

Introduction to the Data Acquisition Service (DAS) of the Web Intelligence Hub (WIH)

Mészáros Mátyás – Eurostat Unit A5

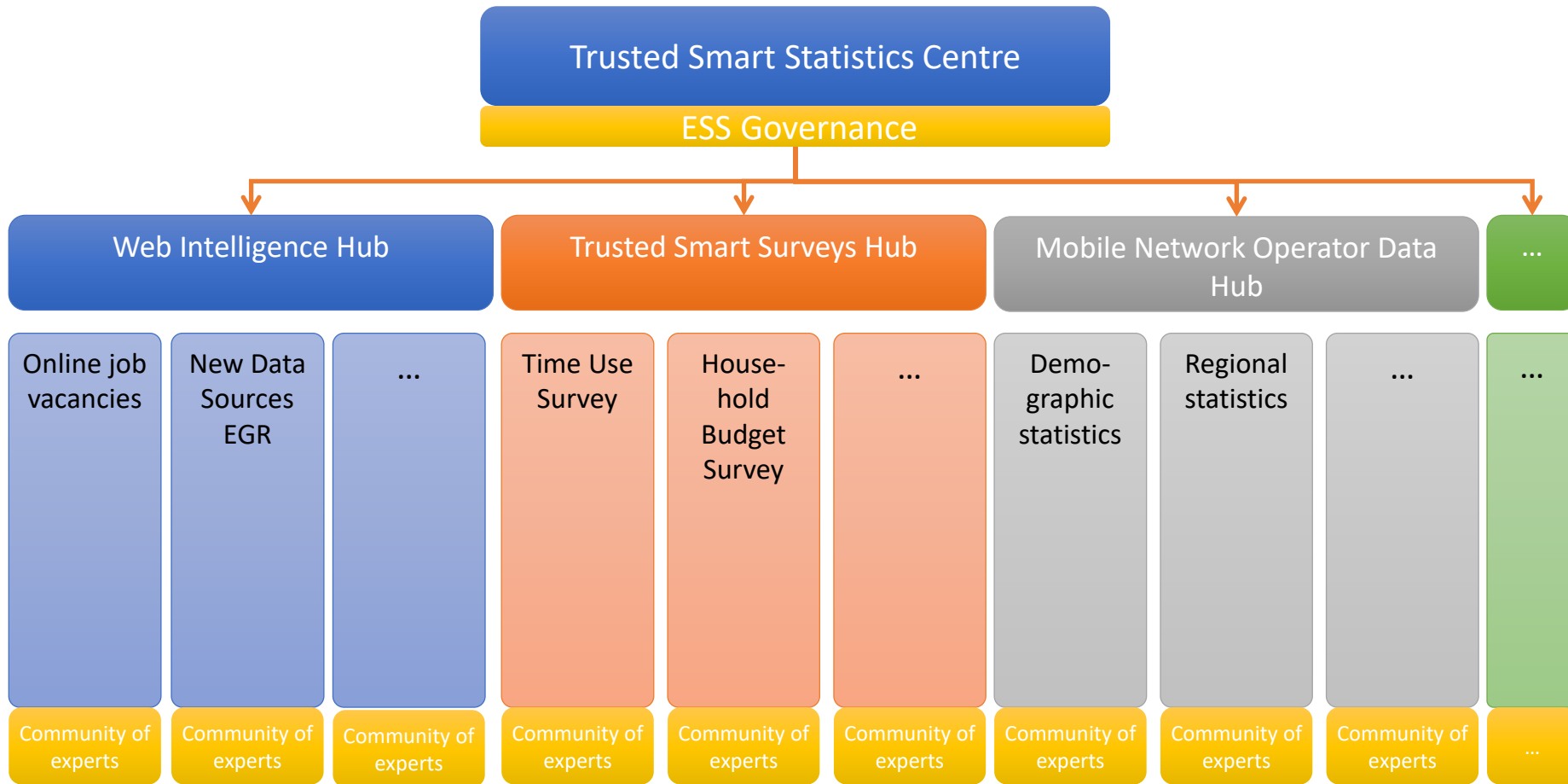
2024/09/17

Table of Contents

- Brief introduction of the Web Intelligence Hub (WIH)
- A short overview of the main web technologies
- The WIH Data Acquisition Service (DAS)
- The DAS in practice
- Q&A

The Web Intelligence Hub (WIH)

Trusted Smart Statistics



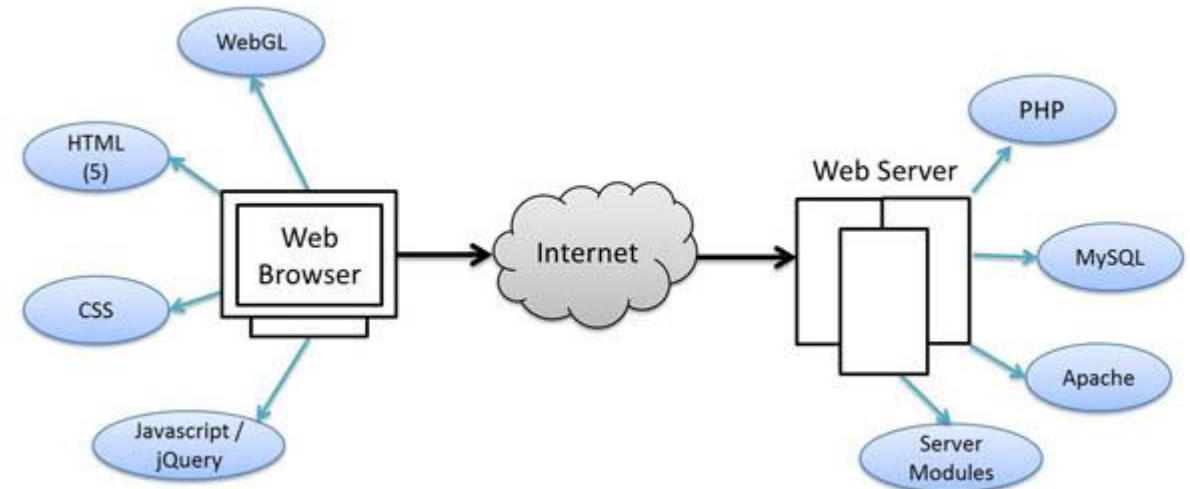
The Web Intelligence Hub (WIH)

