Dokumentacija završnog rada “Alat za transformaciju računalno generiranih alarma u statičku web stranicu”

Mentori: doc. dr. sc. Domagoj Ševerdija
Damir Birovljević (ENEA)

Student: Dominik Spajić

Uvod

U suvremenim IT sustavima, upravljanje alarmima je ključan element za održavanje stabilnosti i sigurnosti. Aplikacija razvijena u sklopu ovog završnog projekta omogućava prikupljanje, analizu i vizualizaciju alarma generiranih unutar sustava. Glavna funkcionalnost aplikacije je prikaz svih relevantnih alarma na jednoj statičkoj web stranici, s mogućnostima pretraživanja i filtriranja, čime se olakšava praćenje i rješavanje potencijalnih problema.

Aplikacija je dizajnirana za rad s velikim brojem alarma, koji su prikazani na jednostavan i pregledan način. Njena upotreba omogućava brzo prepoznavanje kritičnih problema i olakšava proces njihovog otklanjanja.

Tehnologije

Aplikacija je implementirana koristeći sljedeće tehnologije i alate:

- **Python:** za parsiranje ulaznih datoteka i generiranje HTML dokumenta
- **HTML/CSS:** Za generiranje statičke web stranice koja prikazuje alarme.
- **JavaScript:** Za omogućavanje pretraživanja i filtriranja unutar generirane web stranice.

Postavljanje i korištenje aplikacije

1. Priprema podataka

Aplikacija očekuje ulazne datoteke u specifičnom formatu gdje svaka linija predstavlja jedan alarm. Datoteka mora imati sljedeću strukturu:

- **Zaglavlje:** Prva linija sadrži nazive polja (ID, Informacija, Parametri, Ozbiljnost, Objašnjenje, Opis, Popravak, Kontekst).
- **Podaci:** Svaka naredna linija sadrži podatke o jednom alarmu, odvojene posebnim znakom ¤ (U+00A4).

*Primjer ulazne linije (*razdvojena u više redova radi čitljivosti na slici*):

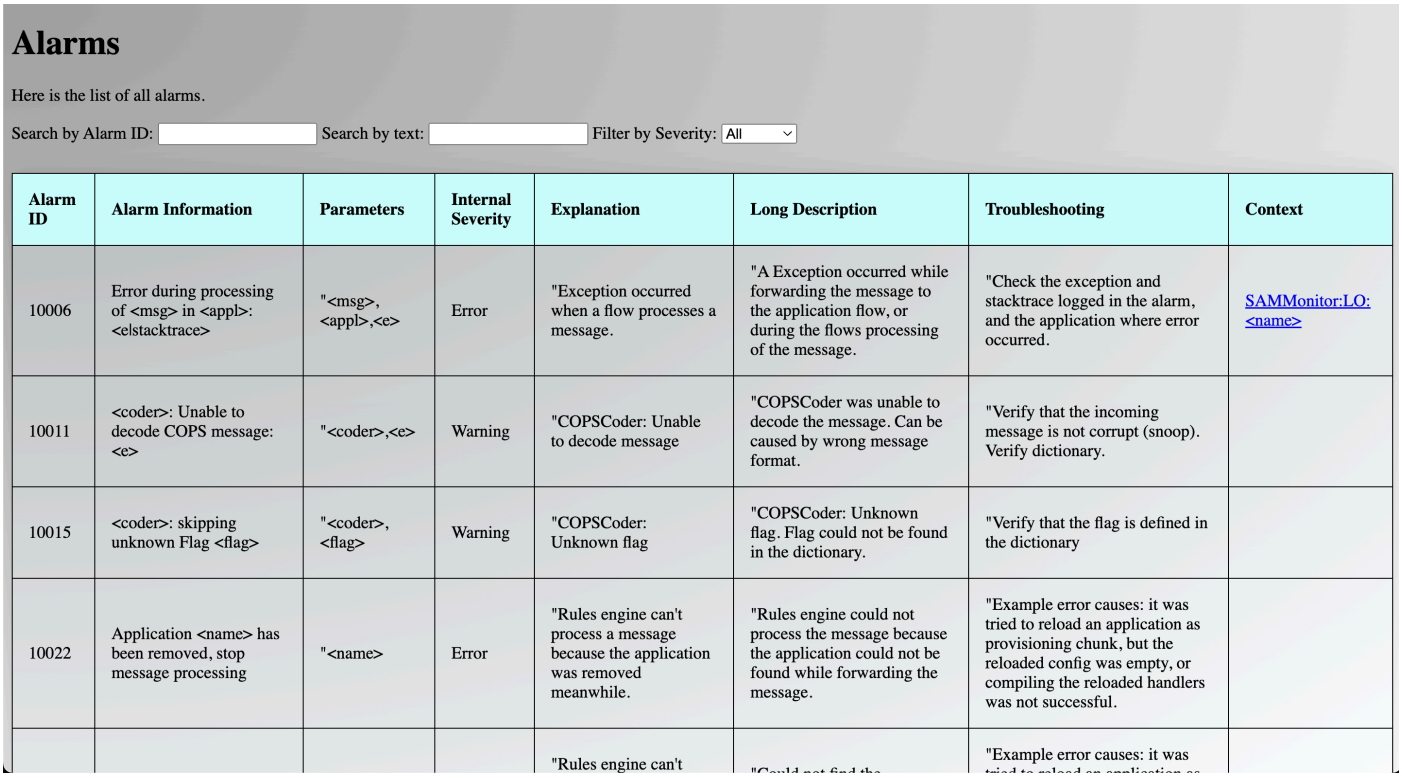
```
10131¤"<me>: Unable to create coder: <e>"¤parameter="<me>,<e>"¤'Error'¤short="CoderPlug: Failed to create coder"
¤long="CoderPlug could not create coder due to wrong configuration.
"¤repair="Verify the values of configuration parameters 'Protocol', 'Encoding' and the dictionary name."
¤context=/++?
```

