Министерство цифрового развития, связи и массовых коммуникаций РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский Государственный Университет Телекоммуникаций и

Информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики (ПмиК)

09.03.01Программное обеспечение средств вычислительной техники и автоматизированных систем

(очная форма обучения)

«Приложение-симулятор логических схем»

ОТЧЕТ ПО РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЕ

По дисциплине «Визуальное программирование и человеко-машинное взаимодействие»

Выполнил:

студент гр. ИП-114 И.Н. Штапов

Проверил:

Ассистент каф. ПМиК И.А. Меркулов

Новосибирск 2023 г.

Оглавление

[Задание: 3](#_Toc135216558)

[Создание Use-Case диаграммы приложения: 4](#_Toc135216559)

[Разработка графического интерфейса: 6](#_Toc135216560)

[Меню управления: 7](#_Toc135216561)

[Краткое руководство для использования редактора схем: 8](#_Toc135216562)

[Листинг 10](#_Toc135216563)

# **Задание:**

Реализовать приложение-симулятор логических схем.

Работа состоит из следующих этапов:

1. Создание Use-Case диаграммы приложения. По окончании этапа должны быть построены Use-

Case диаграммы.

2. Разработка графического интерфейса (схематичное изображение интерфейса и описание

возможностей элементов, достижения сценариев описанных в Use-Case диаграмме посредством

этих элементов). По окончании этапа должна быть построена схема интерфейса с подробным

описанием элементов и достижения сценариев из use-case диаграммы.

3. Проектирование приложения - создание ER-диаграмм, диаграмм классов. По окончании этапа

должны быть построены диаграммы классов с описанием (обязательно), ER-диаграммы

(необязательно).

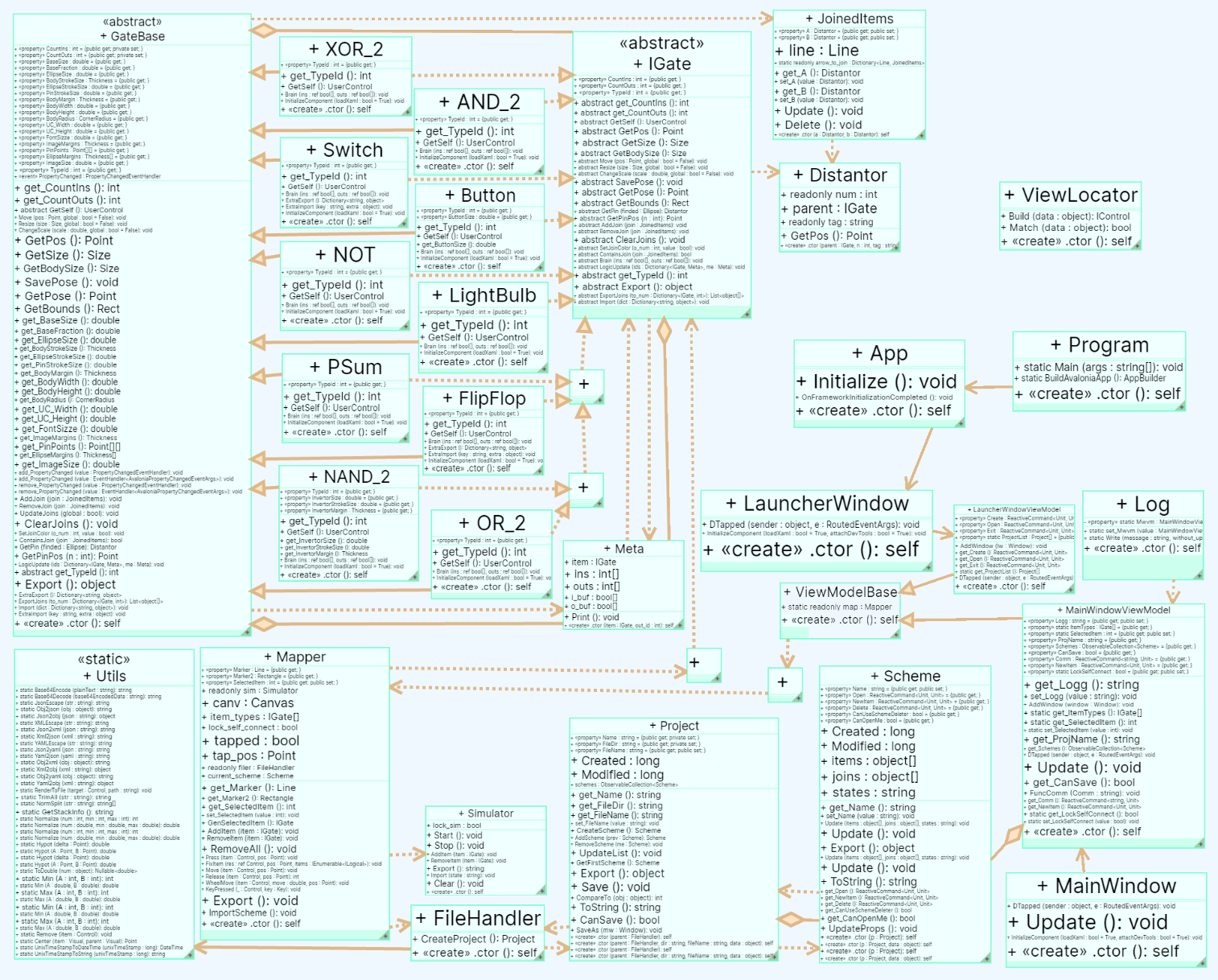
4. Разработка. При разработке используется TDD и упрощённый git flow (одна функциональность -

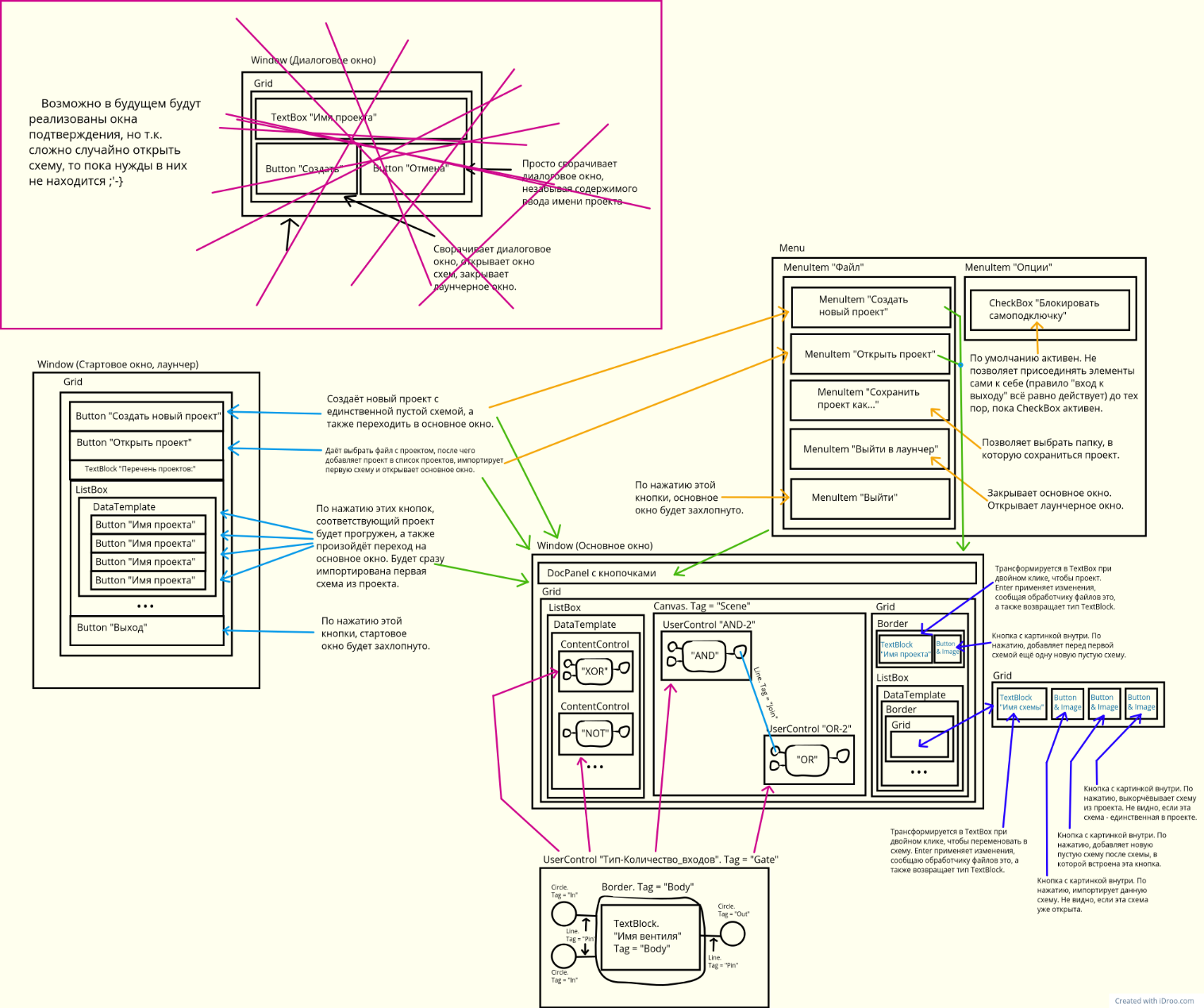
одна ветка, коммиты в логических точках).

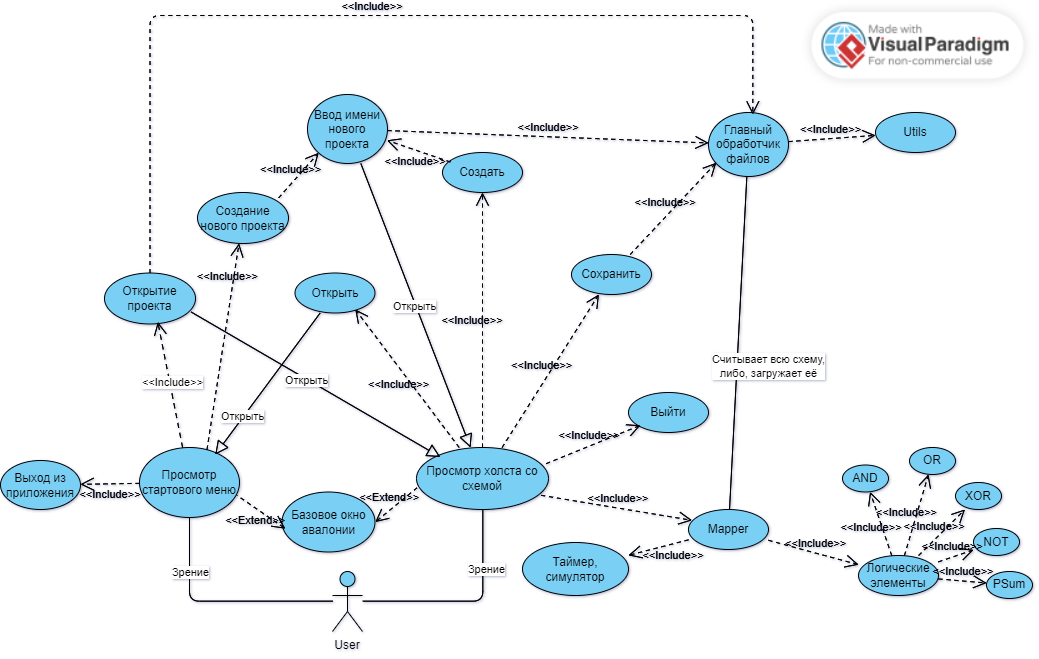
В репозитории приложения должен находиться отчёт по первым трём пунктам и проекты с исходным кодом и юнит-тестами.

Мой вариант: формат хранения проекта – SQLite, формат хранения списка открывавшихся проектов – YAML, дополнительные логические элементы – Декодер

# **Создание Use-Case диаграммы приложения:**







# **Разработка графического интерфейса:**

Было создано основное окно приложения, содержащее список проектов и три кнопки, с помощью которых вы можете создать или открыть проект, а также выйти из приложения.