- An initiative designed to harness web content for the development of European Statistics
- By automating the collection of content from the web to reduce the response burden on individuals and businesses
- A collaborative hub aiming to bring together statisticians, researchers, and analysts to utilize web content ethically and transparently for producing statistics
- Harmonize web scraping practices, and avoid scraping the same source multiple times
- Building facilities to collect data from the web (Data Acquisition Service) and to analyze it (Datalab)

Intro into web technologies

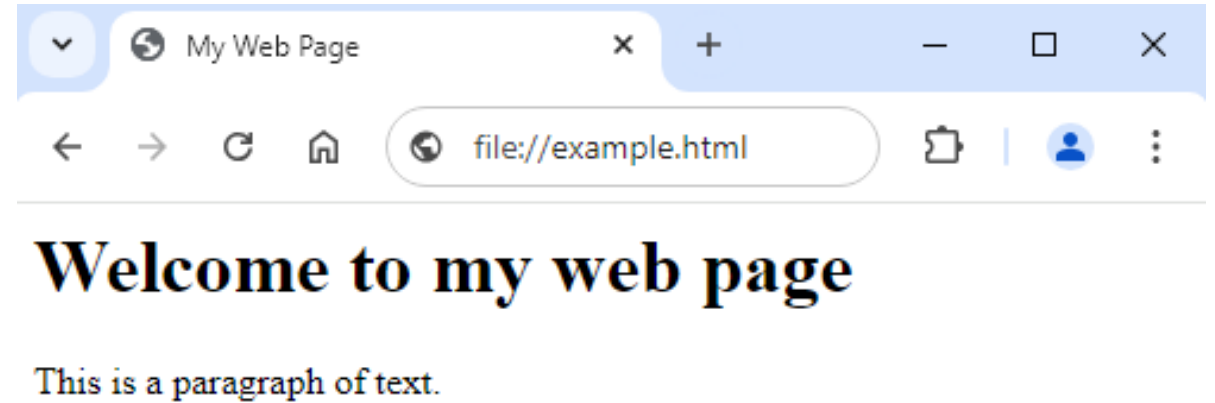
Building blocks of the Web

- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheets)
- JS (JavaScript)
- Server-side technologies (e.g. PHP, Ruby, Python)
- Databases (e.g. PostgreSQL, MySQL)



HTML (Hypertext Markup Language)

- Used for defining the structure and content of web pages, e.g. headings, paragraphs, images, links, forms, tables, etc.
- Provides the basic structure of a web page
- HTML5 introduced new features like video, audio, and canvas elements
- Example:



```

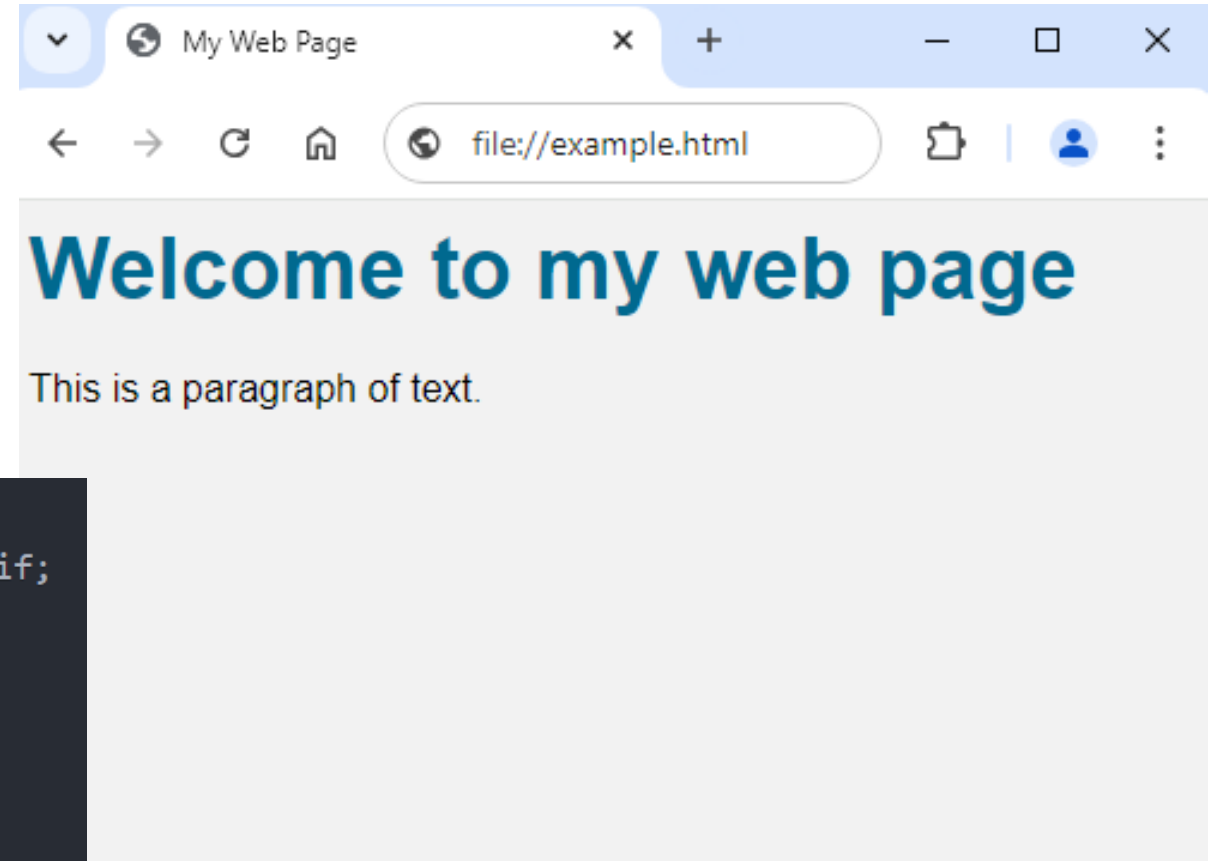
<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>
    <h1>Welcome to my web page</h1>
    <p>This is a paragraph of text.</p>
  </body>
</html>

