



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №9

Технологія розробки програмного забезпечення

Тема роботи: «РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE»

Виконала
студентка групи ІА-12:
Яковенко Дар'я

Перевірив:
вик. Колеснік В. М.

Київ 2023

Тема: РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE

Мета:

1. Реалізувати функціонал для роботи в розподіленому оточенні (логіку роботи).
2. Реалізувати взаємодію розподілених частин.

Завдання:

28 JSON Tool (ENG) (strategy, command, observer, template method, flyweight)
Display JSON schema with syntax highlight. Validate JSON schema and display errors. Create user friendly table\list box\other for read and update JSON schema properties metadata (description, example, data type, format, etc.). Auto save\restore when edit, maybe history. Can check JSON value by schema (Put schema and JSON = valid\invalid, display errors). Export schema as markdown table. JSON to "flat" view.

Хід роботи

Клієнт-серверні додатки.



Клієнт-серверна взаємодія, як правило, організовується за допомогою 3-х рівневої структури: клієнтська частина, загальна частина, серверна частина. Оскільки велика частина даних загальна (класи, використовувані системою), їх прийнято виносити в загальну частину (middleware) системи. Клієнтська частина

містить візуальне відображення і логіку обробки дії користувача; код для встановлення сеансу зв'язку з сервером і виконання відповідних викликів. Серверна частина містить основну логіку роботи програми (бізнес-логіку) або ту її частину, яка відповідає зберіганню або обміну даними між клієнтом і сервером або клієнтами.

Було реалізовано клієнт-серверний варіант для мого додатку для роботи з даними JSON.

Наступний код представляє собою **серверну частину**:

```
server.js
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const fs = require('fs');
4  const app = express();
5  const port = 3000;
6  app.use(bodyParser.json());
7
8  app.post( path: '/create-file', handlers: (req :... , res : Response<ResBody, LocalsObj> ) => {
9      const {jsonFileName, jsonContent, metadataFileName, metadataContent} = req.body;
10     saveToFile( folderName: 'JsonProjects', jsonFileName, jsonContent);
11     saveToFile( folderName: 'MetadataFiles', metadataFileName, metadataContent);
12
13     res.send( body: 'Сервер: Файл успішно створено');
14 });
15
16
17 app.post( path: '/save-file', handlers: (req :... , res : Response<ResBody, LocalsObj> ) => {
18     const {jsonFileName, jsonContent, metadataFileName, metadataContent} = req.body;
19
20     saveToFile( folderName: 'JsonProjects', jsonFileName, jsonContent);
21     saveToFile( folderName: 'MetadataFiles', metadataFileName, metadataContent);
22
23     res.send( body: 'Сервер: Файл успішно збережено');
24 });
25
26 app.get('/get-file-list', (req :... , res : Response<ResBody, LocalsObj> ) => {
27     const files = getFileList( folderName: 'JsonProjects');
28     res.status( code: 200).json( body: {files});
29 });
30
31 1 usage
32 function getFileList(folderName) {
33     const folderPath = `./${folderName}`;
34     if (fs.existsSync(folderPath)) {
35         const files = fs.readdirSync(folderPath);
```

```

35     return files;
36 }
37 return [];
38 }
39
40 4 usages
41 function saveToFile(folderName, fileName, content) {
42     const folderPath = `./${folderName}`;
43
44     if (!fs.existsSync(folderPath)) {
45         fs.mkdirSync(folderPath);
46     }
47
48     fs.writeFileSync( file: `${folderPath}/${fileName}`, content);
49 }
50
51 app.get('/get-file-content', (req :... , res : Response<ResBody, LocalsObj> ) => {
52     const fileName = req.query.fileName;
53
54     if (!fileName) {
55         res.status( code: 400).send( body: 'Bad Request: Missing fileName parameter');
56         return;
57     }
58
59     try {
60         const jsonContent = fs.readFileSync( path: `./JsonProjects/${fileName}`, options: 'utf-8');
61         const metadataFileName = `metadata_${fileName.replace( searchValue: /\.json$/, replaceValue: '.txt')}`;
62         const metadataContent = fs.readFileSync( path: `./MetadataFiles/${metadataFileName}`, options: 'utf-8');
63         const response = {
64             jsonContent: jsonContent,
65             metadataContent: metadataContent
66         };
67
68         res.json(response);
69     } catch (error) {
70         console.error(error);
71         res.status( code: 500).send( body: 'Internal Server Error');
72     }
73 });
74
75 app.listen(port, hostname: () => {
76     console.log(`Server is running at http://localhost:${port}`);
77 });