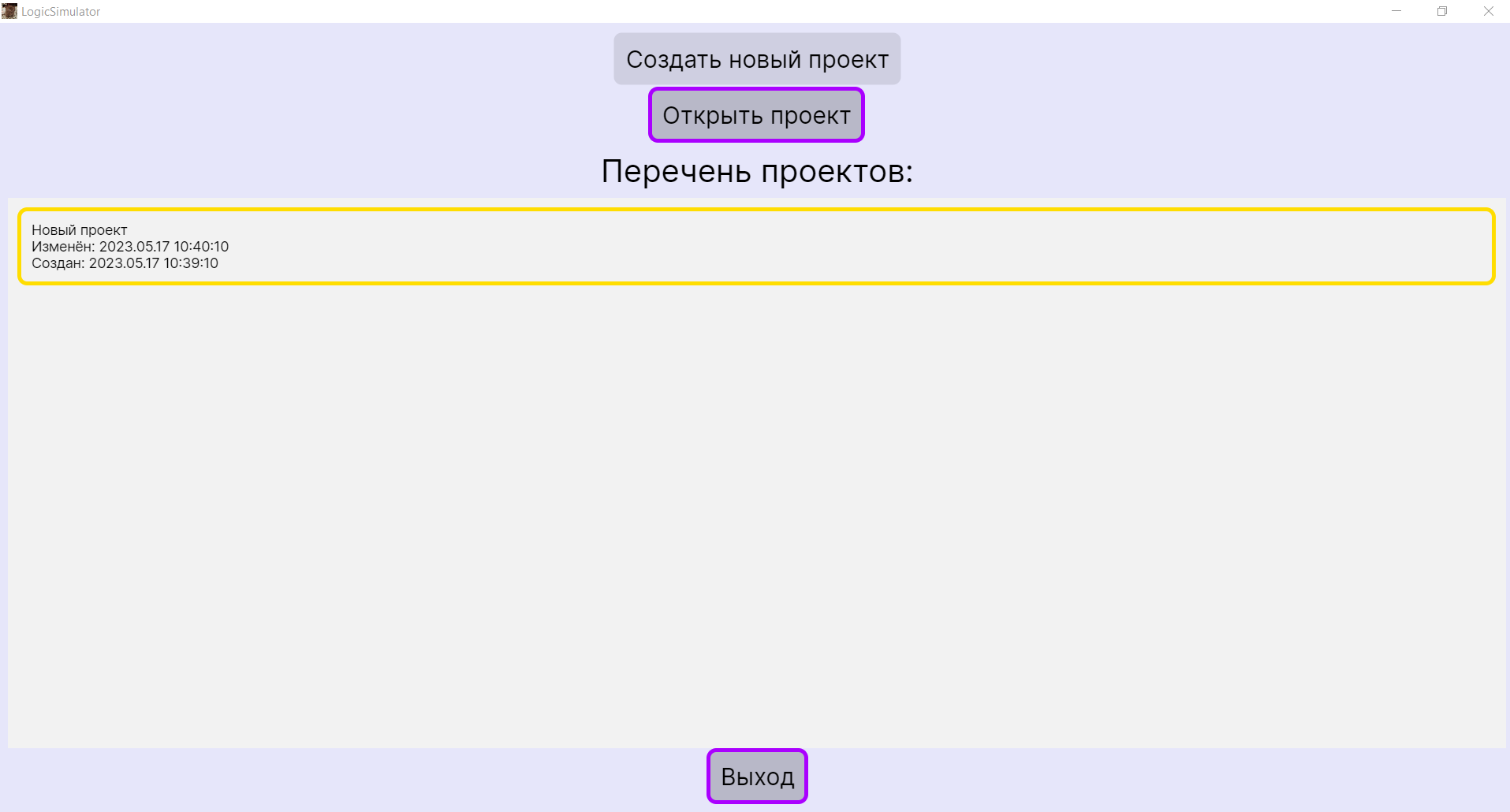


Рис. 1 – стартовое окно

При нажатии кнопки «Создать новый проект» открывается окно на котором размещены логические элементы, которые можно добавлять на схему. Также в верхнем левом углу расположено меню управления, имеющее 2 вкладки: файл и опции.

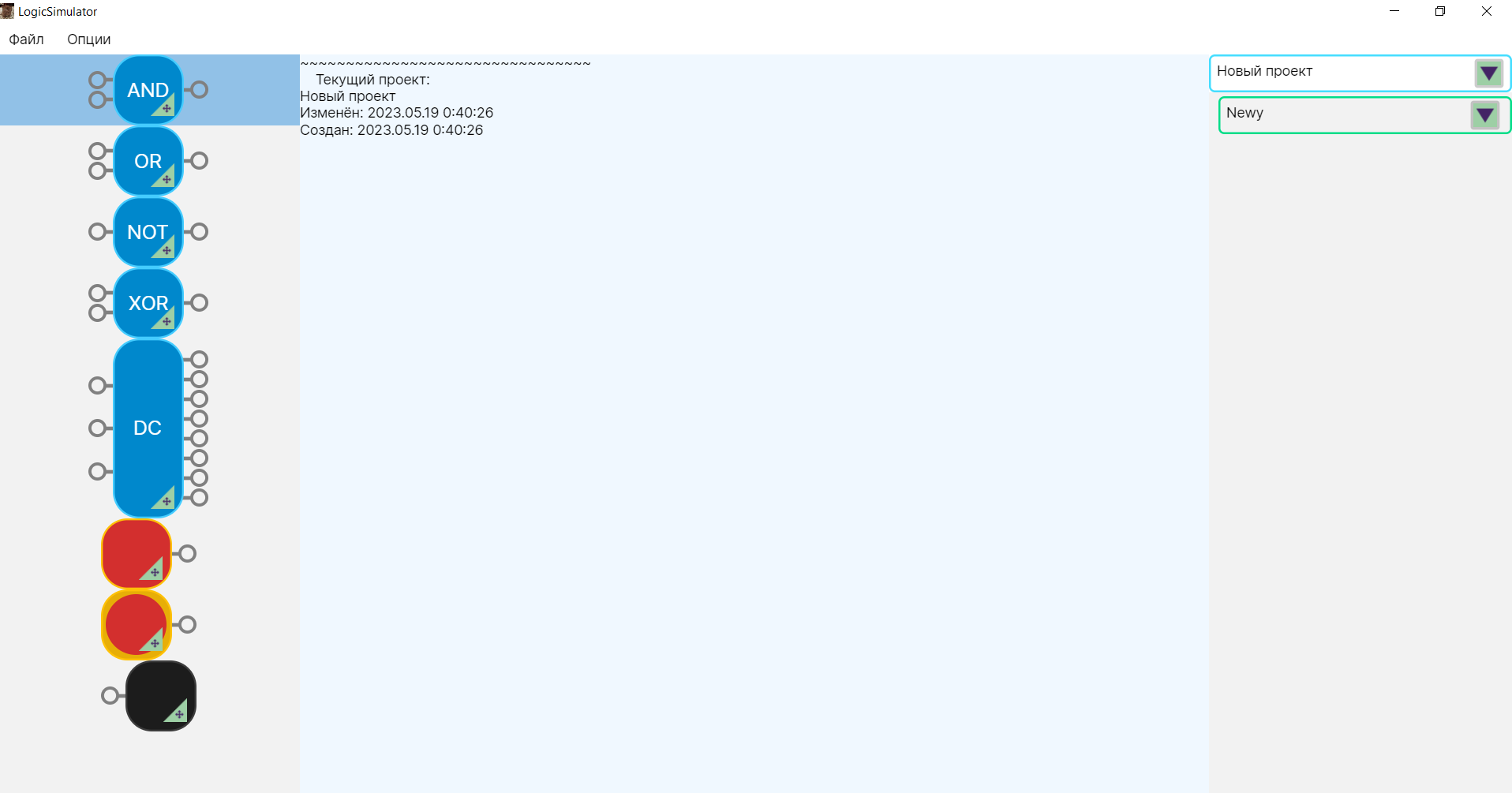


Рис. 2 – Окно создания проекта

# **Меню управления:**

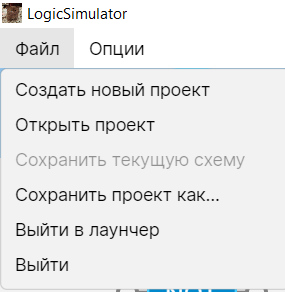
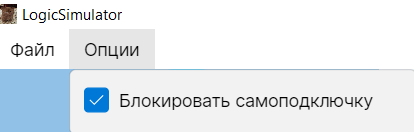
 

Рис. 3 и 4 – меню вкладок

Вкладка «Файл» включает 6 пунктов: "Создать новый проект", "Открыть проект", "Сохранить текущую схему", "Сохранить проект как", "Выйти в лаунчер" и "Выйти".

Пункт "Создать новый проект" - создаёт новый проект.

Пункт "Открыть проект" - открывает диалоговое окно открытия файла проекта, при выборе файла проекта, проект подгружается в приложение - его схемы отображаются в списке схем проекта, на холсте появляется отображение первой схемы в проекте.

Пункт "Сохранить текущую схему" – сохраняет схему в файл, в котором вы находитесь.

Пункт "Сохранить проект как" - открывает диалоговое окно сохранения проекта в файл. При выборе файла содержимое проекта сохраняется в него.

Пункт "Выйти в лаунчер" – выходит в главное меню.

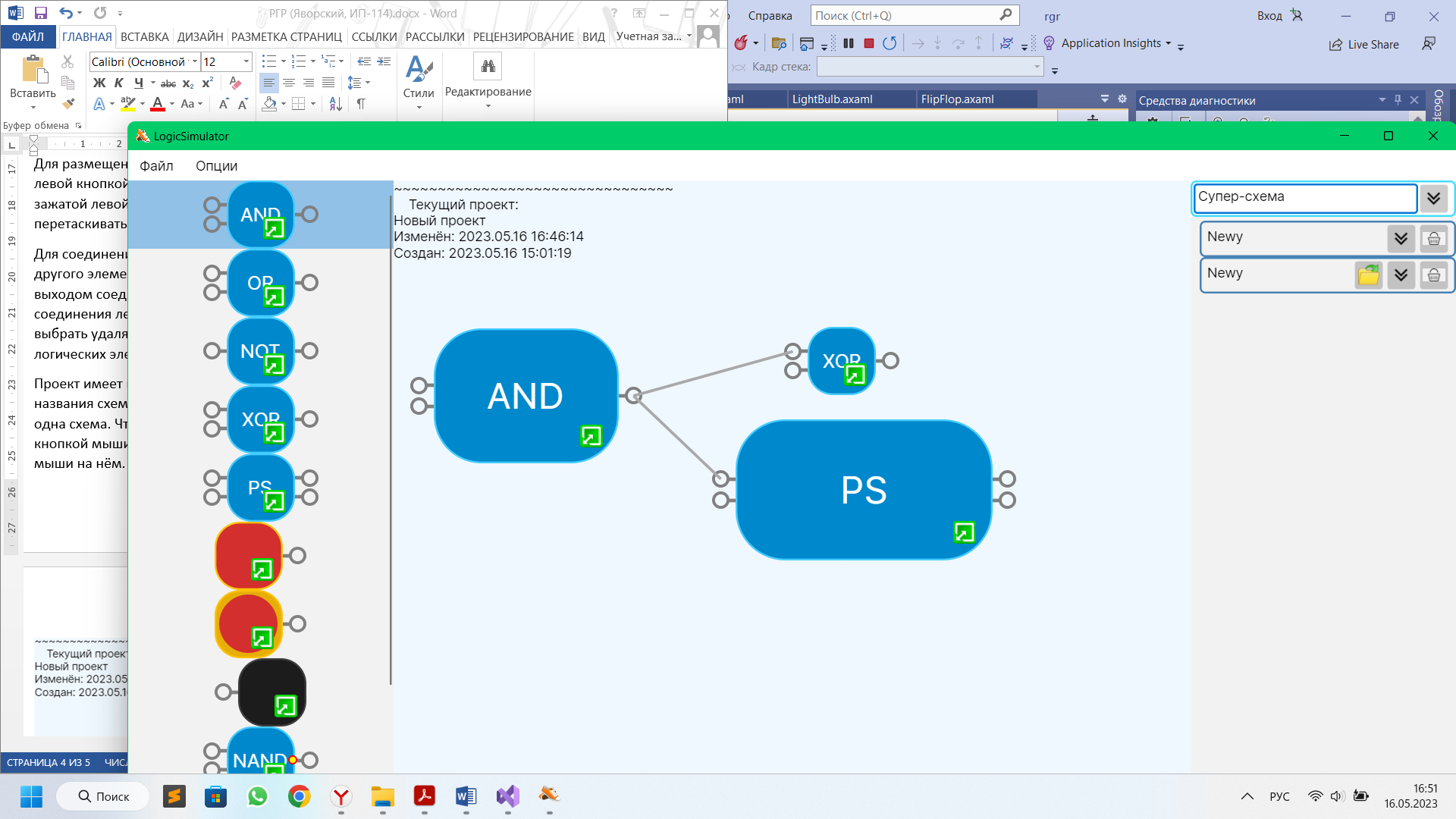
Пункт "Выйти" - закрывает приложение.

# **Краткое руководство для использования редактора схем:**

Для размещения элемента на схеме, нужно выбрать логический элемент на панели и кликнуть левой кнопкой мыши на холст в то место на которое нужно поместить логический элемент. При зажатой левой кнопкой мыши на логическом элементе, расположенном на холсте, его можно перетаскивать.

Для соединения логических элементов необходимо перетащить выход одного элемента на вход другого элемента или наоборот. При этом после соединения появляется линия между входом и выходом соединённых элементов. Для удаления соединения необходимо выбрать линию соединения левой кнопкой мыши и нажать клавишу delete. Для удаления элемента, необходимо выбрать удаляемый элемент левой кнопкой мыши и нажать клавишу delete. Выходные сигналы логических элементов должны рассчитываться в реальном времени.