```


CSS (Cascading Style Sheets)

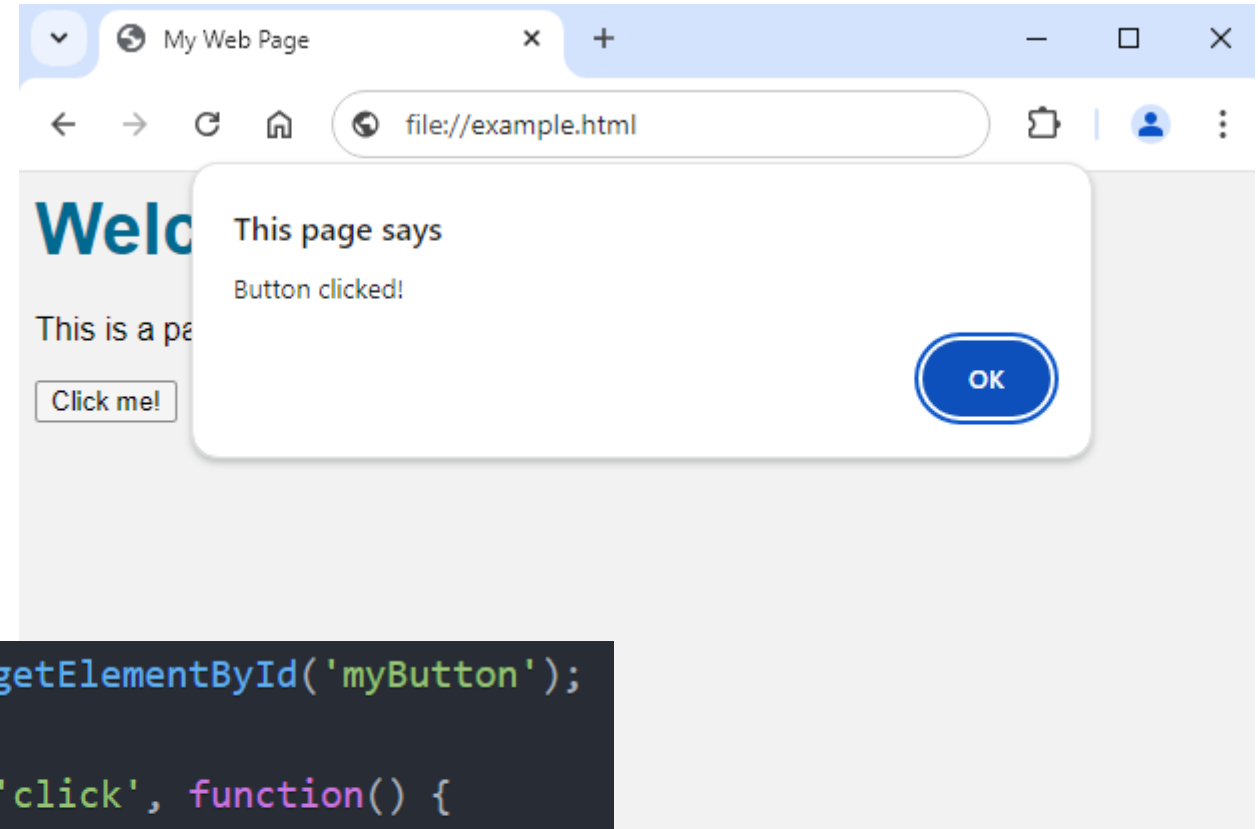
- Used for controlling the layout, typography, and visual styling of web pages by defining colors, fonts, spacing, borders, and other visual styles
- Allows for responsive design and media queries
- Example:

```
body {  
  font-family: Arial, sans-serif;  
  background-color: #f2f2f2;  
}  
  
h1 {  
  color: #00698f;  
  font-size: 36px;  
  margin-bottom: 20px;  
}
```



