



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине «Проектирование и разработка серверных частей интернет-ресурсов»

Студент группы ИКБО-21-23

Муравьев А.О.

(подпись студента)

Руководитель практической работы

Благирев М.М.

(подпись руководителя)

Работа представлена

«___» _____ 2025 г.

Допущен к работе

«___» _____ 2025 г.

Москва 2025

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ	3
ХОД РАБОТЫ	4
ВЫВОД	12
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ ...	ОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.

ЦЕЛЬ РАБОТЫ

Предлагается, используя серверную конфигурацию, разработанную в прошлой практической работе выполнить следующие упражнения.

Предполагается создать 3 независимых сервиса, устойчивых к минимальному набору самых простых ошибок. Предполагается создание 1 общего проекта с разделенными сервисами, разделением проекта на файлы для разделения функционала и переиспользования файлов. Каждый сервис должен состоять как минимум из 2 файлов.

ХОД РАБОТЫ

Упражнение 1 - Drawer

Задача: создать веб-сервис Drawer для рисования svg объектов. Ему передается один параметр - целое число, представляющее закодированная фигура для рисования.

Суть реализации: при отправке GET-запроса с информацией о фигуре сервер возвращает код svg-фигуры с запрашиваемыми данными.

Обработкой запросов занимается класс WebServer. Он принимает и обрабатывает сырые запросы, используя другие компоненты программы.

Фрагмент класса WebServer показан на рисунке 1.

```
10 public class WebServer(string? wwwRoot = null)
11 {
12     [! usage] alexomur *
13     private async Task HandleClient(TcpClient client)
14     {
15         using (client)
16         {
17             NetworkStream stream = client.GetStream();
18             StreamReader reader = new(stream, Encoding.UTF8, detectEncodingFromByteOrderMarks: false, bufferSize: 8192, leaveOpen: true);
19             StreamWriter writer = new(stream, new UTF8Encoding(false), bufferSize: 8192, leaveOpen: true);
20             writer.NewLine = "\r\n";
21             try
22             {
23                 string requestLine = await reader.ReadLineAsync() ?? "";
24                 if (requestLine.Length == 0)
25                 {
26                     return;
27                 }
28                 string[] first = requestLine.Split(' ');
29                 if (first.Length < 3)
30                 {
31                     await WriteError(writer, status: 400, message: "Bad Request");
32                     return;
33                 }
34                 string method = first[0];
35                 string rawTarget = first[1];
36                 Dictionary<string, string> headers = new(StringComparer.OrdinalIgnoreCase);
37                 while (true)
38                 {
39                     string line = await reader.ReadLineAsync() ?? "";
40                     if (line.Length == 0)
41                     {
42                         break;
43                     }
44                     string[] parts = line.Split(':', 2);
45                     string key = parts[0].Trim();
46                     string value = parts[1].Trim();
47                     headers[key] = value;
48                 }
49                 string target = rawTarget.Trim();
50                 if (target.Length > 0)
51                 {
52                     target = target.TrimStart('/');
53                     if (target.Length > 0)
54                     {
55                         target = target.TrimStart('/');
56                     }
57                 }
58                 string path = target;
59                 string query = "";
60                 if (target.Contains('?'))
61                 {
62                     path = target.Split('?')[0];
63                     query = target.Split('?')[1];
64                 }
65                 string fileName = path;
66                 string extension = "";
67                 if (path.Contains('.'))
68                 {
69                     extension = path.Split('.')[1].Trim();
70                 }
71                 string contentType = "";
72                 if (extension.Length > 0)
73                 {
74                     contentType = GetContentType(extension);
75                 }
76                 string response = "";
77                 if (method.Equals("GET", StringComparison.OrdinalIgnoreCase))
78                 {
79                     response = GetResponse(path, query, contentType);
80                 }
81                 else if (method.Equals("POST", StringComparison.OrdinalIgnoreCase))
82                 {
83                     response = GetResponse(path, query, contentType);
84                 }
85                 else
86                 {
87                     await WriteError(writer, status: 405, message: "Method Not Allowed");
88                     return;
89                 }
90                 await writer.WriteLineAsync(response);
91                 await writer.FlushAsync();
92             }
93             catch { }
94         }
95     }
96 }
```