Проект имеет вид списка с верхним элементом - названием проекта, все остальные элементы названия схем. Схемы можно добавлять и удалять, но в проекте всегда должна быть минимум одна схема. Чтобы отредактировать схему нужно кликнуть на неё в списке два раза левой кнопкой мыши. Название проекта можно отредактировать, кликнув два раза левой кнопкой мыши по нему.



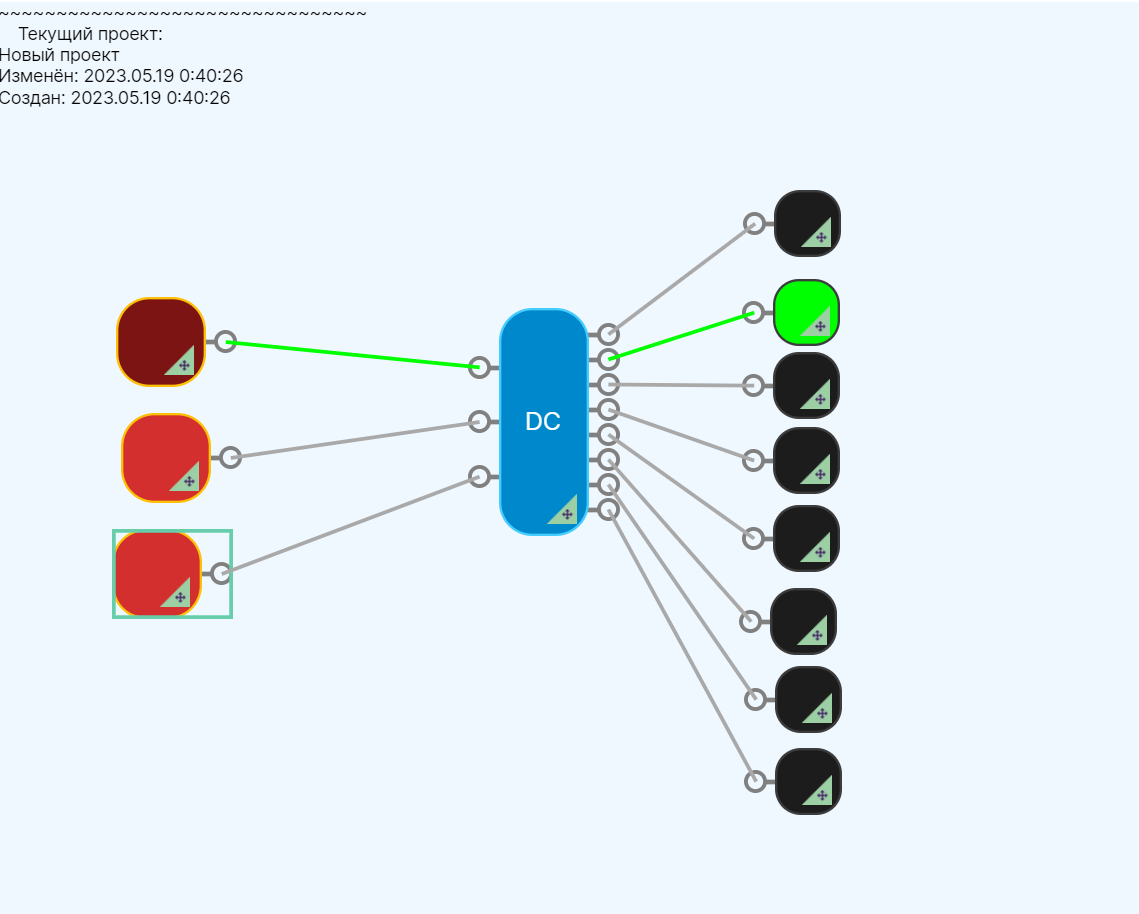


Рис. 5 – Вид схемы в сборке (пример)

Приложение поддерживает следующие логические элементы: И, ИЛИ, НЕ, ИСКЛ-ИЛИ, а также

элементы ВХОД и ВЫХОД. ВХОД позволяет по клику на нему левой кнопкой мыши поменять

выходящее из него значение, ВЫХОД отображает значение сигнала, подающееся ему на вход.

# **Листинг**

**Файл LauncherWindowViewModel.cs**

using Avalonia.Controls.Presenters;

using Avalonia.Controls;

using ReactiveUI;

using System.Reactive;

using LogicSimulator.Views;

using LogicSimulator.Models;

namespace LogicSimulator.ViewModels {

public class LauncherWindowViewModel: ViewModelBase {

Window? me;

private static readonly MainWindow mw = new();

public LauncherWindowViewModel() {

Create = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncCreate(); return new Unit(); });

Open = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncOpen(); return new Unit(); });

Exit = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncExit(); return new Unit(); });

}

public void AddWindow(Window lw) => me = lw;

void FuncCreate() {

var newy = map.filer.CreateProject();

CurrentProj = newy;

mw.Show();

mw.Update();

me?.Close();

}

void FuncOpen() {

if (me == null) return;

var selected = map.filer.SelectProjectFile(me);

if (selected == null) return;

CurrentProj = selected;

mw.Show();

mw.Update();

me?.Close();

}

void FuncExit() {

me?.Close();

mw.Close();

}

public ReactiveCommand<Unit, Unit> Create { get; }

public ReactiveCommand<Unit, Unit> Open { get; }

public ReactiveCommand<Unit, Unit> Exit { get; }

public static Project[] ProjectList { get => map.filer.GetSortedProjects(); }

public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {

var src = (Control?) e.Source;

if (src is ContentPresenter cp && cp.Child is Border bord) src = bord;

if (src is Border bord2 && bord2.Child is TextBlock tb2) src = tb2;

if (src is not TextBlock tb || tb.Tag is not Project proj) return;

CurrentProj = proj;

mw.Show();

mw.Update();

me?.Close();

}

/\*

\* Для тестирования

\*/

public static MainWindow GetMW => mw;

**Файл MainWindowViewModel.cs**

using Avalonia.Controls;

using Avalonia.Controls.Presenters;

using Avalonia.Input;

using LogicSimulator.Models;

using LogicSimulator.Views;

using LogicSimulator.Views.Shapes;

using ReactiveUI;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Reactive;

namespace LogicSimulator.ViewModels {

public class Log {

static readonly List<string> logs = new();

static readonly string path = "../../../Log.txt";

static bool first = true;

static readonly bool use\_file = false;

public static MainWindowViewModel? Mwvm { private get; set; }

public static void Write(string message, bool without\_update = false) {

if (!without\_update) {

foreach (var mess in message.Split('\n')) logs.Add(mess);

while (logs.Count > 45) logs.RemoveAt(0);

if (Mwvm != null) Mwvm.Logg = string.Join('\n', logs);

}

if (use\_file) {

if (first) File.WriteAllText(path, message + "\n");

else File.AppendAllText(path, message + "\n");

first = false;

}

}

}

public class MainWindowViewModel: ViewModelBase {

private string log = "";

public string Logg { get => log; set => this.RaiseAndSetIfChanged(ref log, value); }

public MainWindowViewModel() {

Log.Mwvm = this;

Comm = ReactiveCommand.Create<string, Unit>(n => { FuncComm(n); return new Unit(); });

NewItem = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncNewItem(); return new Unit(); });

}

private Window? mw;

public void AddWindow(Window window) {

var canv = window.Find<Canvas>("Canvas");

mw = window;

map.canv = canv;

if (canv == null) return;

canv.Children.Add(map.Marker);

canv.Children.Add(map.Marker2);

var panel = (Panel?) canv.Parent;

if (panel == null) return;

panel.PointerPressed += (object? sender, PointerPressedEventArgs e) => {

if (e.Source != null && e.Source is Control @control) map.Press(@control, e.GetCurrentPoint(canv).Position);

};

panel.PointerMoved += (object? sender, PointerEventArgs e) => {

if (e.Source != null && e.Source is Control @control) map.Move(@control, e.GetCurrentPoint(canv).Position);

};

panel.PointerReleased += (object? sender, PointerReleasedEventArgs e) => {

if (e.Source != null && e.Source is Control @control) {

int mode = map.Release(@control, e.GetCurrentPoint(canv).Position);

bool tap = map.tapped;

if (tap && mode == 1) {

var pos = map.tap\_pos;

if (canv == null) return;

var newy = map.GenSelectedItem();

newy.Move(pos);

map.AddItem(newy);

}

}

};

panel.PointerWheelChanged += (object? sender, PointerWheelEventArgs e) => {

if (e.Source != null && e.Source is Control @control) map.WheelMove(@control, e.Delta.Y, e.GetCurrentPoint(canv).Position);

};

mw.KeyDown += (object? sender, KeyEventArgs e) => {

if (e.Source != null && e.Source is Control @control) map.KeyPressed(@control, e.Key);

};

}

public static IGate[] ItemTypes { get => map.item\_types; }

public static int SelectedItem { get => map.SelectedItem; set => map.SelectedItem = value; }

/\*

\* Обработка панели со схемами проекта

\*/

Grid? cur\_grid;

TextBlock? old\_b\_child;

object? old\_b\_child\_tag;

string? prev\_scheme\_name;

public static string ProjName { get => CurrentProj == null ? "???" : CurrentProj.Name; }

public static ObservableCollection<Scheme> Schemes { get => CurrentProj == null ? new() : CurrentProj.schemes; }

public void DTapped(object? sender, Avalonia.Interactivity.RoutedEventArgs e) {

var src = (Control?) e.Source;

if (src is ContentPresenter cp && cp.Child is Border bord) src = bord;

if (src is Border bord2 && bord2.Child is Grid g2) src = g2;

if (src is Grid g3 && g3.Children[0] is TextBlock tb2) src = tb2;

if (src is not TextBlock tb) return;

var p = tb.Parent;

if (p == null) return;

if (old\_b\_child != null)

if (cur\_grid != null) cur\_grid.Children[0] = old\_b\_child;

if (p is not Grid g) return;

cur\_grid = g;

old\_b\_child = tb;

old\_b\_child\_tag = tb.Tag;

prev\_scheme\_name = tb.Text;

var newy = new TextBox { Text = tb.Text };

cur\_grid.Children[0] = newy;

newy.KeyUp += (object? sender, KeyEventArgs e) => {

if (e.Key != Key.Return) return;

if (newy.Text != prev\_scheme\_name) {

if ((string?) tb.Tag == "p\_name") CurrentProj?.ChangeName(newy.Text);

else if (old\_b\_child\_tag is Scheme scheme) scheme.ChangeName(newy.Text);

}

cur\_grid.Children[0] = tb;

cur\_grid = null; old\_b\_child = null;

};

}

public void Update() {

Log.Write("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n Текущий проект:\n" + CurrentProj);

map.ImportScheme();

this.RaisePropertyChanged(new(nameof(ProjName)));

this.RaisePropertyChanged(new(nameof(Schemes)));

this.RaisePropertyChanged(new(nameof(CanSave)));

if (mw != null) mw.Width++;

}

public static bool CanSave { get => CurrentProj != null && CurrentProj.CanSave(); }

/\*

\* Кнопки!