JS (JavaScript)

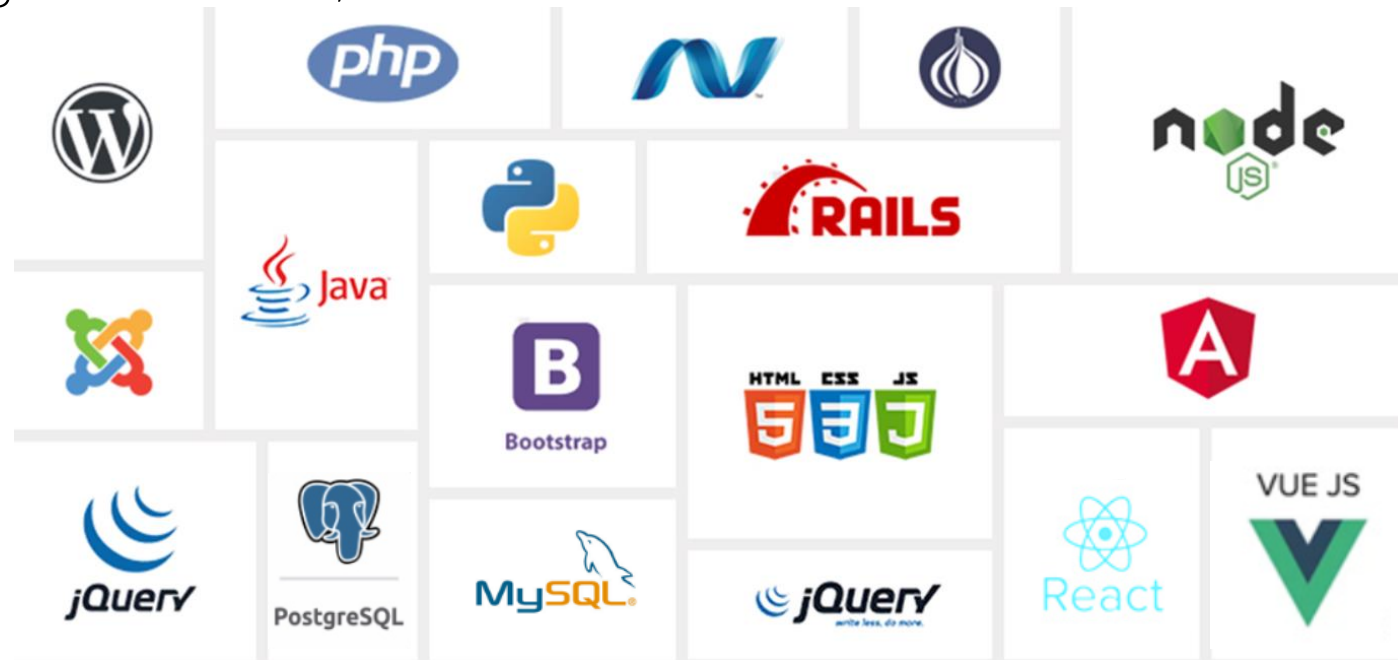
- Used for adding interactivity, animations, and dynamic effects to web pages, web and mobile applications
- Allows for Document Object Model (DOM) manipulation, event handling, and API interactions
- JavaScript frameworks like React, Angular, and Vue.js can simplify development
- Example:



```
const button = document.getElementById('myButton');  
  
button.addEventListener('click', function() {  
  alert('Button clicked!');  
});
```

Other Website Technologies

- Server-side technologies: PHP, Ruby, Python, Node.js
- Databases: MySQL, MongoDB, PostgreSQL, SQLite
- Other frameworks and technologies: WordPress, Drupal, Joomla, Shopify



Tools to collect data from the web

Some tools to scrape the internet

- BeautifulSoup
- Scrapy
- Selenium
- Playwright
- StormCrawler



BeautifulSoup

BeautifulSoup

- Parses HTML and XML documents
- Allows you to navigate and search through the parse tree
- Supports various parser libraries, including lxml and html5lib

```
from bs4 import BeautifulSoup

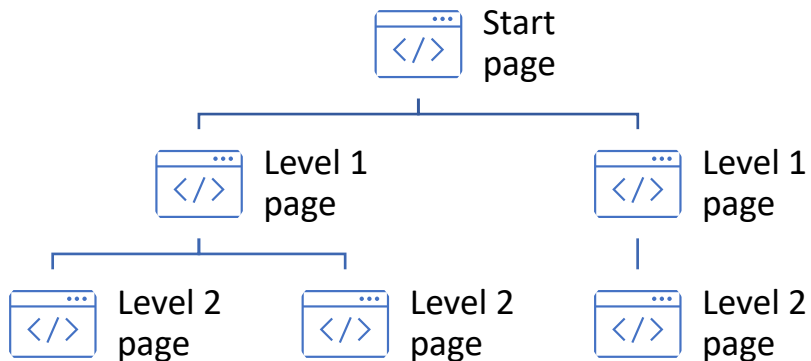
url = "https://www.example.com"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

# Extract data from the page
data = soup.find_all("div", class_="data")

# Print the extracted data
for item in data:
    print(item.text)
```

Scrapy

- Handles common web scraping tasks, such as handling different types of content and storing data
- Supports various data storage formats, including CSV, JSON, and XML
- Allows you to define data pipelines and processing logic using CSS and XPath selectors



```
import scrapy

class ExampleSpider(scrapy.Spider):
    name = "example"
    start_urls = ["https://www.example.com"]

    def parse(self, response):
        data = response.css("div.data::text").get()
        yield {"data": data}
```