```

Призначення:

- Забезпечує серверну логіку.
- Взаємодіє з клієнтом через HTTP ендпоінти.

Endpoints:

1. /create-file та /save-file (POST):

- Призначення:
 - Обробка запитів на створення або збереження файлів.
 - Отримання даних від клієнта та збереження їх відповідно.

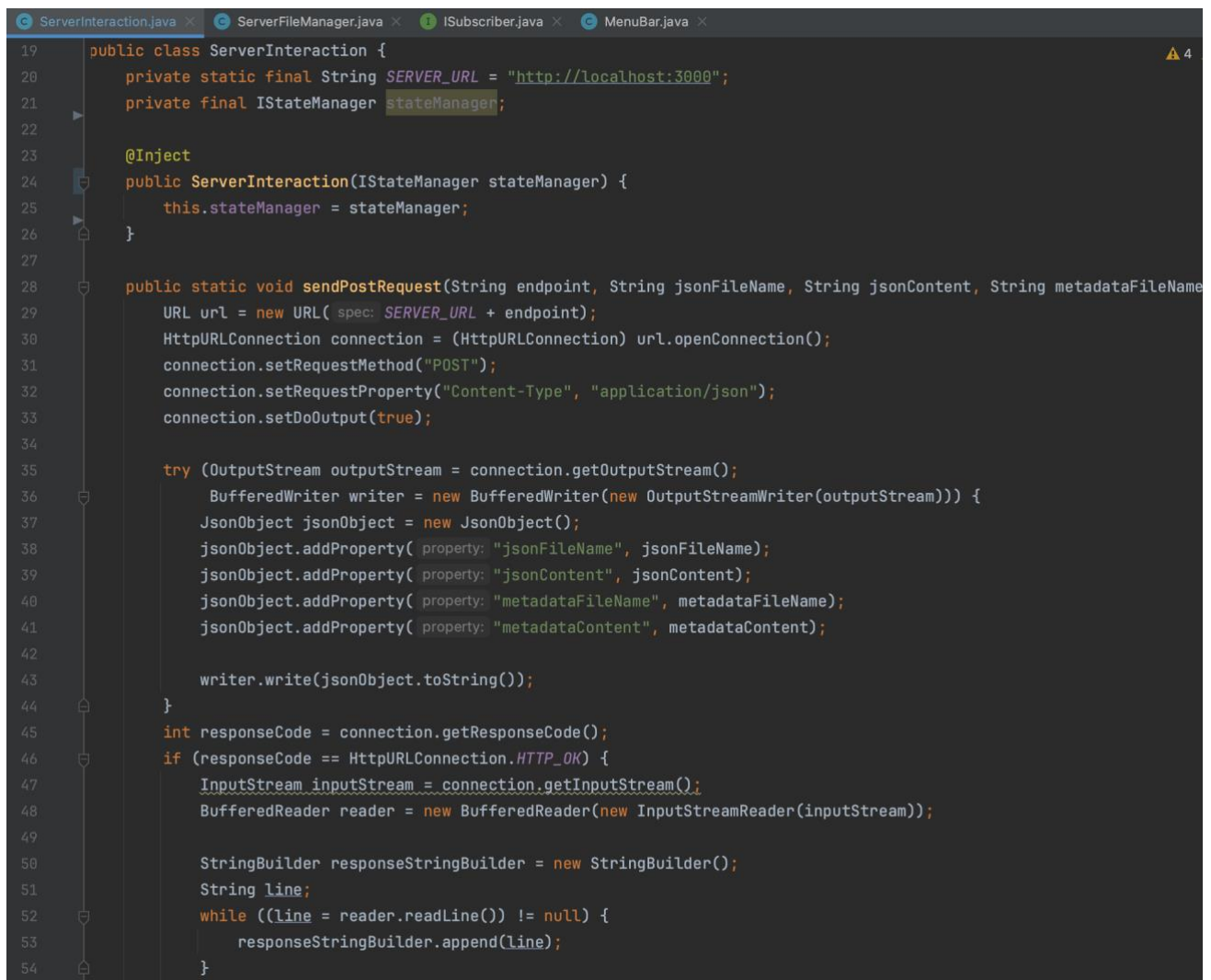
2. /get-file-list (GET):