\*/

public void FuncComm(string Comm) {

switch (Comm) {

case "Create":

var newy = map.filer.CreateProject();

CurrentProj = newy;

Update();

break;

case "Open":

if (mw == null) break;

var selected = map.filer.SelectProjectFile(mw);

if (selected != null) {

CurrentProj = selected;

Update();

}

break;

case "Save":

map.Export();

File.WriteAllText("../../../for\_test.json", Utils.Obj2json((map.current\_scheme ?? throw new System.Exception("Чё?!")).Export()));

break;

case "SaveAs":

map.Export();

if (mw != null) CurrentProj?.SaveAs(mw);

this.RaisePropertyChanged(new(nameof(CanSave)));

break;

case "ExitToLauncher":

new LauncherWindow().Show();

mw?.Hide();

break;

case "Exit":

mw?.Close();

break;

}

}

public ReactiveCommand<string, Unit> Comm { get; }

private static void FuncNewItem() {

CurrentProj?.AddScheme(null);

}

public ReactiveCommand<Unit, Unit> NewItem { get; }

public static bool LockSelfConnect { get => map.lock\_self\_connect; set => map.lock\_self\_connect = value; }

}

}

**Файл ViewModelBase.cs**

using LogicSimulator.Models;

using ReactiveUI;

namespace LogicSimulator.ViewModels {

public class ViewModelBase: ReactiveObject {

public readonly static Mapper map = new();

private static Project? current\_proj;

protected static Project? CurrentProj {

get => current\_proj;

set {

if (value == null) return;

current\_proj = value;

map.current\_scheme = value.GetFirstScheme();

}

}

/\*

\* Для тестирования

\*/

public static Project? TopSecretGetProj() => current\_proj;

}

}

**Файл Distantor.cs**

using Avalonia;

using LogicSimulator.Views.Shapes;

namespace LogicSimulator.Models {

public class Distantor {

public readonly int num;

public IGate parent;

public readonly string tag;

public Distantor(IGate parent, int n, string tag) {

this.parent = parent;

num = n;

this.tag = tag;

}

public Point GetPos() => parent.GetPinPos(num);

}

}

**Файл FileHandler.cs**

using Avalonia.Controls;

using LogicSimulator.ViewModels;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

namespace LogicSimulator.Models {

public class FileHandler {

readonly string AppData;

readonly List<Project> projects = new();

readonly List<string> project\_paths = new();

readonly Dictionary<string, Project> proj\_dict = new();

public FileHandler() {

string app\_data = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

app\_data = Path.Combine(app\_data, "LogicSimulator");

if (!Directory.Exists(app\_data)) Directory.CreateDirectory(app\_data);

AppData = app\_data;

LoadProjectList();

}

private void AddProject(Project proj) {

if (proj.FileDir == null || proj.FileName == null) return;

var path = Path.Combine(proj.FileDir, proj.FileName);

if (proj\_dict.ContainsKey(path)) return;

proj\_dict[path] = proj;

projects.Add(proj);

}

private static string GetProjectFileName(string dir) {

int n = 0;

while (true) {

string name = "proj\_" + ++n + ".db";

if (!File.Exists(Path.Combine(dir, name))) return name;

}

}

public Project CreateProject() {

var proj = new Project(this);

// projects.Add(proj); Перенесено в AppendProject

return proj;

}

private Project? LoadProject(string dir, string fileName) {

try {

var path = Path.Combine(dir, fileName);

if (!File.Exists(path)) return null;

var obj = Utils.SQLite\_proj2obj(path) ?? throw new DataException("Не верная структура SQLite (.db)-файла проекта!");

var proj = new Project(this, dir, fileName, obj);

AddProject(proj);

return proj;

} catch (Exception e) { Log.Write("Неудачная попытка загрузить проект:\n" + e); }

return null;

}

private Project? LoadProject(string path) {

var s\_arr = path.Split(Path.DirectorySeparatorChar).ToList();

var name = s\_arr[^1];

s\_arr.RemoveRange(s\_arr.Count - 1, 1);

var dir = Path.Combine(s\_arr.ToArray());

return LoadProject(dir, name);

}

private void LoadProjectList() {

var file = Path.Combine(AppData, "project\_list.yaml");

if (!File.Exists(file)) return;

object data;

try { data = Utils.Yaml2obj(File.ReadAllText(file)) ?? throw new DataException("Не верная структура YAML-файла списка проектов!"); } catch (Exception e) { Log.Write("Неудачная попытка загрузить список проектов:\n" + e); return; }

if (data is not List<object> @arr) { Log.Write("В списке проектов на верхнем уровне ожидалось увидеть список"); return; }

foreach (var path in @arr) {

if (path is not string @str) { Log.Write("Один из путей списка проектов - не строка: " + path); continue; }

project\_paths.Add(@str);

LoadProject(@str);

}

}

internal static void SaveProject(Project proj) {

var dir = proj.FileDir;

if (dir == null) return;

var name = proj.FileName;

name ??= GetProjectFileName(dir);

proj.FileName = name;

var path = Path.Combine(dir, name);

File.WriteAllBytes(path, Convert.FromBase64String("U1FMaXRlIGZvcm1hdCAzABAAAQEAQCAgAAAAAwAAAAQAAAAAAAAAAAAAAAMAAAAEAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADAC5jAA0AAAADDWwAD1IOQw1sAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIFUAwcXFxcBgwd0YWJsZWpvaW5zam9pbnMEQ1JFQVRFIFRBQkxFIGpvaW5zICgKICAgIG51bV9hIElOVEVHRVIgTk9UIE5VTEwsCiAgICBwaW5fYSBJTlRFR0VSIE5PVCBOVUxMLAogICAgdGFnX2EgVEVYVCAgICBOT1QgTlVMTCwKICAgIG51bV9iIElOVEVHRVIgTk9UIE5VTEwsCiAgICBwaW5fYiBJTlRFR0VSIE5PVCBOVUxMLAogICAgdGFnX2IgVEVYVCAgICBOT1QgTlVMTAopggwCBxcXFwGDd3RhYmxlaXRlbXNpdGVtcwNDUkVBVEUgVEFCTEUgaXRlbXMgKAogICAgaWQgICAgICAgIElOVEVHRVIgTk9UIE5VTEwsCiAgICBwb3NfeCAgICAgUkVBTCAgICBOT1QgTlVMTCwKICAgIHBvc195ICAgICBSRUFMICAgIE5PVCBOVUxMLAogICAgc2l6ZV94ICAgIFJFQUwgICAgTk9UIE5VTEwsCiAgICBzaXplX3kgICAgUkVBTCAgICBOT1QgTlVMTCwKICAgIGJhc2Vfc2l6ZSBSRUFMICAgIE5PVCBOVUxMLAogICAgZXh0cmEgICAgIFRFWFQgICAgTk9UIE5VTEwKKYErAQcXGRkBgjF0YWJsZWhlYWRlcmhlYWRlcgJDUkVBVEUgVEFCTEUgaGVhZGVyICgKICAgIG5hbWUgICAgIFRFWFQgICAgTk9UIE5VTEwsCiAgICBjcmVhdGVkICBJTlRFR0VSIE5PVCBOVUxMLAogICAgbW9kaWZpZWQgSU5URUdFUiBOT1QgTlVMTCwKICAgIHNjaGVtZXMgIElOVEVHRVIgTk9UIE5VTEwKKQ0AAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAANAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADQAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=="));

Utils.Obj2sqlite\_proj(proj, path);

}

private void SaveProjectList() {

var file = Path.Combine(AppData, "project\_list.yaml");

var content = Utils.Obj2yaml(project\_paths) ?? throw new Exception("Такого не бывает");

File.WriteAllText(file, content);

}

internal Project[] GetSortedProjects() {

projects.Sort();

return projects.ToArray();

}

internal void AppendProject(Project proj) {

if (proj.FileDir == null || proj.FileName == null) return;

var path = Path.Combine(proj.FileDir, proj.FileName);

if (project\_paths.Contains(path)) return;

project\_paths.Add(path);

AddProject(proj);

SaveProjectList();

}

internal static string? RequestProjectPath(Window parent) {

var dlg = new OpenFolderDialog {

Title = "Выберите папку, куда надо сохранить новый проект"

};

var task = dlg.ShowAsync(parent);

return task.GetAwaiter().GetResult();

}

internal Project? SelectProjectFile(Window parent) {

var dlg = new OpenFileDialog {

Title = "Выберите файл с проектом (proj\_\*.db), который нужно открыть"

};

dlg.Filters?.Add(new FileDialogFilter() { Name = "SQLite (.db) Files", Extensions = { "db" } });

dlg.Filters?.Add(new FileDialogFilter() { Name = "All Files", Extensions = { "\*" } });

dlg.AllowMultiple = false;

var task = dlg.ShowAsync(parent);

var res = task.GetAwaiter().GetResult();

if (res == null) return null;

var path = res[0];

if (!proj\_dict.TryGetValue(path, out var proj)) {

proj = LoadProject(path);

if (proj != null) AppendProject(proj);

}

return proj;

}

}

}

**Файл JoinedItems.cs**

using Avalonia.Controls.Shapes;

using Avalonia.Media;

using System.Collections.Generic;

namespace LogicSimulator.Models {

public class JoinedItems {

public static readonly Dictionary<Line, JoinedItems> arrow\_to\_join = new();

public JoinedItems(Distantor a, Distantor b) {

A = a; B = b; Update();

a.parent.AddJoin(this);

if (a.parent != b.parent) b.parent.AddJoin(this);

arrow\_to\_join[line] = this;

}

public Distantor A { get; set; }

public Distantor B { get; set; }

public Line line = new() { Tag = "Join", ZIndex = 2, Stroke = Brushes.DarkGray, StrokeThickness = 3 };

public void Update() {

line.StartPoint = A.GetPos();

line.EndPoint = B.GetPos();

}

public void Delete() {

arrow\_to\_join.Remove(line);

line.Remove();

A.parent.RemoveJoin(this);

B.parent.RemoveJoin(this);

}

}

}

**Файл Mapper.cs**

using Avalonia.Controls;

using Avalonia;

using LogicSimulator.ViewModels;

using LogicSimulator.Views.Shapes;

using System;

using System.Collections.Generic;

using DynamicData;

using Avalonia.Controls.Shapes;

using Avalonia.Media;

using Avalonia.LogicalTree;

using System.Linq;

using Button = LogicSimulator.Views.Shapes.Button;

using Avalonia.Input;

namespace LogicSimulator.Models {

public class Mapper {

readonly Line marker = new() { Tag = "Marker", ZIndex = 2, IsVisible = false, Stroke = Brushes.YellowGreen, StrokeThickness = 3 };

readonly Rectangle marker2 = new() { Tag = "Marker", Classes = new("anim"), ZIndex = 2, IsVisible = false, Stroke = Brushes.MediumAquamarine, StrokeThickness = 3 };

public Line Marker { get => marker; }

public Rectangle Marker2 { get => marker2; }

public readonly Simulator sim = new();

public Canvas canv = new();

/\*

\* Маркер

\*/

private IGate? marked\_item;

private JoinedItems? marked\_line;

private void UpdateMarker() {

marker2.IsVisible = marked\_item != null || marked\_line != null;

if (marked\_item != null) {

var bound = marked\_item.GetBounds();

marker2.Margin = new(bound.X, bound.Y);

marker2.Width = bound.Width;

marker2.Height = bound.Height;

marked\_line = null;

}

if (marked\_line != null) {

var line = marked\_line.line;

var A = line.StartPoint;

var B = line.EndPoint;

marker2.Margin = new(Math.Min(A.X, B.X), Math.Min(A.Y, B.Y));

marker2.Width = Math.Abs(A.X - B.X);

marker2.Height = Math.Abs(A.Y - B.Y);

}

}

/\*

\* Выборка элементов

\*/

private int selected\_item = 0;

public int SelectedItem { get => selected\_item; set => selected\_item = value; }

private static IGate CreateItem(int n) {

return n switch {

0 => new AND\_2(),

1 => new OR\_2(),

2 => new NOT(),

3 => new XOR\_2(),

4 => new DeCoder\_3(),

5 => new Switch(),

6 => new Button(),

7 => new LightBulb(),

\_ => null,

};

}

public IGate[] item\_types = Enumerable.Range(0, 12).Select(CreateItem).ToArray();

public IGate GenSelectedItem() => CreateItem(selected\_item);

/\*

\* Хранилище

\*/

readonly List<IGate> items = new();

private void AddToMap(IControl item) {

canv.Children.Add(item);

}

public void AddItem(IGate item) {

items.Add(item);

sim.AddItem(item);

AddToMap(item.GetSelf());

}

public void RemoveItem(IGate item) {

if (marked\_item != null) {

marked\_item = null;

UpdateMarker();

}

if (marked\_line != null && item.ContainsJoin(marked\_line)) {

marked\_line = null;

UpdateMarker();

}

items.Remove(item);

sim.RemoveItem(item);

item.ClearJoins();

((Control) item).Remove();

}

public void RemoveAll() {

foreach (var item in items.ToArray()) RemoveItem(item);

sim.Clear();

}

private void SaveAllPoses() {

foreach (var item in items) item.SavePose();

}

/\*

\* Определение режима перемещения

\*/

int mode = 0;

/\*

\* Режимы:

\* 0 - ничего не делает

\* 1 - двигаем камеру

\* 2 - двигаем элемент

\* 3 - тянем элемент

\* 4 - вышвыриваем элемент

\* 5 - тянем линию от входа (In)

\* 6 - тянем линию от выхода (Out)

\* 7 - тянем линию от узла (IO)

\* 8 - тянем уже существующее соединение - переподключаем

\*/

private static int CalcMode(string? tag) {

if (tag == null) return 0;

return tag switch {

"Scene" => 1,

"Body" => 2,

"Resizer" => 3,

"Deleter" => 4,

"In" => 5,

"Out" => 6,

"IO" => 7,

"Join" => 8,

"Pin" or \_ => 0,

};

}

private void UpdateMode(Control item) => mode = CalcMode((string?) item.Tag);

private static bool IsMode(Control item, string[] mods) {

var name = (string?) item.Tag;

if (name == null) return false;

return mods.IndexOf(name) != -1;

}

private static UserControl? GetUC(Control item) {

while (item.Parent != null) {

if (item is UserControl @UC) return @UC;

item = (Control) item.Parent;

}

return null;

}

private static IGate? GetGate(Control item) {

var UC = GetUC(item);

if (UC is IGate @gate) return @gate;

return null;

}

/\*

\* Обработка мыши

\*/

Point moved\_pos;

IGate? moved\_item;

Point item\_old\_pos;

Size item\_old\_size;

Ellipse? marker\_circle;

Distantor? start\_dist;

int marker\_mode;