Selenium

- Supports various web browsers, including Chrome, Firefox, Safari, Microsoft Explorer/Edge, Opera and more
- Allows you to automate common web browser tasks, such as filling out forms and clicking buttons
- Supports JavaScript rendering and dynamic content

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.get("https://www.example.com")

# Extract data from the page
data = driver.find_element_by_css_selector("div.data").text

# Print the extracted data
print(data)
```


Playwright

- Supports 3 main web browsers: Chrome, Firefox, and Safari
- Allows you to automate common web browser tasks, such as filling out forms and clicking buttons
- Supports headless browsing and JavaScript rendering
- Built-in trace viewer and test reporting

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    browser = p.chromium.launch(headless=True)
    context = browser.new_context()
    page = context.new_page()

    page.goto("https://www.example.com")

    # Extract data from the page
    data = page.query_selector("div.data").text_content()

    # Print the extracted data
    print(data)
```

StormCrawler

- Built on top of Apache Storm, allowing it to scale horizontally and handle large volumes of data
- Easy to extend to parse various document formats with Apache Tika
- Resilient, low latency, polite yet efficient
- Used in large scale FP7 EU research project: Open Web Search – ows.eu



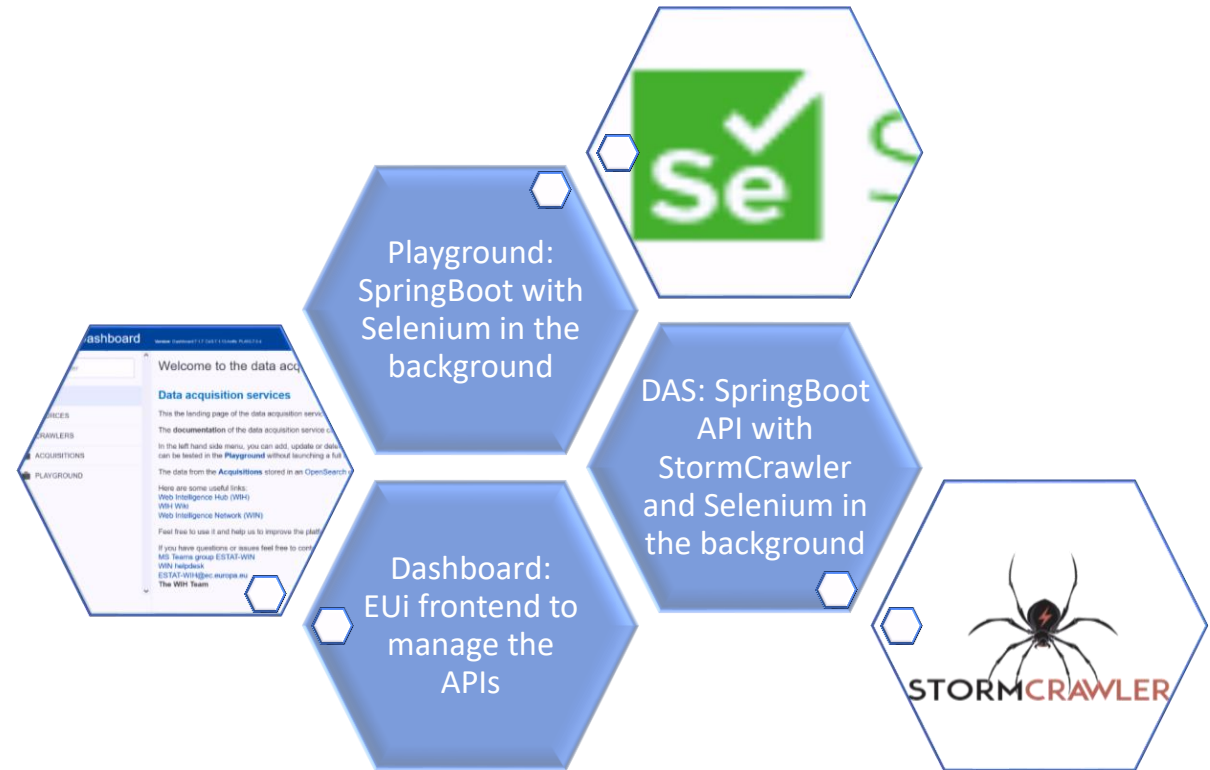
APACHE
STORM[™]