Рисунок 1 – Фрагмент класса WebServer

На рисунке 2 продемонстрирован фрагмент обработчика GET-запроса, который отвечает за сборку сырого ответа с svg-картинкой (картинка собирается в методе SvgRenderer.Render()).

```

10     public class WebServer(string? wwwRoot = null)
98         private async Task HandleClient(TcpClient client)
100             using (client)
106                 try
163                     if (string.Equals(path, "/drawer", StringComparison.OrdinalIgnoreCase) || string.Equals(path, "/drawer.svg", StringComparison.OrdinalIgnoreCase))
217                         string svg = SvgRenderer.Render(shape, colorId: colorIndex, w: width, h: height, stroke, pad: padding);
218                         await WriteResponse(writer, status: 200, contentType: "image/svg+xml", charset: "utf-8", body: Encoding.UTF8.GetBytes(svg));
219                         Log.Info("Served SVG " + shape.ToString() + " " + width.ToString(CultureInfo.InvariantCulture) + "x" + height.ToString(CultureInfo.InvariantCulture));
220                         return;
221                     }
222                     string filePath = Path.Combine(_wwwRoot, path.TrimStart('/'));
223                     filePath = Path.GetFullPath(filePath);
224                     if (!filePath.StartsWith(_wwwRoot, StringComparison.OrdinalIgnoreCase))
225                     {
226                         await WriteError(writer, status: 403, message: "Access denied");
227                         return;
228                     }
229                     if (File.Exists(filePath))
230                     {
231                         byte[] content = await File.ReadAllBytesAsync(filePath);
232                         string mime = ContentTypes.GetMimeType(filePath);
233                         await WriteResponse(writer, status: 200, mime, content);
234                         Log.Info("Served file: " + filePath);
235                         return;
236                     }
237                     await WriteError(writer, status: 404, message: "File not found");
238                 }
239                 catch (Exception e)
240                 {
241                     try { await WriteError(writer, status: 500, message: "Internal server error"); } catch { }

```

Рисунок 2 – Фрагмент обработчика GET-запросов

Основной компонент кода – класс `SvgRenderer`. С помощью тега `<svg>`, неймспейса `w3.org/2000/svg` и тегов `circle` и `rect` создаются круг и прямоугольник соответственно. Более сложные фигуры (треугольник и звезда) создаются с помощью `polygon` и вычислением точек (вершин) фигур. Фрагмент класса `SvgRenderer` показан на рисунках 3-4.

```

7     public static class SvgRenderer
9         public static string Render(ShapeType shape, int colorId, int w, int h, int stroke, int pad)
12             int idx = Math.Clamp(colorId, 0, maxIndex);
13             ColorType ct = (ColorType)idx;
14             string color = ct.ToHex();
15             double padX = w * Math.Clamp(pad, 0, 30) / 100.0;
16             double padY = h * Math.Clamp(pad, 0, 30) / 100.0;
17             double innerW = Math.Max(0, w - 2 * padX);
18             double innerH = Math.Max(0, h - 2 * padY);
19             string strokeAttr = stroke > 0 ? $" stroke=\"{color}\" stroke-width=\"{stroke.ToString(CultureInfo.InvariantCulture)}\" : \"\";
20             string open = $"<svg xmlns=\"http://www.w3.org/2000/svg\" width=\"{w}\" height=\"{h}\" viewBox=\"0 0 {w} {h}\">";
21             if (shape == ShapeType.Circle)
22             {
23                 double r = Math.Min(innerW, innerH) / 2.0;
24                 double cx = w / 2.0;
25                 double cy = h / 2.0;
26                 return FormattableString.Invariant($"
27                 {open}<circle cx=\"{cx:0.###}\" cy=\"{cy:0.###}\" r=\"{r:0.###}\" fill=\"{color}\"{strokeAttr}/>
28                 \"\"\"");
29             }
30             if (shape == ShapeType.Rectangle)
31             {
32                 return FormattableString.Invariant($"
33                 {open}<rect x=\"{padX:0.###}\" y=\"{padY:0.###}\" width=\"{innerW:0.###}\" height=\"{innerH:0.###}\" fill=\"{color}\"{strokeAttr}/>
34                 \"\"\"");
35             }
36             if (shape == ShapeType.Triangle)
37             {
38                 double s = Math.Min(innerW, innerH);
39                 double cx = w / 2.0;