- Призначення:
 - Надсилання списку файлів клієнту.
 - Використовує `getFileList` для отримання списку файлів.

3. `/get-file-content (GET)`:

- Призначення:
 - Надсилання змісту файлу клієнту.
 - Використовується `fs.readFileSync` для читання вмісту `Json` та `Metadata` файлів.

Middleware частина додатку:



```
19 public class ServerInteraction {
20     private static final String SERVER_URL = "http://localhost:3000";
21     private final IStateManager stateManager;
22
23     @Inject
24     public ServerInteraction(IStateManager stateManager) {
25         this.stateManager = stateManager;
26     }
27
28     public static void sendPostRequest(String endpoint, String jsonFileName, String jsonContent, String metadataFileName) {
29         URL url = new URL( spec: SERVER_URL + endpoint);
30         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
31         connection.setRequestMethod("POST");
32         connection.setRequestProperty("Content-Type", "application/json");
33         connection.setDoOutput(true);
34
35         try (OutputStream outputStream = connection.getOutputStream();
36             BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outputStream))) {
37             JSONObject jsonObject = new JSONObject();
38             jsonObject.addProperty( property: "jsonFileName", jsonFileName);
39             jsonObject.addProperty( property: "jsonContent", jsonContent);
40             jsonObject.addProperty( property: "metadataFileName", metadataFileName);
41             jsonObject.addProperty( property: "metadataContent", metadataContent);
42
43             writer.write(jsonObject.toString());
44         }
45
46         int responseCode = connection.getResponseCode();
47         if (responseCode == HttpURLConnection.HTTP_OK) {
48             InputStream inputStream = connection.getInputStream();
49             BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
50
51             StringBuilder responseStringBuilder = new StringBuilder();
52             String line;
53             while ((line = reader.readLine()) != null) {
54                 responseStringBuilder.append(line);
55             }
56         }
57     }
58 }
```

```

55         String responseMessage = responseStringBuilder.toString();
56         System.out.println(responseMessage);
57         connection.disconnect();
58     } else {
59         System.out.println("Failed to create file on the server. Response Code: " + responseCode);
60         connection.disconnect();
61     }
62     connection.disconnect();
63 }
64
65 public static List<String> getFileList() throws IOException {
66     URL url = new URL( spec: SERVER_URL + "/get-file-list");
67     HttpURLConnection connection = (HttpURLConnection) url.openConnection();
68     connection.setRequestMethod("GET");
69     int responseCode = connection.getResponseCode();
70     if (responseCode == HttpURLConnection.HTTP_OK) {
71         InputStream inputStream = connection.getInputStream();
72         BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
73
74         StringBuilder responseStringBuilder = new StringBuilder();
75         String line;
76         while ((line = reader.readLine()) != null) {
77             responseStringBuilder.append(line);
78         }
79         String response = responseStringBuilder.toString();
80
81         Gson gson = new Gson();
82         FileListResponse fileListResponse = gson.fromJson(response, FileListResponse.class);
83
84         if (fileListResponse != null) {
85             List<String> fileList = fileListResponse.getFiles();
86             connection.disconnect();
87             return fileList;
88         } else {
89             System.out.println("Failed to parse file list response.");
90             connection.disconnect();
91             return Collections.emptyList();
92         }
93     } else {
94         System.out.println("Failed to get file list from the server. Response Code: " + responseCode);
95         connection.disconnect();
96         return Collections.emptyList();
97     }
98 }
99
100 public class FileListResponse {
101     private List<String> files;
102
103     public List<String> getFiles() { return files; }
104 }
105
106 }
107