Distributed • Resilient • Real-time

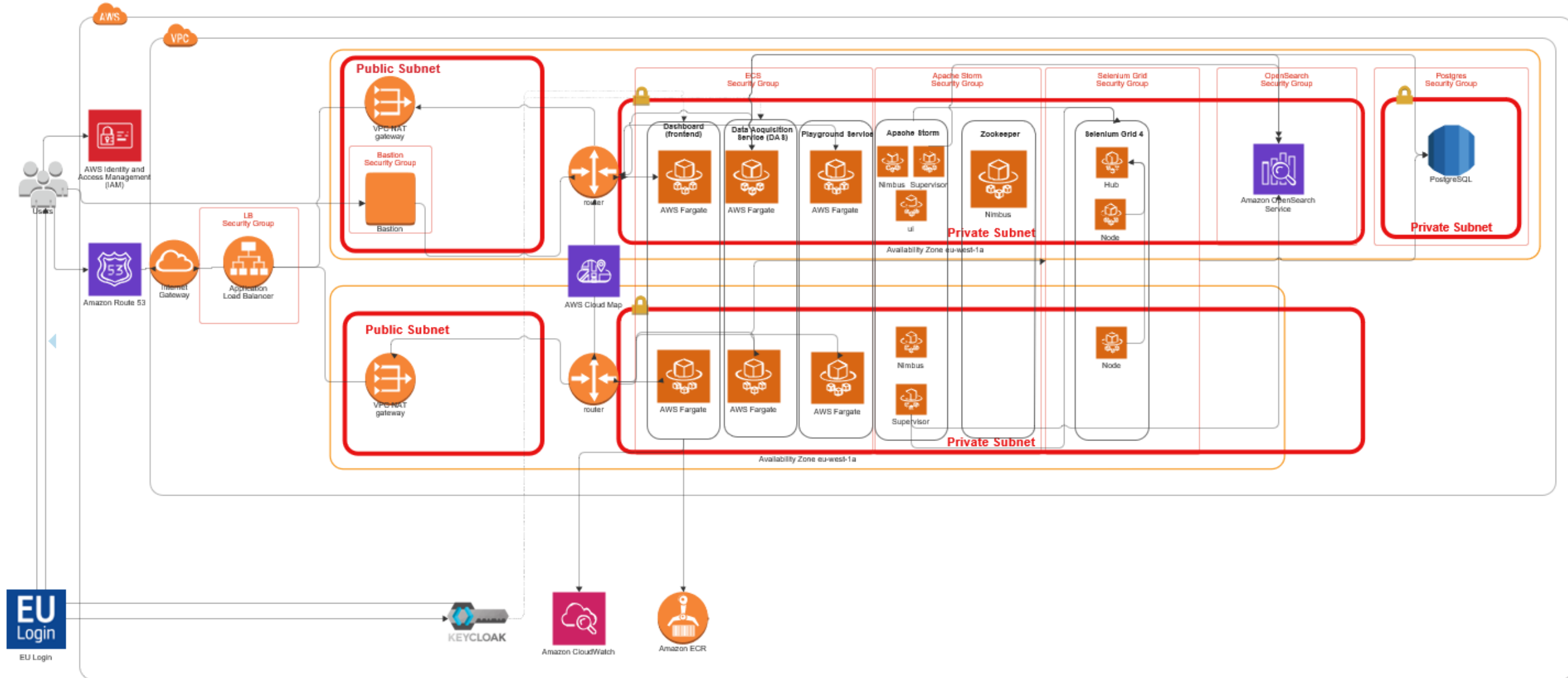
WIH Data Acquisition Service (DAS)

DAS development principles

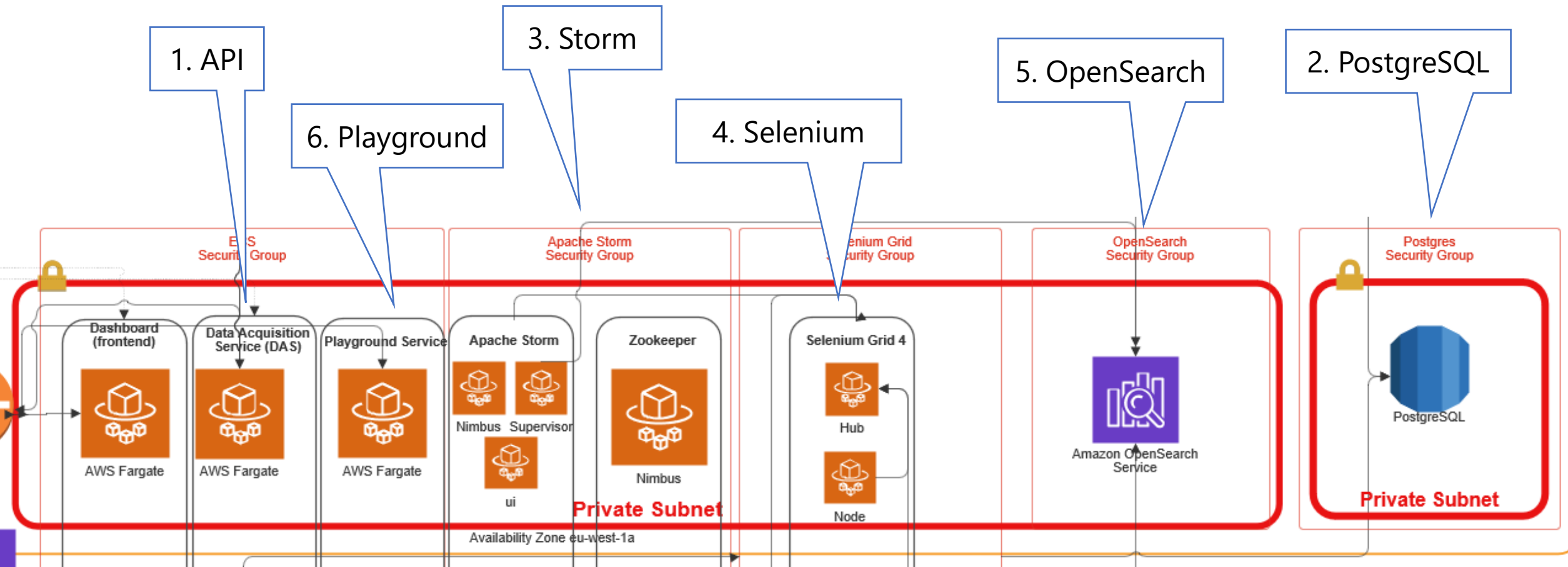
- Scalable
- Build on open-source tools
- Try to use the state of the art
- Can handle static and dynamic content
- No coding only configuration
- Universal, can be used for several use cases (OJA, MNE, price, etc.)
- Separation of use cases with possible collaboration in the same use case