```

Рисунок 3 – Фрагмент класса SvgRenderer, часть 1 из 2

```

7 public static class SvgRenderer
9 public static string Render(ShapeType shape, int colorId, int w, int h, int stroke, int pad)
35 }
36 if (shape == ShapeType.Triangle)
37 {
38     double s = Math.Min(innerW, innerH);
39     double cx = w / 2.0;
40     double cy = h / 2.0;
41     double hh = s * Math.Sqrt(3) / 2.0;
42     double yTop = cy - hh / 2.0;
43     double yBottom = cy + hh / 2.0;
44     double xL = cx - s / 2.0;
45     double xR = cx + s / 2.0;
46     return FormattableString.Invariant($"<open><polygon points="{xL:0.###},{yBottom:0.###},{xR:0.###},{yBottom:0.###},{cx:0.###},{yTop:0.###}" fill="{color}" {strokeAttr}/></svg>
47     """);
48 }
49 if (shape == ShapeType.Star)
50 {
51     double s = Math.Min(innerW, innerH);
52     double cx = w / 2.0;
53     double cy = h / 2.0;
54     double outerR = s / 2.0;
55     double innerR = outerR * 0.5;
56     string pts = BuildStar(cx, cy, outerR, innerR, n:5);
57     return $"<open><polygon points="{pts}" fill="{color}" {strokeAttr}/></svg>";
58 }
59 return $"<open><text x=\"10\" y=\"20\">Unknown</text></svg>";
60 }
61 }

```

Рисунок 4 – Фрагмент класса SvgRenderer, часть 2 из 2

В результате при отправке GET-запроса с данными фигуры мы получаем запрошенную svg-картинку.

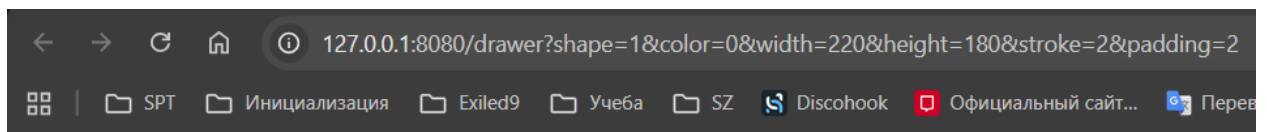


Рисунок 5 – Результат работы Drawer

Упражнение 2 - Sorter

Задача: реализовать сортировку слиянием.

Массив передается скрипту как параметр строки: состоящий из значений элементов массива, разделенных запятыми. Итогом является вебстраница, содержащая отсортированный массив.

Суть решения: класс WebServer принимает сырой GET-запрос, разархивирует данные и отправляет их на сортировку классу MergeSorter, а затем возвращает страничку с отсортированным массивом.