Figura 1: isječak jednog alarma iz datoteke “alarms.all”

2. Generiranje web stranice

Aplikacija generira HTML stranicu s tablicom svih alarma. Stranica sadrži sljedeće elemente:

- **Tablica alarma:** Prikazuje sve učitane alarme s mogućnošću sortiranja.
- **Pretraživanje:** Omogućava pretragu alarma prema ID-u ili tekstu.
- **Filtriranje prema ozbiljnosti:** Omogućava filtriranje alarma prema razini ozbiljnosti (Error, Warning)
- **Povezivanje prema kontekstu:** Alarmi koji dijele isti kontekst mogu se prikazati zajedno putem hiperlinka.



Alarms

Here is the list of all alarms.

Search by Alarm ID: Search by text: Filter by Severity: All

Alarm ID	Alarm Information	Parameters	Internal Severity	Explanation	Long Description	Troubleshooting	Context
10006	Error during processing of <msg> in <appl>: <elstacktrace>	"<msg>, <appl>,<e>	Error	"Exception occurred when a flow processes a message.	"A Exception occurred while forwarding the message to the application flow, or during the flows processing of the message.	"Check the exception and stacktrace logged in the alarm, and the application where error occurred.	SAMMonitor:LO: <name>
10011	<coder>: Unable to decode COPS message: <e>	"<coder>,<e>	Warning	"COPSCoder: Unable to decode message	"COPSCoder was unable to decode the message. Can be caused by wrong message format.	"Verify that the incoming message is not corrupt (snoop). Verify dictionary.	
10015	<coder>: skipping unknown Flag <flag>	"<coder>, <flag>	Warning	"COPSCoder: Unknown flag	"COPSCoder: Unknown flag. Flag could not be found in the dictionary.	"Verify that the flag is defined in the dictionary	
10022	Application <name> has been removed, stop message processing	"<name>	Error	"Rules engine can't process a message because the application was removed meanwhile.	"Rules engine could not process the message because the application could not be found while forwarding the message.	"Example error causes: it was tried to reload an application as provisioning chunk, but the reloaded config was empty, or compiling the reloaded handlers was not successful.	
				"Rules engine can't		"Example error causes: it was	

Figura 2: Prikaz HTML stranice sa pripadnim elementima

3. Korištenje web stranice

Korisnik može pretraživati i filtrirati alarme kako bi brzo identificirao i rješavao potencijalne probleme. Stranica automatski osvježava prikaz prema unesenim kriterijima pretrage i filtriranja.

Figura 3: Prikaz HTML tablice sa statistikom o alarmima

Severity	Count
Total alarms	10
Warning	4
Error	6

Razvoj aplikacije

1. Učitavanje podataka

Aplikacija koristi Python skriptu koja čita ulazne datoteke, parsira podatke i generira HTML kod za prikaz alarma. Svaka linija iz datoteke se dekomponira u odgovarajuće elemente (ID, informacija, parametri itd.), koji se zatim prikazuju u tablici na web stranici.

Na Figuri 3. je prikaz početka aplikacije, gdje učitavamo potrebne biblioteke “os” i “html” za manipulaciju operativnim sustavom i HTML datotekama.