```

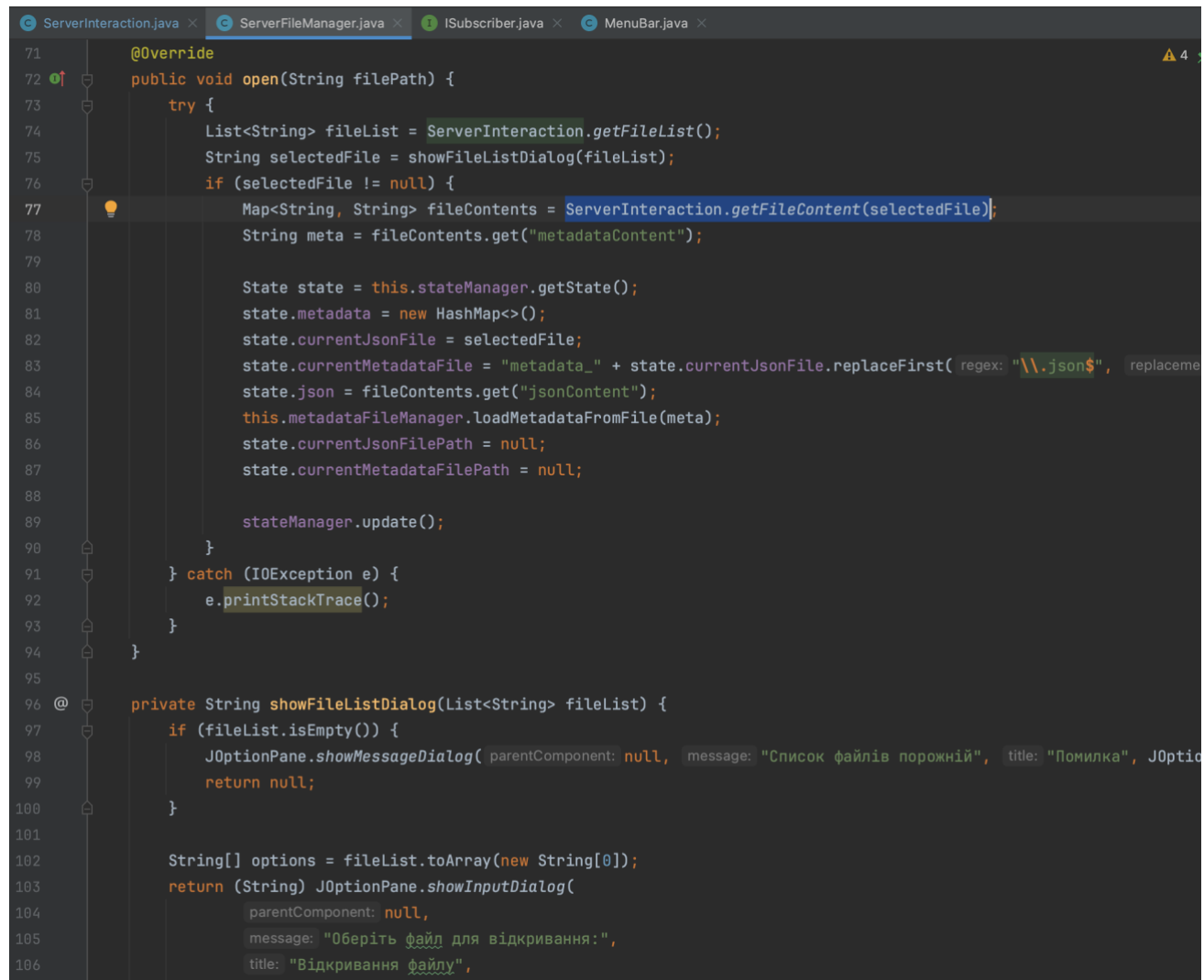
1. ServerInteraction.java:

- Призначення:
 - Надсилає HTTP запити на сервер для взаємодії з файлами.
 - Використовує Gson для обробки JSON.
 - Виводить повідомлення про результат на консоль.