Фрагмент класса WebServer, отвечающий за вызов сортировки показан на рисунке 6.

```
9      public class WebServer(string? wwwRoot = null)
85          private async Task HandleClient(TcpClient client)
87              using (client)
93                  try
146                      if (string.Equals(path, "/sort", StringComparison.OrdinalIgnoreCase))
168                          int[] sorted = MergeSorter.Sort(numbers);
169                          string[] texts = new string[sorted.Length];
170                          for (int i = 0; i < sorted.Length; i++) texts[i] = sorted[i].ToString(CultureInfo.InvariantCulture);
171                          string result = string.Join(", ", texts);
172                          string html = HtmlPage.BuildResult(input: src, sorted: result);
173                          await WriteResponse(writer, status: 200, contentType: "text/html; charset=utf-8", body: Encoding.UTF8.GetBytes(html));
174                          Log.Info("Sorted " + numbers.Length.ToString(CultureInfo.InvariantCulture) + " numbers");
175                          return;
176                      }
177                      string filePath = Path.Combine(_wwwRoot, path.TrimStart('/'));
178                      string full = Path.GetFullPath(filePath);
179                      if (!full.StartsWith(_wwwRoot, StringComparison.OrdinalIgnoreCase))
180                      {
181                          await WriteError(writer, status: 403, message: "Access denied");
182                          return;
183                      }
184                      if (File.Exists(full))
185                      {
186                          byte[] content = await File.ReadAllBytesAsync(full);
187                          string mime = Utils.ContentTypes.GetMimeType(full);
188                          string ct = NeedsUtf8(mime) ? mime + "; charset=utf-8" : mime;
189                          await WriteResponse(writer, status: 200, ct, content);
190                          Log.Info("Served file: " + full);
191                          return;
```

Рисунок 6 – Фрагмент WebServer, отвечающий за вызов сортировки

На рисунке 7 показан статический метод сортировки в классе MergeSorter.

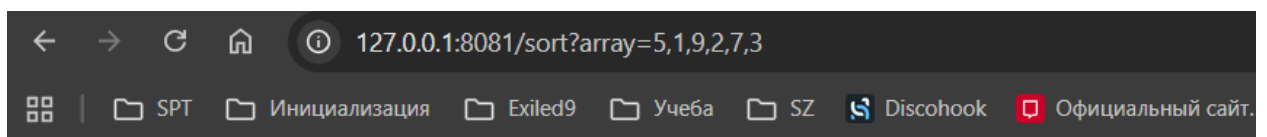
```

5 public static class MergeSorter
7     public static int[] Sort(int[]? source)
12         if (n <= 1) return source.AsSpan().ToArray();
13         int[] buffer = new int[n];
14         int[] a = source.AsSpan().ToArray();
15         int[] b = buffer;
16         int width = 1;
17         while (width < n)
18         {
19             int i = 0;
20             while (i < n)
21             {
22                 int left = i;
23                 int mid = Math.Min(i + width, n);
24                 int right = Math.Min(i + 2 * width, n);
25                 int p = left;
26                 int q = mid;
27                 int k = left;
28                 while (p < mid && q < right)
29                 {
30                     if (a[p] <= a[q])
31                         b[k++] = a[p++];
32                     else
33                         b[k++] = a[q++];
34                 }
35                 while (p < mid) b[k++] = a[p++];
36                 while (q < right) b[k++] = a[q++];
37                 i += 2 * width;
38             }
39             (a, b) = (b, a);
40             width *= 2;
41         }
42         return a;

```

Рисунок 7 – Метод сортировки слиянием

Теперь при отправке GET-запроса серверу с перечислением элементов массива мы получаем страничку с отсортированными элементами этого массива (рисунок 8).