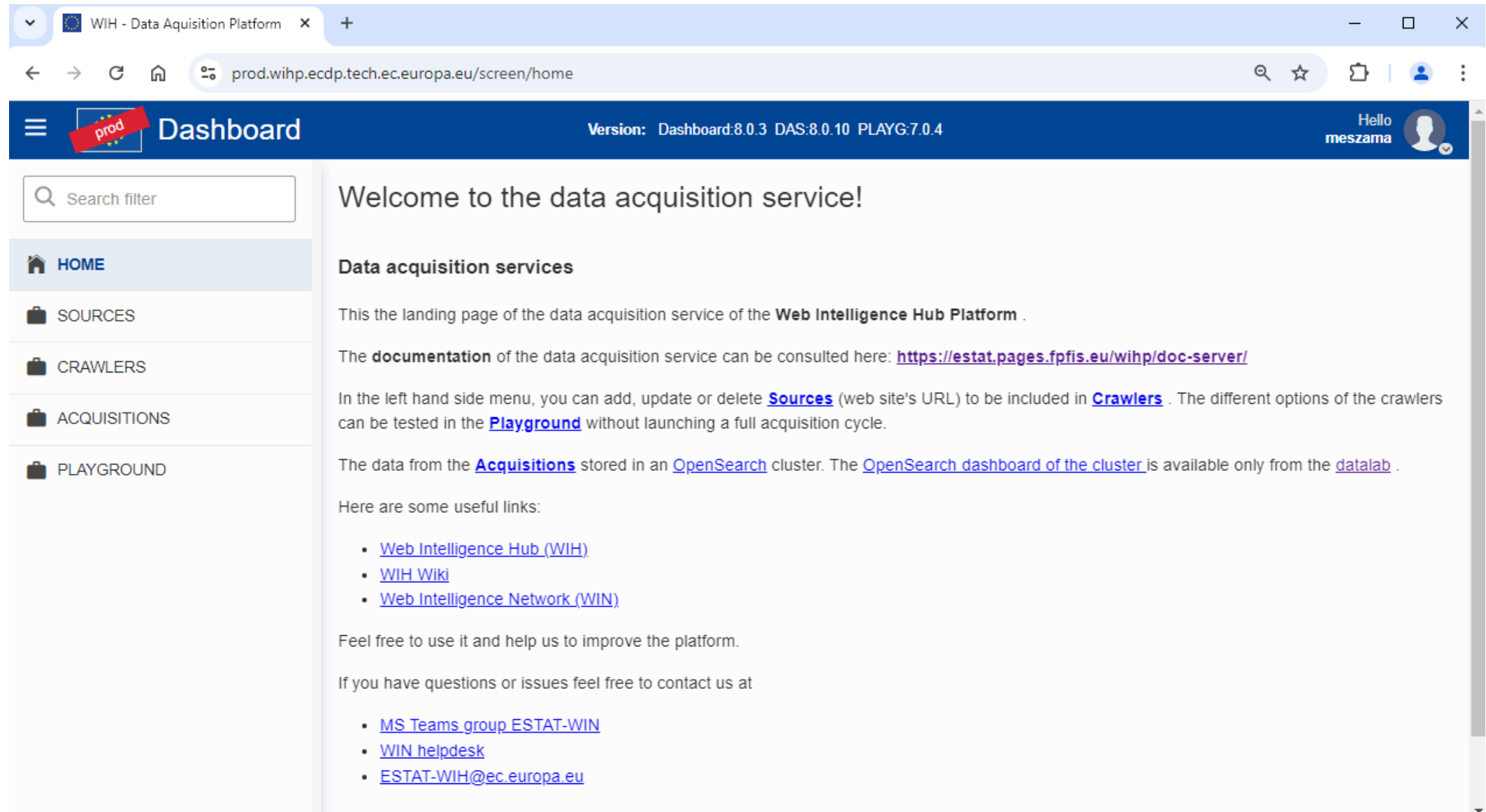
DAS infrastructure overview



DAS backend in details



DAS frontend - Dashboard



The screenshot shows a web browser window with the URL `prod.wihp.ecdp.tech.ec.europa.eu/screen/home`. The dashboard header is dark blue and contains a search filter, the word "Dashboard", version information (Dashboard:8.0.3 DAS:8.0.10 PLAYG:7.0.4), and a user profile for "meszama". The left sidebar lists navigation options: HOME, SOURCES, CRAWLERS, ACQUISITIONS, and PLAYGROUND. The main content area features a welcome message and detailed instructions on using the data acquisition service, including links to documentation, playground, and useful resources.

Search filter

Dashboard Version: Dashboard:8.0.3 DAS:8.0.10 PLAYG:7.0.4 Hello meszama

HOME
SOURCES
CRAWLERS
ACQUISITIONS
PLAYGROUND

Welcome to the data acquisition service!

Data acquisition services

This is the landing page of the data acquisition service of the **Web Intelligence Hub Platform**.

