(FilePath.Substring(1));

//ShowMessage(json.AsJSON);

try

if EditId = -1 then

AppDB.Exec('INSERT INTO tasks (name, files) VALUES (:name, :files);', [par('name', LabeledEdit1.Text), par('files', json.AsJSON)] )

else

AppDB.Exec('UPDATE tasks SET name=:name, files=:files WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('files', json.AsJSON), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TTaskForm.Button2Click(Sender: TObject);

begin

end;

function TTaskForm.TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

var Paths: TStringList;

procedure TreeWalker(Node: TTreeNode; PathString: String);

var

i: Integer;

begin

PathString += '/'+Node.Text;

if Node.HasChildren then

for i:=0 to Node.Count-1 do

TreeWalker(Node.Items[i], PathString)

else

Paths.Add(PathString);

end;

var

Node: TTreeNode;

begin

Paths := TStringList.Create;

for Node in Tree do

begin

if Node.Parent = nil then

TreeWalker(Node, '');

end;

Result := Paths;

end;

function TTaskForm.TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

var

Path: String;

PathParsed: TStringArray;

i, j: Integer;

Node, ChildNode: TTreeNode;

begin

Result := TreeView1.Items;

for Path in Paths do

begin

PathParsed := Path.Split(['/'], TStringSplitOptions.ExcludeEmpty);

Node := Result.FindNodeWithText(PathParsed[0]);

if (Node = nil) or (Node.Parent <> nil) then

Node := Result.AddChild(nil, PathParsed[0]);

for i := 1 to Length(PathParsed)-1 do

begin

ChildNode := Node.FindNode(PathParsed[i]);

if ChildNode = nil then

ChildNode := Result.AddChild(Node, PathParsed[i]);

Node := ChildNode;

end;

end;

end;

constructor TTaskForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TTaskForm.Create(TheOwner: TComponent; TaskForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(TaskForEditing);

end;

procedure TTaskForm.MakeForEdit(TaskId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

Paths: TStringList;

Enum: TJSONEnum;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', TaskId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Paths := TStringList.Create;

for Enum in TJSONArray(GetJSON(ResultTable[0].KeyData['files'])) do

Paths.Add(Enum.Value.AsString);

Self.TreeFromPathsGenerator(Paths);

Self.Caption := 'Изменение задания';

Self.EditId := TaskId;

end;

end.

**Приложение 4. AppDatabase.pas**

unit AppDatabase;

interface

uses

fpjson, sqldb, db, SQLite3Conn, sysutils, Generics.Collections, fgl, Variants, SQLite3;

type

// SQL параметр вы виде пары имя-значение

TParamPair = specialize TPair<String, Variant, String>;

// Динамический массив параметров

TParamsArray = Array of TParamPair;

// Строка таблицы

TRow = specialize TFPGMap<String, Variant>;

// Таблица

TTable = specialize TList<TRow>;

// Класс базы данных приложения

TAppDatabase = class (TInterfacedObject)

// Объект соединения с базой данных

Connection: TSQLite3Connection;

// Объект транзакции

Transaction: TSQLTransaction;

// Объект запроса

Query: TSQLQuery;

// Выполнить SQL запрос с параметрами без возврата результата

procedure Exec(AQuery: String; AParams: TParamsArray);

// Выполнить SQL запрос без параметров и без возврата результата

procedure Exec(AQuery: String);

// Выполнить SQL запрос с параметрами и с возвратом результата

function ExecOut(AQuery: String; AParams: TParamsArray): TTable;

// Выполнить SQL запрос без параметров с возвратом результата

function ExecOut(AQuery: String): TTable;

constructor Create;

destructor Free;

end;

// Функция для упрощенного создания объекта TParamPair

function par(AName: String; AValue: Variant): TParamPair;

implementation

// ===== TDatabase =====

procedure TAppDatabase.Exec(AQuery: String; AParams: TParamsArray);

var

Param: TParamPair;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.ExecSQL;

finally

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

procedure TAppDatabase.Exec(AQuery: String);

begin

Exec(AQuery, nil);

end;

function TAppDatabase.ExecOut(AQuery: String; AParams: TParamsArray): TTable;

var

Param: TParamPair;