Результат сортировки

<p>Вход</p> <p>5,1,9,2,7,3</p>	<p>Выход</p> <p>1, 2, 3, 5, 7, 9</p>
<p>Назад</p>	

Рисунок 8 – Результат работы Sorter

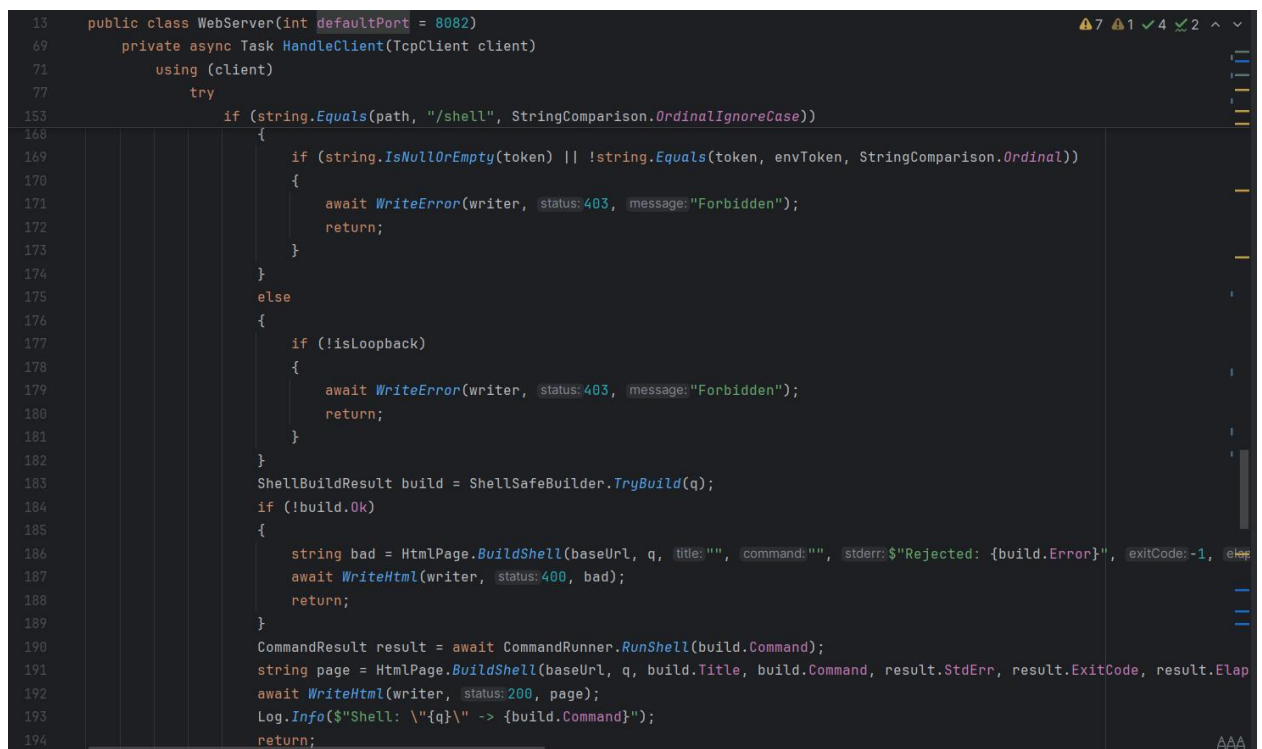
Упражнение 3 - AdminPanel

Задача: реализовать информационно-административную веб-страницу о сервере с помощью таких команд Unix как: ls, ps, whoami, id и так далее.

Суть решения: класс WebServer работает с сырыми запросами и отправляет данные в другие классы, ShellSafeBuilder принимает unix-команды, проверяет их и преобразовывает в зависимости от текущей системы (под windows shell и unix команды), Commands занимается очисткой и уточнением команд (флагами, параметрами и прочим), CommandRunner занимается запуском команд в зависимости от текущей системы.

Для реализации пародии на SSH-доступ был добавлен токен доступа, хранящийся в окружении (конфиге docker-compose). В рамках данной работы токен также будет передаваться через GET-запрос.

Фрагмент класса WebServer, работающий с Shell, показан на рисунке 9.



```
13 public class WebServer(int defaultPort = 8082)
69     private async Task HandleClient(TcpClient client)
71         using (client)
77             try
153                 if (string.Equals(path, "/shell", StringComparison.OrdinalIgnoreCase))
168                 {
169                     if (string.IsNullOrEmpty(token) || !string.Equals(token, envToken, StringComparison.Ordinal))
170                     {
171                         await WriteError(writer, status: 403, message: "Forbidden");
172                         return;
173                     }
174                 }
175                 else
176                 {
177                     if (!IsLoopback)
178                     {
179                         await WriteError(writer, status: 403, message: "Forbidden");
180                         return;
181                     }
182                 }
183                 ShellBuildResult build = ShellSafeBuilder.TryBuild(q);
184                 if (!build.Ok)
185                 {
186                     string bad = HtmlPage.BuildShell(baseUrl, q, title: "", command: "", stderr: $"Rejected: {build.Error}", exitCode: -1, ele
187                     await WriteHtml(writer, status: 400, bad);
188                     return;
189                 }
190                 CommandResult result = await CommandRunner.RunShell(build.Command);
191                 string page = HtmlPage.BuildShell(baseUrl, q, build.Title, build.Command, result.Stderr, result.ExitCode, result.Elap
192                 await WriteHtml(writer, status: 200, page);
193                 Log.Info($"Shell: \"{q}\" -> {build.Command}");
194                 return;
```