Line? old\_join;

bool join\_start;

bool delete\_join = false;

public bool lock\_self\_connect = true;

public void Press(Control item, Point pos) {

UpdateMode(item);

moved\_pos = pos;

moved\_item = GetGate(item);

tapped = true;

if (moved\_item != null) item\_old\_pos = moved\_item.GetPos();

switch (mode) {

case 1:

SaveAllPoses();

break;

case 3:

if (moved\_item == null) break;

item\_old\_size = moved\_item.GetBodySize();

break;

case 5 or 6 or 7:

if (marker\_circle == null) break;

var gate = GetGate(marker\_circle) ?? throw new Exception("Чё?!");

start\_dist = gate.GetPin(marker\_circle);

var circle\_pos = start\_dist.GetPos();

marker.StartPoint = marker.EndPoint = circle\_pos;

marker.IsVisible = true;

marker\_mode = mode;

break;

case 8:

if (item is not Line @join) break;

JoinedItems.arrow\_to\_join.TryGetValue(@join, out var @join2);

if (@join2 == null) break;

if (marked\_line == @join2) {

marked\_line = null;

UpdateMarker();

}

var dist\_a = @join.StartPoint.Hypot(pos);

var dist\_b = @join.EndPoint.Hypot(pos);

join\_start = dist\_a > dist\_b;

old\_join = @join;

marker.StartPoint = join\_start ? @join.StartPoint : pos;

marker.EndPoint = join\_start ? pos : @join.EndPoint;

marker\_mode = CalcMode(join\_start ? @join2.A.tag : @join2.B.tag);

marker.IsVisible = true;

@join.IsVisible = false;

break;

}

Move(item, pos);

}

public void FixItem(ref Control res, Point pos, IEnumerable<ILogical> items) {

foreach (var logic in items) {

var item = (Control) logic;

var tb = item.TransformedBounds;

if (tb != null && tb.Value.Bounds.TransformToAABB(tb.Value.Transform).Contains(pos) && (string?) item.Tag != "Join") res = item; // Гениально! Апгрейд прошёл успешно :D

FixItem(ref res, pos, item.GetLogicalChildren());

}

}

public void Move(Control item, Point pos, bool use\_fix = true) {

if (use\_fix && (mode == 5 || mode == 6 || mode == 7 || mode == 8)) {

var tb = canv.TransformedBounds;

if (tb != null) {

item = new Canvas() { Tag = "Scene" };

var bounds = tb.Value.Bounds.TransformToAABB(tb.Value.Transform);

FixItem(ref item, pos + bounds.TopLeft, canv.Children);

}

}

string[] mods = new[] { "In", "Out", "IO" };

var tag = (string?) item.Tag;

if (IsMode(item, mods) && item is Ellipse @ellipse

&& !(marker\_mode == 5 && tag == "In" || marker\_mode == 6 && tag == "Out" ||

lock\_self\_connect && moved\_item == GetGate(item))) {

if (marker\_circle != null && marker\_circle != @ellipse) {

marker\_circle.Fill = new SolidColorBrush(Color.Parse("#0000"));

marker\_circle.Stroke = Brushes.Gray;

}

marker\_circle = @ellipse;

@ellipse.Fill = Brushes.Lime;

@ellipse.Stroke = Brushes.Green;

} else if (marker\_circle != null) {

marker\_circle.Fill = new SolidColorBrush(Color.Parse("#0000"));

marker\_circle.Stroke = Brushes.Gray;

marker\_circle = null;

}

if (mode == 8) delete\_join = (string?) item.Tag == "Deleter";

var delta = pos - moved\_pos;

if (delta.X == 0 && delta.Y == 0) return;

if (Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2) > 9) tapped = false;

switch (mode) {

case 1:

foreach (var item\_ in items) {

var pose = item\_.GetPose();

item\_.Move(pose + delta, true);

}

UpdateMarker();

break;

case 2:

if (moved\_item == null) break;

var new\_pos = item\_old\_pos + delta;

moved\_item.Move(new\_pos);

UpdateMarker();

break;

case 3:

if (moved\_item == null) break;

var new\_size = item\_old\_size + new Size(delta.X, delta.Y);

moved\_item.Resize(new\_size);

UpdateMarker();

break;

case 5 or 6 or 7:

var end\_pos = marker\_circle == null ? pos : marker\_circle.Center(canv);

marker.EndPoint = end\_pos;

break;

case 8:

if (old\_join == null) break;

var p = marker\_circle == null ? pos : marker\_circle.Center(canv);

if (join\_start) marker.EndPoint = p;

else marker.StartPoint = p;

break;

}

}

public bool tapped = false;

public Point tap\_pos;

public int Release(Control item, Point pos, bool use\_fix = true) {

Move(item, pos, use\_fix);

switch (mode) {

case 5 or 6 or 7:

if (start\_dist == null) break;

if (marker\_circle != null) {

var gate = GetGate(marker\_circle) ?? throw new Exception("Чё?!");

var end\_dist = gate.GetPin(marker\_circle);

var newy = new JoinedItems(start\_dist, end\_dist);

AddToMap(newy.line);

}

marker.IsVisible = false;

marker\_mode = 0;

break;

case 8:

if (old\_join == null) break;

JoinedItems.arrow\_to\_join.TryGetValue(old\_join, out var @join);

if (marker\_circle != null && @join != null) {

var gate = GetGate(marker\_circle) ?? throw new Exception("Чё?!");

var p = gate.GetPin(marker\_circle);

@join.Delete();

var newy = join\_start ? new JoinedItems(@join.A, p) : new JoinedItems(p, @join.B);

AddToMap(newy.line);

} else old\_join.IsVisible = true;

marker.IsVisible = false;

marker\_mode = 0;

old\_join = null;

if (delete\_join) @join?.Delete();

delete\_join = false;

break;

}

if (tapped) Tapped(item, pos);

int res\_mode = mode;

mode = 0;

moved\_item = null;

return res\_mode;

}

private void Tapped(Control item, Point pos) {

tap\_pos = pos;

switch (mode) {

case 2 or 8:

if (item is Line @line) {

if (!JoinedItems.arrow\_to\_join.TryGetValue(@line, out var @join)) break;

marked\_item = null;

marked\_line = @join;

UpdateMarker();

break;

}

if (moved\_item == null) break;

marked\_item = moved\_item;

UpdateMarker();

break;

}

}

public void WheelMove(Control item, double move, Point pos) {

int mode = CalcMode((string?) item.Tag);

double scale = move > 0 ? 1.1 : 1 / 1.1;

double inv\_scale = 1 / scale;

switch (mode) {

case 1:

foreach (var gate in items) {

gate.ChangeScale(scale, true);

var item\_pos = gate.GetPos();

var delta = item\_pos - pos;

delta \*= scale;

var new\_pos = delta + pos;

gate.Move(new\_pos, false);

}

UpdateMarker();

break;

case 2:

var gate2 = GetGate(item);

if (gate2 == null) return;

gate2.ChangeScale(inv\_scale);

UpdateMarker();

break;

}

}

public void KeyPressed(Control \_, Key key) {

switch (key) {

case Key.Up:

case Key.Left:

case Key.Right:

case Key.Down:

int dx = key == Key.Left ? -1 : key == Key.Right ? 1 : 0;

int dy = key == Key.Up ? -1 : key == Key.Down ? 1 : 0;

marked\_item?.Move(marked\_item.GetPos() + new Point(dx \* 10, dy \* 10));

UpdateMarker();

break;

case Key.Delete:

if (marked\_item != null) RemoveItem(marked\_item);

if (marked\_line != null) {

marked\_line.Delete();

marked\_line = null;

UpdateMarker();

}

break;

}

}

/\*

\* Экспорт и импорт

\*/

public readonly FileHandler filer = new();

public Scheme? current\_scheme;

public void Export() {

if (current\_scheme == null) return;

var arr = items.Select(x => x.Export()).ToArray();

Dictionary<IGate, int> item\_to\_num = new();

int n = 0;

foreach (var item in items) item\_to\_num.Add(item, n++);

List<object[]> joins = new();

foreach (var item in items) joins.Add(item.ExportJoins(item\_to\_num));

sim.Clean();

string states = sim.Export();

try { current\_scheme.Update(arr, joins.ToArray(), states); }

catch (Exception e) { Log.Write("Save error:\n" + e); }

}

public void ImportScheme(bool start = true) {

if (current\_scheme == null) return;

sim.Stop();

sim.lock\_sim = true;

RemoveAll();

List<IGate> list = new();

foreach (var item in current\_scheme.items) {

if (item is not Dictionary<string, object> @dict) { Log.Write("Не верный тип элемента: " + item); continue; }

if (!@dict.TryGetValue("id", out var @value)) { Log.Write("id элемента не обнаружен"); continue; }

if (@value is not int @id) { Log.Write("Неверный тип id: " + @value); continue; }

var newy = CreateItem(@id);

newy.Import(@dict);

AddItem(newy);

list.Add(newy);

}

var items\_arr = list.ToArray();

List<JoinedItems> joinz = new();

foreach (var obj in current\_scheme.joins) {

object[] join;

if (obj is List<object> @j) join = @j.ToArray();

else if (obj is object[] @j2) join = @j2;

else { Log.Write("Одно из соединений не того типа: " + obj + " " + Utils.Obj2json(obj)); continue; }

if (join.Length != 6 ||

join[0] is not int @num\_a || join[1] is not int @pin\_a || join[2] is not string @tag\_a ||

join[3] is not int @num\_b || join[4] is not int @pin\_b || join[5] is not string @tag\_b) { Log.Write("Содержимое списка соединения ошибочно"); continue; }

var newy = new JoinedItems(new(items\_arr[@num\_a], @pin\_a, tag\_a), new(items\_arr[@num\_b], @pin\_b, tag\_b));

AddToMap(newy.line);

joinz.Add(newy);

}

foreach (var join in joinz) join.Update();

sim.Import(current\_scheme.states);

sim.lock\_sim = false;

if (start) sim.Start();

}

}

}

**Файл Project.cs**

using Avalonia.Controls;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

namespace LogicSimulator.Models {

public class Project: IComparable {

public string Name { get; private set; }

public long Created;

public long Modified;

public ObservableCollection<Scheme> schemes = new();

public string? FileDir { get; private set; }

public string? FileName { get; set; }

private readonly FileHandler parent;

public Project(FileHandler parent) { // Новый проект

this.parent = parent;

Name = "Новый проект";

Created = Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

FileDir = null;

FileName = null;

CreateScheme();

}

public Project(FileHandler parent, string dir, string fileName, object data) { // Импорт

this.parent = parent;

FileDir = dir;

FileName = fileName;

if (data is not Dictionary<string, object> dict) throw new Exception("Ожидался словарь в корне проекта");

if (!dict.TryGetValue("name", out var value)) throw new Exception("В проекте нет имени");

if (value is not string name) throw new Exception("Тип имени проекта - не строка");

Name = name;

if (!dict.TryGetValue("created", out var value2)) throw new Exception("В проекте нет времени создания");

if (value2 is not int create\_t) throw new Exception("Время создания проекта - не строка");

Created = create\_t;

if (!dict.TryGetValue("modified", out var value3)) throw new Exception("В проекте нет времени изменения");

if (value3 is not int mod\_t) throw new Exception("Время изменения проекта - не строка");

Modified = mod\_t;

if (!dict.TryGetValue("schemes", out var value4)) throw new Exception("В проекте нет списка схем");

if (value4 is not List<object> arr) throw new Exception("Списко схем проекта - не массив строк");

foreach (var s\_data in arr) {

if (s\_data == null) throw new Exception("Одно из файловых имёт списка схем проекта - null");

var scheme = new Scheme(this, s\_data);

schemes.Add(scheme);

}

}

public Scheme CreateScheme() {

var scheme = new Scheme(this);

schemes.Add(scheme);

Save();

return scheme;

}

public Scheme AddScheme(Scheme? prev) {

var scheme = new Scheme(this);

int pos = prev == null ? 0 : schemes.IndexOf(prev) + 1;

schemes.Insert(pos, scheme);

Save();

return scheme;

}

public void RemoveScheme(Scheme me) {

schemes.Remove(me);

Save();

}

public void UpdateList() {

foreach (var scheme in schemes) scheme.UpdateProps();

}

public Scheme GetFirstScheme() => schemes[0];

public object Export() {

return new Dictionary<string, object> {

["name"] = Name,

["created"] = Created,

["modified"] = Modified,

["schemes"] = schemes.Select(x => x.Export()).ToArray(),

};

}

public void Save() => FileHandler.SaveProject(this);

public int CompareTo(object? obj) {

if (obj is not Project proj) throw new ArgumentNullException(nameof(obj));

return (int)(proj.Modified - Modified);

}

public override string ToString() {

return Name + "\nИзменён: " + Modified.UnixTimeStampToString() + "\nСоздан: " + Created.UnixTimeStampToString();

}

internal void ChangeName(string name) {

Name = name;

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Save();