Definirana je funkcija **clean_field** koja služi za parsiranje parametra iz linije datoteke alarma. Ona prima kao parametar varijablu **field** koja predstavlja člana polja koje sadržava parametre naših alarma. Služi za uklananje svih nepotrebnih simbola i praznih prostora te prefiksa. Funkcija vraća “očišćenu” vrijednost parametra alarma.

```python
import os
import html

def clean_field(field):
    field = field.strip().strip('"').strip("'")
    prefixes = ["parameter=", "short=", "long=", "repair=", "context="]
    for prefix in prefixes:
        if field.startswith(prefix):
            field = field[len(prefix):]
    return html.escape(field)
```

Figura 3: Isječak funkcije clean_field

Na Figuri 4. je prikazana funkcija **load_alarms** koja učitava datoteku alarma, iterira kroz linije te parsira i sprema vrijednosti parametara alarma u riječnik (dictionary) u “key:value” obliku. Funkcija vraća kao izlaznu vrijednost riječnik “alarms”.

Funkcija u iterativnom procesu preskače prvu liniju u našoj datoteci sa informacijama alarma jer prva linija u generiranoj datoteci alarma ne sadrži informacije o alarmu, već samo imena parametara te ju tako preskačemo u procesu parsiranja.

```
def load_alarms(file_paths):
    alarms = []

    for file_path in file_paths:
        text = open(file_path, "r", encoding='utf-8')
        lines = text.readlines()
        for line in lines[1:]:
            fields = line.split('¤')
            print(fields)
            if len(fields) < 8:
                    continue
            alarm = {
                    'Alarm ID': clean_field(fields[0]),
                    'Alarm Information': clean_field(fields[1]),
                    'Parameters': clean_field(fields[2]),
                    'Internal Severity': clean_field(fields[3]),
                    'Explanation': clean_field(fields[4]),
                    'Long Description': clean_field(fields[5]),
                    'Troubleshooting': clean_field(fields[6]),
                    'Context': clean_field(fields[7])
                }
            alarms.append(alarm)
    return alarms
```

Figura 4: Isječak koda funkcije load_alarms

2. Generiranje HTML-a

Funkcija **generate_html** služi za procesiranje podataka alarma u HTML elemente te razvoj HTML datoteke. Na Figuri 5 možemo vidjeti isječak koda funkcije koji kreira izbornik za filtriranje po parametru “severity”, odnosno filtriranje po ozbiljnosti.

Nadalje, možemo vidjeti kako funkcija za svaki alarm spremljen u našem riječniku stvara hiperlink za filtriranje po kontekstu (ukoliko ga ima), te stvara HTML element retka za tablicu koji sadrži informacije o parametrima tog alarma.

Na kraju, funkcija radi dvije varijable koje sadrže ukupan broj alarma te broj alarma po određenim skupinama ozbiljnosti.

```
def generate_html(alarms):
    severity_options = set(alarm['Internal Severity'] for alarm in alarms)
    severity_options_html = '\n'.join(f'<option value="{severity}">{severity}</option>' for severity in sorted(severity_options))

    alarms_html = ''
    for alarm in alarms:
        context = alarm['Context']
        context_html = f'<a href="#" class="context-link" data-context="{context}" onclick="filterByContext(event)">{context}</a>'

        alarms_html += f'''
        <tr>
            <td>{(alarm["Alarm ID"])}</td>
            <td>{alarm["Alarm Information"]}</td>
            <td>{alarm["Parameters"]}</td>
            <td>{alarm["Internal Severity"]}</td>
            <td>{alarm["Explanation"]}</td>
            <td>{alarm["Long Description"]}</td>
            <td>{alarm["Troubleshooting"]}</td>
            <td>{context_html}</td>
        </tr>
        '''

    total_alarms = len(alarms)
    severity_counts = {severity: sum(1 for alarm in alarms if alarm['Internal Severity'] == severity) for severity in severity_options}
```

Figura 5: Isječak koda funkcije generate_html

Funkcija zatim stvara HTML element tablice koji sadržava informacije o alarmima, kao što su broj ukupnih alarma te broj alarma po ozbiljnosti.

Varijable “style” i “script sadržavaju CSS i JavaScript kod za stiliziranje stranice te funckionalnosti filtriranja i pretraživanja.

Naposlijetku, funkcija stvara varijablu html_content koja sadrži ukupni sadržaj koji pišemo u našu novonastalu HTML datoteku.

```
stats_html = f'''…
for severity, count in severity_counts.items():
    stats_html += f'''
        <tr>
            <td>{severity}</td>
            <td>{count}</td>
        </tr>
    '''
stats_html += '''
    </tbody>
</table>
'''
```

Figura 6**:** Isječak koda statistike alarma

```
style = '''
<style>
    * {
        box-sizing: border-box;
    }

    table {
        width: 100%;
        border-collapse: collapse;
        overflow: hidden;
        box-shadow: 0 0 20px rgba(0,0,0,0.1);
        margin-top: 2%;
    }

    th, td {
        border: 1px solid black;
        text-align: left;
        padding: 15px;
        background-color: rgba(231, 253, 255, 0.2);
        color: #000000;
    }

    th {
        cursor: pointer;
        text-align: left;
    }

    thead th {
        background-color: #c9feff;
    }

    html {
        height: 100%;
        width: 100%;
        background-size: cover;
        background-image: radial-gradient(ellipse at top left, #a0a0a0, #ffffff);
    }
```