Рисунок 9 – Фрагмент класса WebServer, отвечающий за сборку команды и её вызов

Фрагмент класса ShellSafeBuilder, проверяющий команды, показан на рисунке 10.

```

7 public static class ShellSafeBuilder
8 {
9     private static readonly HashSet<string> UnixAllowed = new HashSet<string>(StringComparer.OrdinalIgnoreCase)
10     {
11         "whoami", "id", "uname", "uptime", "df", "free", "ps", "env", "pwd", "hostname", "ls", "cat", "head", "tail"
12     };
13
14     private static readonly Regex SafeArgUnix = new Regex("[A-Za-z0-9_\\-\\.\\/~@:+=,\\s]{1,2048}$");
15     private static readonly Regex SafeArgWin = new Regex("[A-Za-z0-9_\\-\\.\\/~\\\\\\\\:0+=,\\s]{1,2048}$");
16     private static readonly char[] Ws = new[] { ' ', '\t', '\n', '\n' };
17
18     [usage] alexomur
19     public static ShellBuildResult TryBuild(string input)
20     {
21         if (string.IsNullOrEmpty(input)) return ShellBuildResult.Fail(error: "Empty", hint: "Введите команду");
22         string trimmed = input.Trim();
23         if (trimmed.Length > 2048) return ShellBuildResult.Fail(error: "Too long", hint: "Слишком длинная команда");
24         string[] parts = SplitOnce(trimmed);
25         string cmd = parts[0];
26         string rest = parts[1];
27         bool isWindows = OperatingSystem.IsWindows();
28         if (isWindows)
29         {
30             if (!SafeArgWin.IsMatch(trimmed)) return ShellBuildResult.Fail(error: "Bad chars", hint: "Недопустимые символы");
31             if (string.Equals(cmd, "whoami", StringComparison.OrdinalIgnoreCase)) return Ok(title: "whoami", command: "whoami", key: "whoami");
32             if (string.Equals(cmd, "hostname", StringComparison.OrdinalIgnoreCase)) return Ok(title: "hostname", command: "hostname", key: "hostname");
33             if (string.Equals(cmd, "dir", StringComparison.OrdinalIgnoreCase) || string.Equals(cmd, "ls", StringComparison.OrdinalIgnoreCase))
34             {
35                 string path = string.IsNullOrEmpty(rest) ? "." : rest.Trim();
36                 string ps = "Get-ChildItem -Force -LiteralPath " + ShellQuotes.PwshQuote(path) + " | Format-Table -AutoSize";
37                 return Ok(title: "ls -la " + path, command: ps, key: "ls");
38             }
39             if (string.Equals(cmd, "type", StringComparison.OrdinalIgnoreCase) || string.Equals(cmd, "cat", StringComparison.OrdinalIgnoreCase))