}

public bool CanSave() => FileDir != null;

public void SaveAs(Window mw) {

FileDir = FileHandler.RequestProjectPath(mw);

Save();

parent.AppendProject(this);

}

/\*

\* Для тестирования

\*/

public void SetDir(string path) => FileDir = path;

}

}

**Файл Sheme.cs**

using LogicSimulator.ViewModels;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Reactive;

namespace LogicSimulator.Models {

public class Scheme : ReactiveObject {

public string Name { get; set; }

public long Created;

public long Modified;

public object[] items;

public object[] joins;

public string states;

private readonly Project parent;

public Scheme(Project p) { // Новая схема

Created = Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Name = "Newy";

items = joins = Array.Empty<object>();

states = "0";

parent = p;

Open = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncOpen(); return new Unit(); });

NewItem = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncNewItem(); return new Unit(); });

Delete = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncDelete(); return new Unit(); });

}

public Scheme(Project p, object data) { // Импорт

parent = p;

if (data is not Dictionary<string, object> dict) throw new Exception("Ожидался словарь в корне схемы");

if (!dict.TryGetValue("name", out var value)) throw new Exception("В схеме нет имени");

if (value is not string name) throw new Exception("Тип имени схемы - не строка");

Name = name;

if (!dict.TryGetValue("created", out var value2)) throw new Exception("В схеме нет времени создания");

if (value2 is not int create\_t) throw new Exception("Время создания схемы - не строка");

Created = create\_t;

if (!dict.TryGetValue("modified", out var value3)) throw new Exception("В схеме нет времени изменения");

if (value3 is not int mod\_t) throw new Exception("Время изменения схемы - не строка");

Modified = mod\_t;

if (!dict.TryGetValue("items", out var value4)) throw new Exception("В схеме нет списка элементов");

if (value4 is not List<object> arr) throw new Exception("Список элементов схемы - не массив объектов");

items = arr.ToArray();

if (!dict.TryGetValue("joins", out var value5)) throw new Exception("В схеме нет списка соединений");

if (value5 is not List<object> arr2) throw new Exception("Список соединений схемы - не массив объектов");

joins = arr2.ToArray();

if (!dict.TryGetValue("states", out var value6)) throw new Exception("В схеме нет списка состояний");

if (value6 is not string arr3) throw new Exception("Список состояний схемы - не строка");

states = arr3;

Open = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncOpen(); return new Unit(); });

NewItem = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncNewItem(); return new Unit(); });

Delete = ReactiveCommand.Create<Unit, Unit>(\_ => { FuncDelete(); return new Unit(); });

}

public void Update(object[] items, object[] joins, string states) {

this.items = items;

this.joins = joins;

this.states = states;

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

Update();

}

public object Export() {

return new Dictionary<string, object> {

["name"] = Name,

["created"] = Created,

["modified"] = Modified,

["items"] = items,

["joins"] = joins,

["states"] = states,

};

}

public void Update() {

Modified = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

parent.Modified = Modified;

parent.Save();

}

public override string ToString() => Name;

internal void ChangeName(string name) {

Name = name;

Update();

}

/\*

\* Кнопочки

\*/

void FuncOpen() {

ViewModelBase.map.current\_scheme = this;

ViewModelBase.map.ImportScheme();

parent.UpdateList();

}

void FuncNewItem() {

parent.AddScheme(this);

parent.UpdateList();

}

void FuncDelete() {

parent.RemoveScheme(this);

parent.UpdateList();

}

public ReactiveCommand<Unit, Unit> Open { get; }

public ReactiveCommand<Unit, Unit> NewItem { get; }

public ReactiveCommand<Unit, Unit> Delete { get; }

public bool CanUseSchemeDeleter { get => parent.schemes.Count > 1; }

public bool CanOpenMe { get => ViewModelBase.map.current\_scheme != this; }

public void UpdateProps() {

this.RaisePropertyChanged(nameof(CanUseSchemeDeleter));

this.RaisePropertyChanged(nameof(CanOpenMe));

}

}

}

**Файл Simulator.cs**

using LogicSimulator.ViewModels;

using LogicSimulator.Views.Shapes;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LogicSimulator.Models {

public class Meta {

public IGate? item;

public int[] ins;

public int[] outs;

public bool[] i\_buf;

public bool[] o\_buf;

public Meta(IGate item, int out\_id) {

this.item = item;

ins = Enumerable.Repeat(0, item.CountIns).ToArray();

outs = Enumerable.Range(out\_id, item.CountOuts).ToArray();

i\_buf = Enumerable.Repeat(false, item.CountIns).ToArray();

o\_buf = Enumerable.Repeat(false, item.CountOuts).ToArray();

}

public void Print() {

Log.Write("Элемент: " + item + " | Ins: " + Utils.Obj2json(ins) + " | Outs: " + Utils.Obj2json(outs));

}

}

public class Simulator {

public Simulator() {

Start();

}

private Task? task;

private bool stop\_sim = false;

public bool lock\_sim = false;

public void Start() {

if (task != null || lock\_sim) return;

stop\_sim = false;

task = Task.Run(async () => {

for (; ; ) {

await Task.Delay(1000 / 1000);

try { Tick(); } catch (Exception e) { Log.Write("Logical crush: " + e); continue; }

if (stop\_sim) return;

}

});

}

public void Stop() {

if (task == null) return;

stop\_sim = true;

task.GetAwaiter().GetResult();

task = null;

}

List<bool> outs = new() { false };

List<bool> outs2 = new() { false };

readonly List<Meta> items = new();

readonly Dictionary<IGate, Meta> ids = new();

public void AddItem(IGate item) {

Stop();

int out\_id = outs.Count;

for (int i = 0; i < item.CountOuts; i++) {

outs.Add(false);

outs2.Add(false);

}

Meta meta = new(item, out\_id);

items.Add(meta);

ids.Add(item, meta);

Start();

}

public void RemoveItem(IGate item) {

Stop();

Meta meta = ids[item];

meta.item = null;

foreach (var i in Enumerable.Range(0, meta.outs.Length)) {

int n = meta.outs[i];

outs[n] = outs2[n] = false;

}

ids.Remove(item);

Start();

}

private void Tick() {

foreach (var meta in items) {

var item = meta.item;

if (item == null) continue;

item.LogicUpdate(ids, meta);

int[] i\_n = meta.ins, o\_n = meta.outs;

bool[] ib = meta.i\_buf, ob = meta.o\_buf;

for (int i = 0; i < ib.Length; i++) ib[i] = outs[i\_n[i]];

item.Brain(ref ib, ref ob);

for (int i = 0; i < ob.Length; i++) {

bool res = ob[i];

outs2[o\_n[i]] = res;

item.SetJoinColor(i, res);

}

}

(outs2, outs) = (outs, outs2);

if (comparative\_test\_mode) {

prev\_state = cur\_state;

cur\_state = Export();

}

}

public void Clean() {

int n = 0;

int[] arr = Enumerable.Repeat(-1, outs.Count).ToArray();

StringBuilder sb = new();

sb.Append('0');

foreach (var meta in items)

if (meta.item != null)

foreach (var @out in meta.outs) {

arr[@out] = ++n;

sb.Append(outs[@out] ? '1' : '0');

}

arr[0] = 0;

foreach (var meta in items) {

meta.outs = meta.outs.Select(x => arr[x]).ToArray();

meta.ins = meta.ins.Select(x => arr[x]).ToArray();

}

Import(sb.ToString());

}

public string Export() => string.Join("", outs.Select(x => x ? '1' : '0'));

public void Import(string state) {

if (state.Length == 0) state = "0";

outs = state.Select(x => x == '1').ToList();

outs2 = outs.ToList(); // clone

}

public void Clear() {

outs = new() { false };

outs2 = new() { false };

items.Clear();

ids.Clear();

}

/\*

\* Для тестирования

\*/

public void TopSecretPublicTickMethod() => Tick();

public Switch[] GetSwitches() => items.Select(x => x.item).OfType<Switch>().ToArray();

public LightBulb[] GetLightBulbs() => items.Select(x => x.item).OfType<LightBulb>().ToArray();

private bool comparative\_test\_mode = false;

private string prev\_state = "0";

private string cur\_state = "0";

public bool ComparativeTestMode {

get => comparative\_test\_mode;

set {

comparative\_test\_mode = value;

if (value) prev\_state = cur\_state = Export();

}

}

public bool SomethingHasChanged => prev\_state != cur\_state;

}

}

**Файл Utils.cs**

using System.Text;

using System.Xml.Linq;

using System.Collections.Generic;

using System.Linq;

using Avalonia.Controls;

using Avalonia.Media.Imaging;

using Avalonia;

using Avalonia.Media;

using System.Text.Json;

using LogicSimulator.ViewModels;

using System.Collections;

using System.Diagnostics;

using System;

using Avalonia.Controls.Shapes;

using System.Data.SQLite;

using System.Data;

namespace LogicSimulator.Models {

public static class Utils {

/\*

\* Base64 абилка

\*/

public static string Base64Encode(string plainText) {

var plainTextBytes = Encoding.UTF8.GetBytes(plainText);

return System.Convert.ToBase64String(plainTextBytes);

}

public static string Base64Decode(string base64EncodedData) {

var base64EncodedBytes = System.Convert.FromBase64String(base64EncodedData);

return Encoding.UTF8.GetString(base64EncodedBytes);

}

/\*

\* JSON абилка

\*/

public static string JsonEscape(string str) {

StringBuilder sb = new();

foreach (char i in str) {

sb.Append(i switch {

'"' => "\\\"",

'\\' => "\\\\",

'$' => "{$",

\_ => i

});

}

return sb.ToString();

}

public static string Obj2json(object? obj) {

switch (obj) {

case null: return "null";

case string @str: return '"' + JsonEscape(str) + '"';

case bool @bool: return @bool ? "true" : "false";

case short @short: return @short.ToString();

case int @int: return @int.ToString();

case long @long: return @long.ToString();

case float @float: return @float.ToString().Replace(',', '.');

case double @double: return @double.ToString().Replace(',', '.');

case Point @point: return "\"$p$" + (int) @point.X + "," + (int) @point.Y + '"';

case Size @size: return "\"$s$" + (int) @size.Width + "," + (int) @size.Height + '"';

case Points @points: return "\"$P$" + string.Join("|", @points.Select(p => (int) p.X + "," + (int) p.Y)) + '"';

case SolidColorBrush @color: return "\"$C$" + @color.Color + '"';

case Thickness @thickness: return "\"$T$" + @thickness.Left + "," + @thickness.Top + "," + @thickness.Right + "," + @thickness.Bottom + '"';

case Dictionary<string, object?> @dict: {

StringBuilder sb = new();

sb.Append('{');

foreach (var entry in @dict) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(Obj2json(entry.Key));

sb.Append(": ");

sb.Append(Obj2json(entry.Value));

}

sb.Append('}');

return sb.ToString();

}

case IEnumerable @list: {

StringBuilder sb = new();

sb.Append('[');

foreach (object? item in @list) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(Obj2json(item));

}

sb.Append(']');

return sb.ToString();

}

default: return "(" + obj.GetType() + " ???)";

}

}

private static object JsonHandler(string str) {

if (str.Length < 3 || str[0] != '$' || str[2] != '$') return str.Replace("{$", "$");

string data = str[3..];

string[] thick = str[1] == 'T' ? data.Split(',') : System.Array.Empty<string>();

return str[1] switch {

'p' => Point.Parse(data),

's' => Size.Parse(data),

'C' => new SolidColorBrush(Color.Parse(data)),

'T' => new Thickness(double.Parse(thick[0]), double.Parse(thick[1]), double.Parse(thick[2]), double.Parse(thick[3])),

\_ => str,

};

}

private static object? JsonHandler(object? obj) {

if (obj == null) return null;

if (obj is List<object?> @list) return @list.Select(JsonHandler).ToList();

if (obj is Dictionary<string, object?> @dict) {

return new Dictionary<string, object?>(@dict.Select(pair => new KeyValuePair<string, object?>(pair.Key, JsonHandler(pair.Value))));

}

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return null;

case JsonValueKind.Object:

Dictionary<string, object?> res = new();

foreach (var el in @item.EnumerateObject()) res[el.Name] = JsonHandler(el.Value);

return res;

case JsonValueKind.Array:

List<object?> res2 = @item.EnumerateArray().Select(item => JsonHandler((object?) item)).ToList();

return res2;

case JsonValueKind.String:

var s = JsonHandler(@item.GetString() ?? "");

// Log.Write("JS: '" + @item.GetString() + "' -> '" + s + "'");

return s;

case JsonValueKind.Number:

if (@item.ToString().Contains('.')) return @item.GetDouble();

// Иначе это целое число

long a = @item.GetInt64();

int b = @item.GetInt32();

// short c = @item.GetInt16();

if (a != b) return a;

// if (b != c) return b;

return b;

case JsonValueKind.True: return true;

case JsonValueKind.False: return false;

case JsonValueKind.Null: return null;

}

}

Log.Write("JT: " + obj.GetType());

return obj;