Figura 7.1: isječak CSS koda

```
    tbody tr:hover {
        background-color: rgba(255,255,255,0.3);
    }

    tbody td {
        position: relative;
    }

    tbody td:hover:before {
        content: "";
        position: absolute;
        left: 0;
        right: 0;
        top: -9999px;
        bottom: -9999px;
        background-color: rgba(255,255,255,0.2);
        z-index: -1;
    }

    .stats {
        width: 50%;
    }
</style>
'''
```

Figura 7.2: isječak CSS koda

```
    script = '''
    <script>
        function filterAlarms() {
  const idInput = document.getElementById("search-id").value.toLowerCase();
  const textInput = document.getElementById("search-text").value.toLowerCase();
  const severityFilter = document.getElementById("filter-severity").value;
  const rows = document.querySelectorAll("#alarms-table tr");

  rows.forEach((row) => {
    const cells = row.getElementsByTagName("td");
    const idMatch = cells[0].textContent.toLowerCase().includes(idInput);
    const textMatch = Array.from(cells).some((cell) =>
      cell.textContent.toLowerCase().includes(textInput)
    );
    const severityMatch =
      severityFilter === "All" || cells[3].textContent === severityFilter;
    row.style.display = idMatch && textMatch && severityMatch ? "" : "none";
  });
}

document.querySelectorAll(".context-link").forEach((link) => {
  link.addEventListener("click", (event) => {
    event.preventDefault();
    const context = event.target.dataset.context;
    document.querySelectorAll("#alarms-table tr").forEach((row) => {
      const contextCell = row.getElementsByTagName("td")[7];
      row.style.display =
        contextCell && contextCell.textContent === context ? "" : "none";
    });
  });
});
```

Figura 8.1: isječak JavaScript koda

```
function filterByContext(event) {
  event.preventDefault();

  const context = event.target.getAttribute('data-context');

  const rows = document.querySelectorAll('#alarms-table tr');

  rows.forEach(row => {
      const link = row.querySelector('.context-link');
      if (link) {
          const rowContext = link.getAttribute('data-context');
          if (rowContext === context) {
              row.style.display = '';
          } else {
              row.style.display = 'none';
          }
      }
  });
}
    </script>
    '''
```

Figura 8.2: isječak JavaScript koda

```
html_content = f'''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Alarms</title>
    {style}
</head>
<body>
    <h1>Alarms</h1>
    <p>Here is the list of all alarms.</p>
    <label for="search-id">Search by Alarm ID:</label>
    <input type="text" id="search-id" onkeyup="filterAlarms()">
    <label for="search-text">Search by text:</label>
    <input type="text" id="search-text" onkeyup="filterAlarms()">
    <label for="filter-severity">Filter by Severity:</label>
    <select id="filter-severity" onchange="filterAlarms()">
        <option value="All">All</option>
        {severity_options_html}
    </select>
    <table>
        <thead>
            <tr>
                <th>Alarm ID</th>
                <th>Alarm Information</th>
                <th>Parameters</th>
                <th>Internal Severity</th>
                <th>Explanation</th>
                <th>Long Description</th>
                <th>Troubleshooting</th>
                <th>Context</th>
            </tr>
        </thead>
        <tbody id="alarms-table">
            {alarms_html}
        </tbody>
    </table>
    <div id="stats">
        {stats_html}
    </div>
   {script}
</body>
</html>
'''

with open('alarms.html', 'w', encoding='utf-8') as f:
    f.write(html_content)
```

Figura 9: cijelokupni HTML sadržaj

Testiranje i implementacija

Nakon generiranja web stranice, aplikacija je testirana s različitim ulaznim datotekama kako bi se osigurala točnost prikaza i funkcionalnost pretrage i filtriranja. Web stranica je pregledana na različitim uređajima kako bi se osigurala kompatibilnost i responzivnost.

```
#Main script
file_paths = ['alarms1.txt']
alarms = load_alarms(file_paths)
generate_html(alarms)
```

Figura 10: glavna skripta aplikacije

Zaključak

Aplikacija za upravljanje i prikaz alarma razvijena u sklopu ovog završnog rada predstavlja jednostavno, ali učinkovito rješenje za praćenje stanja sustava. Korištenjem ove aplikacije, administratori sustava mogu brzo identificirati i reagirati na potencijalne probleme, čime se poboljšava stabilnost i sigurnost cijelog IT okruženja.