```

Рисунок 10 – Фрагмент класса ShellSafeBuilder

Фрагмент класса Commands, уточняющий команды, показан на рисунке

11.

```

6 public static class Commands
7 {
8     static CommandSpec Build(string key, string path)
9     {
10         isWindows = OperatingSystem.IsWindows();
11         string.Equals(key, "whoami", StringComparison.OrdinalIgnoreCase) return new CommandSpec(title: "whoami", command: "whoami");
12         string.Equals(key, "id", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "whoami /all", command: "whoami /all") :
13         string.Equals(key, "uname", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "system info", command: "[System.Environment]::OSVersion.ToString()") :
14         string.Equals(key, "uptime", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "uptime", command: "(Get-Date) - (Get-Date) - ([System.Environment]::OSVersion.ToString())") :
15         string.Equals(key, "df", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "drives", command: "Get-PSDrive -PSProvider FileSystem") :
16         string.Equals(key, "free", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "memory", command: "Get-CimInstance Win32_PerfFormattedData-Win32_PerfMon-Win32_PerfMon") :
17         string.Equals(key, "ps", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "processes", command: "Get-Process | Format-Table -AutoSize") :
18         string.Equals(key, "env", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "env", command: "Get-ChildItem env: | Format-Table -AutoSize") :
19         string.Equals(key, "pwd", StringComparison.OrdinalIgnoreCase) return isWindows ? new CommandSpec(title: "pwd", command: "Get-Location") : new CommandSpec(title: key, command: key);
20         string.Equals(key, "hostname", StringComparison.OrdinalIgnoreCase) return new CommandSpec(title: "hostname", command: "hostname");
21         string.Equals(key, "ls", StringComparison.OrdinalIgnoreCase)
22         {
23             string safe = SanitizePath(path, isWindows);
24             if (string.IsNullOrEmpty(safe)) safe = ".";
25             if (isWindows)
26             {
27                 string cmdW = "Get-ChildItem -Force -LiteralPath " + ShellQuotes.PwshQuote(safe) + " | Format-Table -AutoSize";
28                 string titleW = "ls -la " + safe;
29                 return new CommandSpec(titleW, command: cmdW);
30             }
31             else
32             {
33                 string cmd = "ls -la -- " + ShellQuotes.BashQuote(safe);
34                 string title = "ls -la " + safe;
35                 return new CommandSpec(title, command: cmd);
36             }
37         }
38         return new CommandSpec(title: key, command: key);
39     }
40 }

```

Рисунок 11 – Фрагмент класса Commands

Фрагмент класса CommandRunner, отвечающий за запуск команд,

показан на рисунке 12.

```

9      public static class CommandRunner
11         public static async Task<CommandResult> RunShell(string command)
14             ProcessStartInfo psi = new ProcessStartInfo();
15             if (isWindows)
16             {
17                 string escaped = command.Replace("`", "`").Replace("\", "\\");
18                 psi.FileName = "powershell";
19                 psi.Arguments = "-NoProfile -ExecutionPolicy Bypass -Command \"" + escaped + "\"";
20             }
21             else
22             {
23                 string shell = File.Exists(path: "/bin/sh") ? "/bin/sh" : "sh";
24                 psi.FileName = shell;
25                 psi.Arguments = "-lc \"" + command.Replace("\", "\\") + "\"";
26             }
27             psi.RedirectStandardOutput = true;
28             psi.RedirectStandardError = true;
29             psi.UseShellExecute = false;
30             psi.CreateNoWindow = true;
31             psi.StandardOutputEncoding = Encoding.UTF8;
32             psi.StandardErrorEncoding = Encoding.UTF8;
33             Process p = new Process();
34             p.StartInfo = psi;
35             DateTime start = DateTime.UtcNow;
36             p.Start();
37             string stdout = await p.StandardOutput.ReadToEndAsync();
38             string stderr = await p.StandardError.ReadToEndAsync();
39             await p.WaitForExitAsync();
40             int code = p.ExitCode;
41             long ms = (long)(DateTime.UtcNow - start).TotalMilliseconds;
42             CommandResult result = new CommandResult(code, stdout, stderr, ms);
43             return result;

```

Рисунок 12 – Фрагмент класса CommandRunner

ВЫВОД

Таким образом, были разработаны 3 независимых сервиса: Drawer, Sorter и AdminPanel.

Исходный код проекта расположен по адресу:

<https://github.com/alexomur/MireaBackend/tree/master/Prac2>