}

public static object? Json2obj(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return JsonHandler(data);

}

/\*

\* XML абилка

\*/

public static string XMLEscape(string str) {

StringBuilder sb = new();

foreach (char i in str) {

sb.Append(i switch {

'"' => "&quot;",

'\'' => "&apos;",

'>' => "&gt;",

'<' => "&lt;",

'&' => "&amp;",

\_ => i

});

}

return sb.ToString();

}

private static bool IsComposite(object? obj) {

if (obj == null) return false;

if (obj is List<object?> || obj is Dictionary<string, object?> || obj is not JsonElement @item) return true;

var T = @item.ValueKind;

return T == JsonValueKind.Object || T == JsonValueKind.Array;

}

private static string Dict2XML(Dictionary<string, object?> dict, string level) {

StringBuilder attrs = new();

StringBuilder items = new();

foreach (var entry in dict)

if (IsComposite(entry.Value))

items.Append(level + "\t<" + entry.Key + ">" + ToXMLHandler(entry.Value, level + "\t\t") + level + "\t</" + entry.Key + ">");

else attrs.Append(" " + entry.Key + "=\"" + ToXMLHandler(entry.Value, "{err}") + "\"");

if (items.Length == 0) return level + "<Dict" + attrs.ToString() + "/>";

return level + "<Dict" + attrs.ToString() + ">" + items.ToString() + level + "</Dict>";

}

private static string List2XML(List<object?> list, string level) {

StringBuilder attrs = new();

StringBuilder items = new();

int num = 0;

foreach (var entry in list) {

if (IsComposite(entry)) items.Append(ToXMLHandler(entry, level + "\t"));

else attrs.Append($" \_{num}='" + ToXMLHandler(entry, "{err}") + "'");

num++;

}

if (items.Length == 0) return level + "<List" + attrs.ToString() + "/>";

return level + "<List" + attrs.ToString() + ">" + items.ToString() + level + "</List>";

}

private static string ToXMLHandler(object? obj, string level) {

if (obj == null) return "null";

if (obj is List<object?> @list) return List2XML(@list, level);

if (obj is Dictionary<string, object?> @dict) return Dict2XML(@dict, level);

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return "undefined";

case JsonValueKind.Object:

return Dict2XML(new Dictionary<string, object?>(@item.EnumerateObject().Select(pair => new KeyValuePair<string, object?>(pair.Name, pair.Value))), level);

case JsonValueKind.Array:

return List2XML(@item.EnumerateArray().Select(item => (object?) item).ToList(), level);

case JsonValueKind.String:

var s = XMLEscape(@item.GetString() ?? "null");

// Log.Write("XS: '" + @item.GetString() + "' -> '" + s + "'");

return s;

case JsonValueKind.Number: return "$" + @item.ToString(); // escape NUM

case JsonValueKind.True: return "\_BOOL\_yeah";

case JsonValueKind.False: return "\_BOOL\_nop";

case JsonValueKind.Null: return "null";

}

}

Log.Write("XT: " + obj.GetType());

return "<UnknowType>" + obj.GetType() + "</UnknowType>";

}

public static string? Json2xml(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return "<?xml version=\"1.0\" encoding=\"utf-8\"?>" + ToXMLHandler(data, "\n");

}

private static string ToJSONHandler(string str) {

if (str.Length > 1 && str[0] == '$' && str[1] <= '9' && str[1] >= '0') return str[1..]; // unescape NUM

return str switch {

"null" => "null",

"undefined" => "undefined",

"\_BOOL\_yeah" => "true",

"\_BOOL\_nop" => "false",

\_ => '"' + str.Replace("\\", "\\\\") + '"',

};

}

private static string ToJSONHandler(XElement xml) {

var name = xml.Name.LocalName;

StringBuilder sb = new();

if (name == "Dict") {

sb.Append('{');

foreach (var attr in xml.Attributes()) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(ToJSONHandler(attr.Name.LocalName));

sb.Append(": ");

sb.Append(ToJSONHandler(attr.Value));

}

foreach (var el in xml.Elements()) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(ToJSONHandler(el.Name.LocalName));

sb.Append(": ");

sb.Append(ToJSONHandler(el.Elements().ToArray()[0]));

}

sb.Append('}');

} else if (name == "List") {

var attrs = xml.Attributes().ToArray();

var els = xml.Elements().ToArray();

int count = attrs.Length + els.Length;

var res = new string[count];

var used = new bool[count];

int num;

foreach (var attr in attrs) {

num = int.Parse(attr.Name.LocalName[1..]);

res[num] = ToJSONHandler(attr.Value);

used[num] = true;

}

num = 0;

foreach (var el in els) {

while (used[num]) num++;

res[num++] = ToJSONHandler(el);

}

sb.Append('[');

foreach (var item in res) {

if (sb.Length > 1) sb.Append(", ");

sb.Append(item);

}

sb.Append(']');

} else sb.Append("Type??" + name);

return sb.ToString();

}

public static string Xml2json(string xml) => ToJSONHandler(XElement.Parse(xml));

/\*

\* YAML абилка

\*/

public static string YAMLEscape(string str) {

string[] arr = new[] { "true", "false", "null", "undefined", "" };

if (arr.Contains(str)) return '"' + str + '"';

string black\_list = " -:\"\n\t";

bool escape = "0123456789[{".Contains(str[0]);

if (!escape)

foreach (char i in str)

if (black\_list.Contains(i)) { escape = true; break; }

if (!escape) return str;

StringBuilder sb = new();

sb.Append('"');

foreach (char i in str) {

sb.Append(i switch {

'"' => "\\\"",

'\\' => "\\\\",

\_ => i

});

}

sb.Append('"');

return sb.ToString();

}

private static string Dict2YAML(Dictionary<string, object?> dict, string level) {

if (dict.Count == 0) return " {}";

StringBuilder res = new();

foreach (var entry in dict)

res.Append(level + YAMLEscape(entry.Key) + ":" + (IsComposite(entry.Value) ? "" : " ") + ToYAMLHandler(entry.Value, level + "\t"));

return res.ToString();

}

private static string List2YAML(List<object?> list, string level) {

if (list.Count == 0) return " []";

StringBuilder res = new();

foreach (var entry in list)

res.Append(level + "-" + (IsComposite(entry) ? "" : " ") + ToYAMLHandler(entry, level + "\t"));

return res.ToString();

}

private static string ToYAMLHandler(object? obj, string level) {

if (obj == null) return "null";

if (obj is List<object?> @list) return List2YAML(@list, level);

if (obj is Dictionary<string, object?> @dict) return Dict2YAML(@dict, level);

if (obj is JsonElement @item) {

switch (@item.ValueKind) {

case JsonValueKind.Undefined: return "undefined";

case JsonValueKind.Object:

return Dict2YAML(new Dictionary<string, object?>(@item.EnumerateObject().Select(pair => new KeyValuePair<string, object?>(pair.Name, pair.Value))), level);

case JsonValueKind.Array:

return List2YAML(@item.EnumerateArray().Select(item => (object?) item).ToList(), level);

case JsonValueKind.String:

var s = YAMLEscape(@item.GetString() ?? "null");

// Log.Write("YS: '" + @item.GetString() + "' -> " + s);

return s;

case JsonValueKind.Number: return @item.ToString();

case JsonValueKind.True: return "true";

case JsonValueKind.False: return "false";

case JsonValueKind.Null: return "null";

}

}

Log.Write("YT: " + obj.GetType());

throw new Exception("Чё?!");

}

public static string? Json2yaml(string json) {

json = json.Trim();

if (json.Length == 0) return null;

object? data;

if (json[0] == '[') data = JsonSerializer.Deserialize<List<object?>>(json);

else if (json[0] == '{') data = JsonSerializer.Deserialize<Dictionary<string, object?>>(json);

else return null;

return "---" + ToYAMLHandler(data, "\n") + "\n"; // Конец будет обязателен, как в питоне!

}

private static void YAML\_Log(string mess, int level = 0) {

if (level >= 4) Log.Write(mess);

}

private static string YAML\_ParseString(ref string yaml, ref int pos) {

char first = ' ';

while (" \n\t".Contains(first)) first = yaml[pos++];

bool quote = first == '"';

StringBuilder sb = new();

if (quote) {

char c = yaml[pos++];

while (c != '"') {

sb.Append(c);

c = yaml[pos++];

}

c = yaml[pos++];

if (c != ':' && c != '\n') throw new Exception("После '\"' может быть только ':', либо '\n'");

if (c == ':') pos--;

} else {

sb.Append(first);

char c = yaml[pos++];

while (c != ':' && c != '\n') {

sb.Append(c);

c = yaml[pos++];

}

if (c == ':') pos--;

}

YAML\_Log("Parsed str: " + sb.ToString(), 1);

return sb.ToString();

}

private static string YAML\_ParseNum(ref string yaml, ref int pos) {

char c = yaml[pos++];

StringBuilder sb = new();

while ("0123456789.".Contains(c)) {

sb.Append(c);

c = yaml[pos++];

}

if (c != '\n') throw new Exception("После числа всяко должен быть '\n");

YAML\_Log("Parsed num: " + sb.ToString(), 1);

return sb.ToString();

}

private static string YAML\_ParseItem(ref string yaml, ref int pos) {

char first = ' ';

while (" \n\t".Contains(first)) first = yaml[pos++];

pos--;

if (first == '"')

return '"' + YAML\_ParseString(ref yaml, ref pos) + '"';

if ("0123456789".Contains(first))

return YAML\_ParseNum(ref yaml, ref pos);

string str = YAML\_ParseString(ref yaml, ref pos);

string[] arr = new[] { "true", "false", "null", "undefined", "", "[]", "{}" };

if (arr.Contains(str)) return str;

return '"' + str + '"';

}

private static string YAML\_ParseLayer(ref string yaml, ref int pos) {

if (pos == yaml.Length) return ""; // Конец файла

StringBuilder sb = new();

char first = yaml[pos++];

while (" \t".Contains(first)) {

sb.Append(first);

first = yaml[pos++];

}

pos--;

return sb.ToString();

}

private static string YAML\_ToJSONHandler(ref string yaml, ref int pos) {

var layer = YAML\_ParseLayer(ref yaml, ref pos);

if (pos == yaml.Length) return ""; // Конец файла

char first = yaml[pos++];

switch (first) {

case '[':

if (yaml[pos++] != ']' || yaml[pos++] != '\n') throw new Exception("После [ ожидалось ]\\n");

return "[]";

case '{':

if (yaml[pos++] != '}' || yaml[pos++] != '\n') throw new Exception("После { ожидалось }\\n");

return "{}";

case '-': {

StringBuilder res = new();

res.Append('[');

bool First = true;

pos--;

while (true) {

if (pos == yaml.Length) break; // Конец файла

if (First) First = false;

else {

var saved\_pos2 = pos;

var layer3 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("DOWN\_LAYER: '" + layer + "', '" + layer3 + "'");

if (layer != layer3) {

if (layer3.Length > layer.Length) throw new Exception("Ожидался элемент списка вместо подъёма");

if (!layer.StartsWith(layer3)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos2; break;

}

res.Append(", ");

}

if (yaml[pos++] != '-') throw new Exception("Ожидалось '-' в следующем элементе списка");

char c = yaml[pos++];

if (c == ' ') {

var value = YAML\_ParseItem(ref yaml, ref pos);

res.Append(value);

} else if (c == '\n') {

} else throw new Exception("После '-' ожидалось ' ', либо '\n'");

int saved\_pos = pos;

var layer2 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("LAYER: '" + layer + "', '" + layer2 + "'");

if (layer2.Length < layer.Length) {

if (!layer.StartsWith(layer2)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos; break;

}

if (!layer2.StartsWith(layer)) throw new Exception("Странность в следующем layer'е");

if (layer == layer2) { YAML\_Log("Сохранение"); pos = saved\_pos; continue; }

YAML\_Log("Подъём");

if (c == '\n') {

pos = saved\_pos;

var value = YAML\_ToJSONHandler(ref yaml, ref pos);

res.Append(value);

} else throw new Exception("Здесь не может быть подъёма");

}

res.Append(']');

YAML\_Log("Список рождён: " + res.ToString(), 2);

return res.ToString(); }

case '"':

default: {

pos--;

StringBuilder res = new();

res.Append('{');

bool First = true;

while (true) {

if (pos == yaml.Length) break; // Конец файла

if (First) First = false;

else {

var saved\_pos2 = pos;

var layer3 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("DICT\_LAYER: '" + layer + "', '" + layer3 + "'");

if (layer != layer3) {

if (layer3.Length > layer.Length) throw new Exception("Ожидался элемент словаря вместо подъёма");

if (!layer.StartsWith(layer3)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos2; break;

}

res.Append(", ");

}

var key = YAML\_ParseString(ref yaml, ref pos);

res.Append('"');

res.Append(key);

res.Append("\": ");

if (yaml[pos++] != ':') throw new Exception("После ключа ожидалось ':'");

char c = yaml[pos++];

if (c == ' ') {

var value = YAML\_ParseItem(ref yaml, ref pos);

res.Append(value);