The **documentation** of the data acquisition service can be consulted here: <https://estat.pages.fpfis.eu/wihp/doc-server/>

In the left hand side menu, you can add, update or delete **Sources** (web site's URL) to be included in **Crawlers**. The different options of the crawlers can be tested in the **Playground** without launching a full acquisition cycle.

The data from the **Acquisitions** stored in an **OpenSearch** cluster. The [OpenSearch dashboard of the cluster](#) is available only from the [datalab](#).

Here are some useful links:

- [Web Intelligence Hub \(WIH\)](#)
- [WIH Wiki](#)
- [Web Intelligence Network \(WIN\)](#)

Feel free to use it and help us to improve the platform.

If you have questions or issues feel free to contact us at

- [MS Teams group ESTAT-WIN](#)
- [WIN helpdesk](#)
- ESTAT-WIH@ec.europa.eu

DAS in action

Demo I.

- Adding source Eurostat <https://ec.europa.eu/eurostat> to the DEMO group/use case
- Create Crawler
- Start acquisition
- Adding emit outlinks
- Adding max depth filter
- Adding domain filter
- Adding host filter
- Adding regex filter data
- Show how the URLs are changing

Demo II.

Static vs dynamic

- Add source
https://www.selenium.dev/selenium/web/document_write_in_onload.html
- Start crawler as static
- Start as dynamic
- Compare content

Demo III.

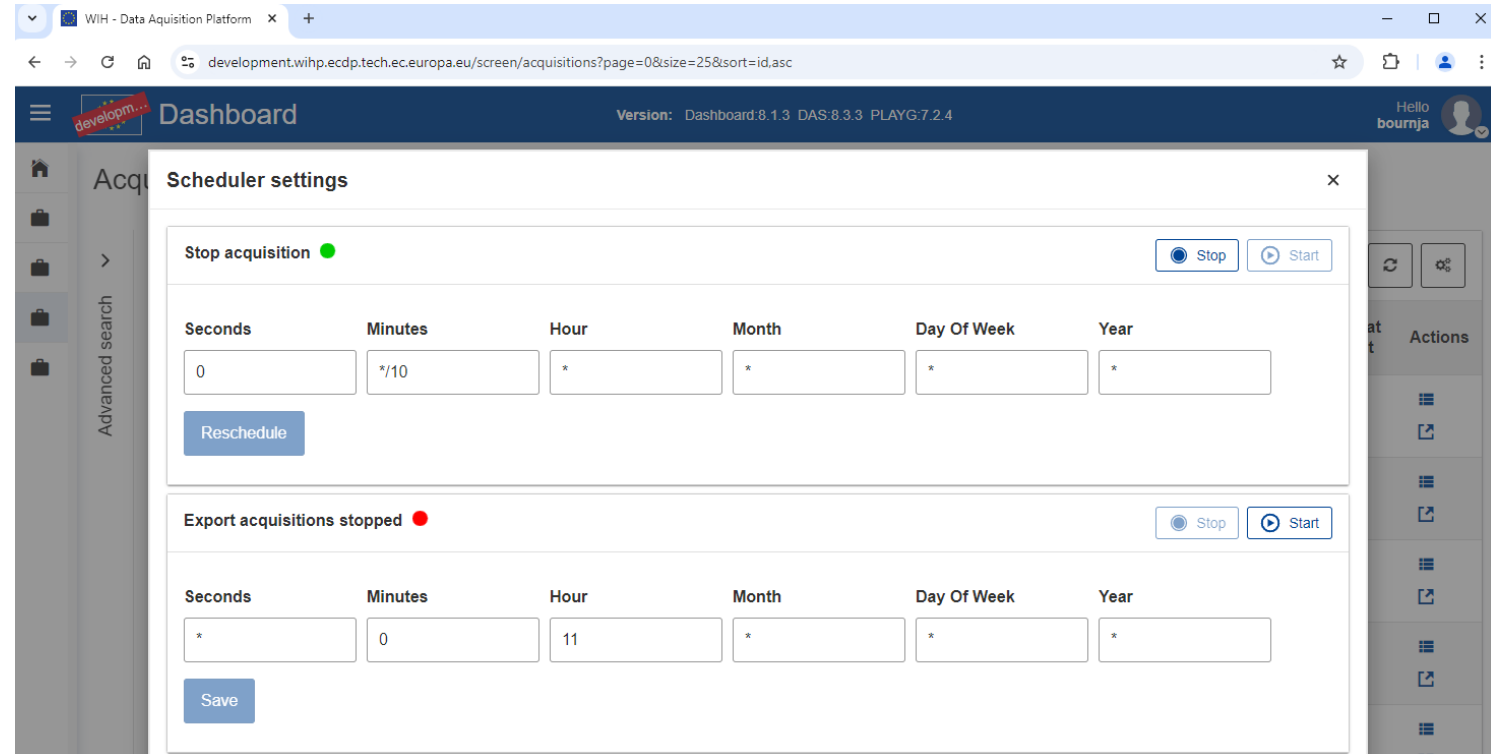
OJA use case

- Create KSH crawler
- Schedule the acquisition
- Show PDF extraction
- Create GUS crawler
- Adding parse filter

Upcoming new features

DAS new features in upcoming releases

- Automatic sizing of resources
- Regular scheduling of acquisitions
- Incremental crawling
- Automatic stopping browsers when no new URL found
- Moving stopped acquisitions to S3 storage and access data through API



Thank you for your attention!

Comments/Questions?