2. FileListResponse class:

- Призначення:
 - Модель для парсингу відповіді від сервера зі списком файлів.

Клас **ServerFileManager** – клієнтська частина додатку:



```
71 @Override
72 public void open(String filePath) {
73     try {
74         List<String> fileList = ServerInteraction.getFileList();
75         String selectedFile = showFileListDialog(fileList);
76         if (selectedFile != null) {
77             Map<String, String> fileContents = ServerInteraction.getFileContent(selectedFile);
78             String meta = fileContents.get("metadataContent");
79
80             State state = this.stateManager.getState();
81             state.metadata = new HashMap<>();
82             state.currentJsonFile = selectedFile;
83             state.currentMetadataFile = "metadata_" + state.currentJsonFile.replaceFirst(regex: "\\..json$", replace
84             state.json = fileContents.get("jsonContent");
85             this.metadataFileManager.loadMetadataFromFile(meta);
86             state.currentJsonFilePath = null;
87             state.currentMetadataFilePath = null;
88
89             stateManager.update();
90         }
91     } catch (IOException e) {
92         e.printStackTrace();
93     }
94 }
95
96 @private String showFileListDialog(List<String> fileList) {
97     if (fileList.isEmpty()) {
98         JOptionPane.showMessageDialog(parentComponent: null, message: "Список файлів порожній", title: "Помилка", JOptio
99         return null;
100     }
101
102     String[] options = fileList.toArray(new String[0]);
103     return (String) JOptionPane.showInputDialog(
104         parentComponent: null,
105         message: "Оберіть файл для відкриття:",
106         title: "Відкриття файлу",
```



```

107         JOptionPane.PLAIN_MESSAGE,
108         icon: null,
109         options,
110         options[0]
111     );
112 }
113
114 @Override
115 public void save() {
116     State state = this.stateManager.getState();
117
118     try {
119         String metadataContent = "[";
120         for (Map.Entry<String, String> entry : stateManager.getState().metadata.entrySet()) {
121             metadataContent += entry.getKey() + "=" + entry.getValue() + ", ";
122         }
123         if (!stateManager.getState().metadata.isEmpty()) {
124             metadataContent = metadataContent.substring(0, metadataContent.length() - 2);
125         }
126         metadataContent += "]";
127         ServerInteraction.sendPostRequest(endpoint: "/save-file", state.currentJsonFile, state.json, state.currentMet
128         this.stateManager.update();
129     } catch (IOException e) {
130         e.printStackTrace();
131     }
132 }

```

ServerFileManager відповідає за взаємодію з сервером для створення, відкриття та збереження файлів. Опишу наведені методи на скріншоті:

- **Метод open:**
 - Відображає діалог вибору файлу для відкриття.
 - Отримує вміст та метадані обраного файлу з сервера.
 - Оновлює стан програми та завантажує метадані.
- **Метод showFileListDialog:**
 - Відображає діалог для вибору файлу зі списку.
 - Повертає обраний файл або **null**, якщо список порожній.
- **Метод save:**
 - Створює рядок, представляючий метадані у відповідному форматі.
 - Відправляє POST-запит на сервер для збереження файлу разом із метаданими.
 - Оновлює стан програми.

Висновки:

У цій лабораторній роботі я ознайомилася з шаблонами, реалізувала функціонал для роботи в розподіленому оточенні, виконала взаємодію мого додатку з сервером.