} else if (c == '\n') {

} else throw new Exception("После ключа и ':' ожидалось ' ', либо '\n'");

int saved\_pos = pos;

var layer2 = YAML\_ParseLayer(ref yaml, ref pos);

YAML\_Log("LAYER: '" + layer + "', '" + layer2 + "'");

if (layer2.Length < layer.Length) {

if (!layer.StartsWith(layer2)) throw new Exception("Странность в упавшем layer'е");

YAML\_Log("Падение"); pos = saved\_pos; break;

}

if (!layer2.StartsWith(layer)) throw new Exception("Странность в следующем layer'е");

if (layer == layer2) { YAML\_Log("Сохранение"); pos = saved\_pos; continue; }

YAML\_Log("Подъём");

if (c == '\n') {

pos = saved\_pos;

var value = YAML\_ToJSONHandler(ref yaml, ref pos);

res.Append(value);

} else throw new Exception("Здесь не может быть подъёма");

}

res.Append('}');

YAML\_Log("Словарь рождён: " + res.ToString(), 2);

return res.ToString(); }

}

}

public static string Yaml2json(string yaml) {

try {

yaml = yaml.Replace("\r", "");

if (!yaml.StartsWith("---\n")) throw new Exception("Это не YAML");

int pos = 4;

var res = YAML\_ToJSONHandler(ref yaml, ref pos);

YAML\_Log("data: " + res, 3);

return res;

} catch (Exception e) { Log.Write("Ошибка YAML парсера: " + e); throw; }

}

/\*

\* Misc

\*/

public static string? Obj2xml(object? obj) => Json2xml(Obj2json(obj));

public static object? Xml2obj(string xml) => Json2obj(Xml2json(xml));

public static string? Obj2yaml(object? obj) => Json2yaml(Obj2json(obj));

public static object? Yaml2obj(string xml) => Json2obj(Yaml2json(xml));

public static void RenderToFile(Control target, string path) {

// var target = (Control?) tar.Parent;

// if (target == null) return;

double w = target.Bounds.Width, h = target.Bounds.Height;

var pixelSize = new PixelSize((int) w, (int) h);

var size = new Size(w, h);

using RenderTargetBitmap bitmap = new(pixelSize);

target.Measure(size);

target.Arrange(new Rect(size));

bitmap.Render(target);

bitmap.Save(path);

}

public static string TrimAll(this string str) {

StringBuilder sb = new();

for (int i = 0; i < str.Length; i++) {

if (i > 0 && str[i] == ' ' && str[i - 1] == ' ') continue;

sb.Append(str[i]);

}

return sb.ToString().Trim();

}

public static string[] NormSplit(this string str) => str.TrimAll().Split(' ');

public static string GetStackInfo() {

var st = new StackTrace();

var sb = new StringBuilder();

for (int i = 1; i < 11; i++) {

var frame = st.GetFrame(i);

if (frame == null) continue;

var method = frame.GetMethod();

if (method == null || method.ReflectedType == null) continue;

sb.Append(method.ReflectedType.Name + " " + method.Name + " | ");

if (i == 5) sb.Append("\n ");

}

return sb.ToString();

}

public static int Normalize(this int num, int min, int max) {

if (num < min) return min;

if (num > max) return max;

return num;

}

public static double Normalize(this double num, double min, double max) {

if (num < min) return min;

if (num > max) return max;

return num;

}

public static double Hypot(this Point delta) {

return Math.Sqrt(Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2));

}

public static double Hypot(this Point A, Point B) {

Point delta = A - B;

return Math.Sqrt(Math.Pow(delta.X, 2) + Math.Pow(delta.Y, 2));

}

public static double? ToDouble(this object num) {

return num switch {

int @int => @int,

long @long => @long,

double @double => @double,

\_ => null,

};

}

public static int Min(this int A, int B) => A < B ? A : B;

public static int Max(this int A, int B) => A > B ? A : B;

public static double Min(this double A, double B) => A < B ? A : B;

public static double Max(this double A, double B) => A > B ? A : B;

public static void Remove(this Control item) {

var p = (Panel?) item.Parent;

p?.Children.Remove(item);

}

public static Point Center(this Visual item, Visual? parent) {

var tb = item.TransformedBounds;

if (tb == null) return new();

var bounds = tb.Value.Bounds.TransformToAABB(tb.Value.Transform);

var res = bounds.Center;

if (parent == null) return res;

var tb2 = parent.TransformedBounds;

if (tb2 == null) return res;

var bounds2 = tb2.Value.Bounds.TransformToAABB(tb2.Value.Transform);

return res - bounds2.TopLeft;

}

public static DateTime UnixTimeStampToDateTime(this long unixTimeStamp) {

DateTime dateTime = new(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);

dateTime = dateTime.AddSeconds(unixTimeStamp).ToLocalTime();

return dateTime;

}

public static string UnixTimeStampToString(this long unixTimeStamp) {

return UnixTimeStampToDateTime(unixTimeStamp).ToString("yyyy/MM/dd H:mm:ss");

}

/\*

\* SQLite\_proj абилка

\*/

private static string Repr(this double d) => d.ToString().Replace(',', '.');

private static string Repr(this double? d) => d is not double value ? "0" : value.Repr();

private static string Repr(this Point? p) => p is not Point value ? "0, 0" : $"{value.X.Repr()}, {value.Y.Repr()}";

private static string Repr(this Size? p) => p is not Size value ? "0, 0" : $"{value.Width.Repr()}, {value.Height.Repr()}";

internal static void Obj2sqlite\_proj(Project proj, string path) {

// Log.Write(Obj2yaml(proj.Export()) + "");

using var con = new SQLiteConnection("Data Source=" + path);

con.Open();

if (con.State != ConnectionState.Open) { Log.Write("Не удалось открыть SQLite: " + con.State); return; }

string comm = $"INSERT INTO header VALUES ('{proj.Name}', {proj.Created}, {proj.Modified}, {proj.schemes.Count});";

new SQLiteCommand(comm, con).ExecuteReader().Close();

int n = 0;

foreach (var scheme in proj.schemes) {

List<string> res = new();

foreach (var item in scheme.items) {

if (item is not Dictionary<string, object> @dict) { Log.Write("Не верный тип элемента: " + item); continue; }

int? item\_id = null;

Point? pos = null;

Size? size = null;

double? base\_size = null;

List<string> extra = new();

foreach (var pair in dict) {

object value = pair.Value;

switch (pair.Key) {

case "id":

if (value is int @id) item\_id = @id;

else Log.Write("Неверный тип id-записи элемента: " + value);

break;

case "pos":

if (value is Point \_pos) pos = \_pos;

else Log.Write("Неверный тип pos-записи элемента: " + value);

break;

case "base\_size":

double? b\_size = value.ToDouble();

if (b\_size != null) base\_size = (double) b\_size;

else Log.Write("Неверный тип base\_size-записи элемента: " + value);

break;

case "size":

if (value is Size \_size) size = \_size;

else Log.Write("Неверный тип size-записи элемента: " + value);

break;

default:

value = value switch {

bool @bool => "$" + (@bool ? "1" : "0"),

string @str => "s" + @str,

\_ => throw new Exception("Не известный тип extra-данных: " + value.GetType().Name),

};

extra.Add(pair.Key + "=" + value);

break;

}

}

if (item\_id == null) { Log.Write("Не обнаружена id-запись элемента схемы :/"); continue; }

if (pos == null) { Log.Write("Не обнаружена pos-запись элемента схемы :/"); continue; }

if (size == null) { Log.Write("Не обнаружена size-запись элемента схемы :/"); continue; }

if (base\_size == null) { Log.Write("Не обнаружена base\_size-запись элемента схемы :/"); continue; }

string extra\_s = string.Join('&', extra);

res.Add($"({item\_id}, {pos.Repr()}, {size.Repr()}, {base\_size.Repr()}, '{extra\_s}')");

}

res.Add($"(-1234, {n}, {n}, {n}, {n}, {n}, '{scheme.states}')");

comm = $"INSERT INTO items VALUES {string.Join(", ", res)};";

// Log.Write(comm);

// File.WriteAllText("../../../check.txt", comm);

new SQLiteCommand(comm, con).ExecuteReader().Close();

res.Clear();

foreach (var obj in scheme.joins) {

object[] join;

if (obj is List<object> @j) join = @j.ToArray();

else if (obj is object[] @j2) join = @j2;

else { Log.Write("Одно из соединений не того типа: " + obj + " " + Obj2json(obj)); continue; }

if (join.Length != 6 ||

join[0] is not int @num\_a || join[1] is not int @pin\_a || join[2] is not string @tag\_a ||

join[3] is not int @num\_b || join[4] is not int @pin\_b || join[5] is not string @tag\_b) { Log.Write("Содержимое списка соединения ошибочно"); continue; }

res.Add($"({@num\_a}, {@pin\_a}, '{@tag\_a}', {@num\_b}, {@pin\_b}, '{@tag\_b}')");

}

res.Add($"(-1234, {n++}, '', {scheme.Created}, {scheme.Modified}, '{scheme.Name}')");

comm = $"INSERT INTO joins VALUES {string.Join(", ", res)};";

// Log.Write(comm);

new SQLiteCommand(comm, con).ExecuteReader().Close();

}

con.Dispose();

}

internal static object SQLite\_proj2obj(string path) {

using var con = new SQLiteConnection("Data Source=" + path);

con.Open();

if (con.State != ConnectionState.Open) throw new Exception("Не удалось открыть SQLite: " + con.State);

StringBuilder sb = new();

object res;

List<Dictionary<string, object>> schemes = new();

var sql\_comm = new SQLiteCommand("SELECT \* FROM header", con);

sql\_comm.ExecuteNonQuery();

using (var reader = sql\_comm.ExecuteReader()) {

if (!reader.HasRows) throw new Exception("Не вышло считать заголовочную таблицу SQLite :/");

if (!reader.Read()) throw new Exception("Заголовочная таблица пустует");

var row = Enumerable.Range(0, reader.VisibleFieldCount).Select(x => reader[x]).ToArray();

// Log.Write("row: " + Obj2json(row));

long count = (long) row[3];

List<object> @void = new();

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

schemes.Add(new() {

["name"] = "Новая схема",

["created"] = 0,

["modified"] = 0,

["items"] = @void,

["joins"] = @void,

["states"] = "",

});

res = new Dictionary<string, object> {

["name"] = row[0],

["created"] = (int) (long) row[1],

["modified"] = (int) (long) row[2],

["schemes"] = schemes.Cast<object>().ToList(),

};

}

using (var reader = new SQLiteCommand("SELECT \* FROM items", con).ExecuteReader()) {

if (!reader.HasRows) throw new Exception("Не вышло считать таблицу элементов :/");

int n = 0;

List<object> items = new();

while (reader.Read()) {

var row = Enumerable.Range(0, reader.VisibleFieldCount).Select(x => reader[x]).ToArray();

// Log.Write("item: " + Obj2json(row));

int id = (int) (long) row[0];

string extra = (string) row[6];

if (id == -1234) {

var scheme = schemes[n++];

scheme["items"] = items;

scheme["states"] = extra;

items = new();

continue;

}

var item = new Dictionary<string, object> {

["id"] = id,

["pos"] = new Point((double) row[1], (double) row[2]),

["size"] = new Size((double) row[3], (double) row[4]),

["base\_size"] = (double) row[5],

};

foreach (var pair in extra.Split('&')) {

if (pair == "") continue;

string[] kv = pair.Split('=');

string value = kv[1], data = value[1..];

char type = value[0];

object yeah = type switch {

'$' => data == "1" || (data == "0" ? false : throw new Exception($"Плохая extra (bool): {value}")),

's' => data,

\_ => throw new Exception($"Не известный тип extra-данных: '{type}'. Сами данные: {data}"),

};

item[kv[0]] = yeah;

}

items.Add(item);

}

}

using (var reader = new SQLiteCommand("SELECT \* FROM joins", con).ExecuteReader()) {

if (!reader.HasRows) throw new Exception("Не вышло считать таблицу соединений :/");

int n = 0;

List<object> joins = new();

while (reader.Read()) {

var row = Enumerable.Range(0, reader.VisibleFieldCount).Select(x => reader[x]).ToArray();

// Log.Write("join: " + Obj2json(row));

int num\_a = (int) (long) row[0];

int pin\_a = (int) (long) row[1];

var tag\_a = (string) row[2];

int num\_b = (int) (long) row[3];

int pin\_b = (int) (long) row[4];

var tag\_b = (string) row[5];

if (num\_a == -1234) {

var scheme = schemes[n++];

scheme["joins"] = joins;

scheme["created"] = num\_b;

scheme["modified"] = pin\_b;

scheme["name"] = tag\_b;

joins = new();

continue;

}

var join = new object[] { num\_a, pin\_a, tag\_a, num\_b, pin\_b, tag\_b };

joins.Add(join);

}

}

con.Dispose();

return res;

}

}

}