Row: TRow;

Field: TField;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.Open;

Result := TTable.Create;

while not Query.EOF do

begin

Row := TRow.Create;

for Field in Query.Fields do

Row.Add(Field.FieldName, Field.Value);

Result.Add(Row);

Query.Next;

end;

finally

Query.Close;

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

function TAppDatabase.ExecOut(AQuery: String): TTable;

begin

Result := ExecOut(AQuery, nil);

end;

constructor TAppDatabase.Create;

begin

//if not FileExists('./application.db') then

//begin

// FileCreate('./application.db');

//end;

Connection := TSQLite3Connection.Create(nil);

Transaction := TSQLTransaction.Create(nil);

Query := TSQLQuery.Create(nil);

Connection.DatabaseName := './application.db';

Connection.Transaction := Transaction;

Query.DataBase := Connection;

Connection.Open;

Exec('CREATE TABLE IF NOT EXISTS "groups" ("name" TEXT NOT NULL UNIQUE,"github\_id" TEXT NOT NULL UNIQUE,"id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE)');

Exec('CREATE TABLE IF NOT EXISTS "tasks" ("id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,"name" TEXT NOT NULL,"files" TEXT NOT NULL)');

end;

destructor TAppDatabase.Free;

begin

Transaction.EndTransaction;

Connection.Connected := False;

Connection.Free;

Transaction.Free;

Query.Free;

end;

function par(AName: String; AValue: Variant): TParamPair;

begin

Result.Key := AName;

Result.Value := AValue;

end;

end.

**Приложение 5. SettingFormUnit.pas**

unit SettingsFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls;

type

{ TSettingsForm }

TSettingsForm = class(TForm)

// Кнопка сохранения

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле с именем пользователя GitHub

LabeledEdit1: TLabeledEdit;

// Поле с токеном пользователя GitHub

LabeledEdit2: TLabeledEdit;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Создания формы

procedure FormCreate(Sender: TObject);

private

// Сохранение настроек в конфиг

procedure LoadConfig;

// Загрузка настроек из конфига

procedure SaveConfig;

public

end;

implementation

{$R \*.lfm}

uses

jsonConf;

{ TSettingsForm }

procedure TSettingsForm.Button2Click(Sender: TObject);

begin

Self.Close;

end;

procedure TSettingsForm.FormCreate(Sender: TObject);

begin

LoadConfig;

end;

procedure TSettingsForm.Button1Click(Sender: TObject);

begin

SaveConfig;

Self.Close;

end;

procedure TSettingsForm.LoadConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

LabeledEdit1.Text := c.GetValue('/github/login', '');

LabeledEdit2.Text := c.GetValue('/github/token', '');

c.Free;

end;

procedure TSettingsForm.SaveConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

c.SetValue('/github/login', LabeledEdit1.Text);

c.SetValue('/github/token', LabeledEdit2.Text);

c.Free;

end;

end.

**Приложение 6. Utils.pas**

unit Utils;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Dialogs, fpjson, fphttpclient, Forms;

type

// Класс инкапсуляции объекта Integer

TIntObj = class

private

FI: Integer;

public

property I: Integer Read FI;

constructor Create(IValue: Integer);

end;

// Создание и подготовка HttpClient для работы с GitHub API

function CreateGitHubHTTP: TFPHttpClient;

// Функция быстрого открытия модального окна

procedure OpenDialog(FormClass: TFormClass);

implementation

uses

opensslsockets, jsonConf;

{ TIntObj }

constructor TIntObj.Create(IValue: Integer);

begin

Inherited Create;

FI := IValue;

end;

function CreateGitHubHTTP: TFPHttpClient;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

Result := TFPHTTPClient.Create(nil);

c.Filename := Application.Location+'/config.json';

Result.AllowRedirect := True;

Result.AddHeader('User-Agent', 'Mozilla/5.0 (compatible; fpweb)');

Result.AddHeader('Accept', 'application/vnd.github.v3+json');

Result.UserName := c.GetValue('/github/login', '');

Result.Password := c.GetValue('/github/token', '');

end;

procedure OpenDialog(FormClass: TFormClass);

var

ModalForm: TForm;

begin

ModalForm := FormClass.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

end.