# Projektowanie i Integracja Systemów

# Projekt Etap 3

Wojciech Sekuła, Patryk Stachyra,

Anton Basan, Piotr Grabowski

Prowadzący:

Kamil Żbikowski

## Link do repozytorium:

https://github.com/Pecet13/23Z-PIS-Process-Mining

## Lista wymagań funkcjonalnych

**Iteracja 1:**

- Autoryzacja i uwierzytelnianie użytkownika: Logowanie do baz danych poprzedzone uwierzytelnianiem użytkownika
- Import i integracja danych: Aplikacja umożliwia pobieranie i przetwarzanie danych z plików .csv
- Wyświetlanie surowych danych procesu w postaci tabel
- Wizualizacja procesu w postaci grafu Directly-Follows

**Iteracja 2:**

- Generowanie aktualnych statystyk zawierające między innymi średnie, minimalne i maksymalne czasy trwania aktywności
- Generowanie wykresu ukazującego czas trwania kolejnych aktywności
- Filtracja wyników względem czasu ich występowania
- Wizualizacja danych umożliwiająca porównanie występowania poszczególnych aktywności
- Raportowanie: Generowanie i eksport szczegółowych raportów dotyczących wydajności procesu

## Podział pracy

Wojciech Sekuła - Backend aplikacji zawierającym moduł ładowania danych w trybie batch z bazy danych oraz moduł ładowania danych w trybie online poprzez podłączenie danych w formie online poprzez kolejkę komunikatów (Kafka), dokumentacja

Patryk Stachyra - DevOps: zarządzanie zadaniami za pomocą Jira, wdrażanie aplikacji (Azure), dokumentacja

Anton Basan - Testowanie poprawności działania aplikacji za pomocą biblioteki unittest, dokumentacja

Piotr Grabowski - Frontend aplikacji zawierający wyświetlanie grafu directly-follows graph oraz statystyki dotyczące procesu, a także logika analizy i przetwarzania danych, moduł CI/CD (Jenkins), dokumentacja

## Wybrany stos technologiczny

Moduł ładowania danych w trybie batch:
Język programowania: Python
Biblioteki: Pandas dla operacji na danych, SQLAlchemy dla połączenia z bazami danych.
Baza danych: MySQL

Moduł ładowania danych w trybie online:
System kolejkowania: Apache Kafka

Moduł analizy danych z UI:
Backend: Python
Frontend: Flask
Biblioteka do wizualizacji danych: Chart.js

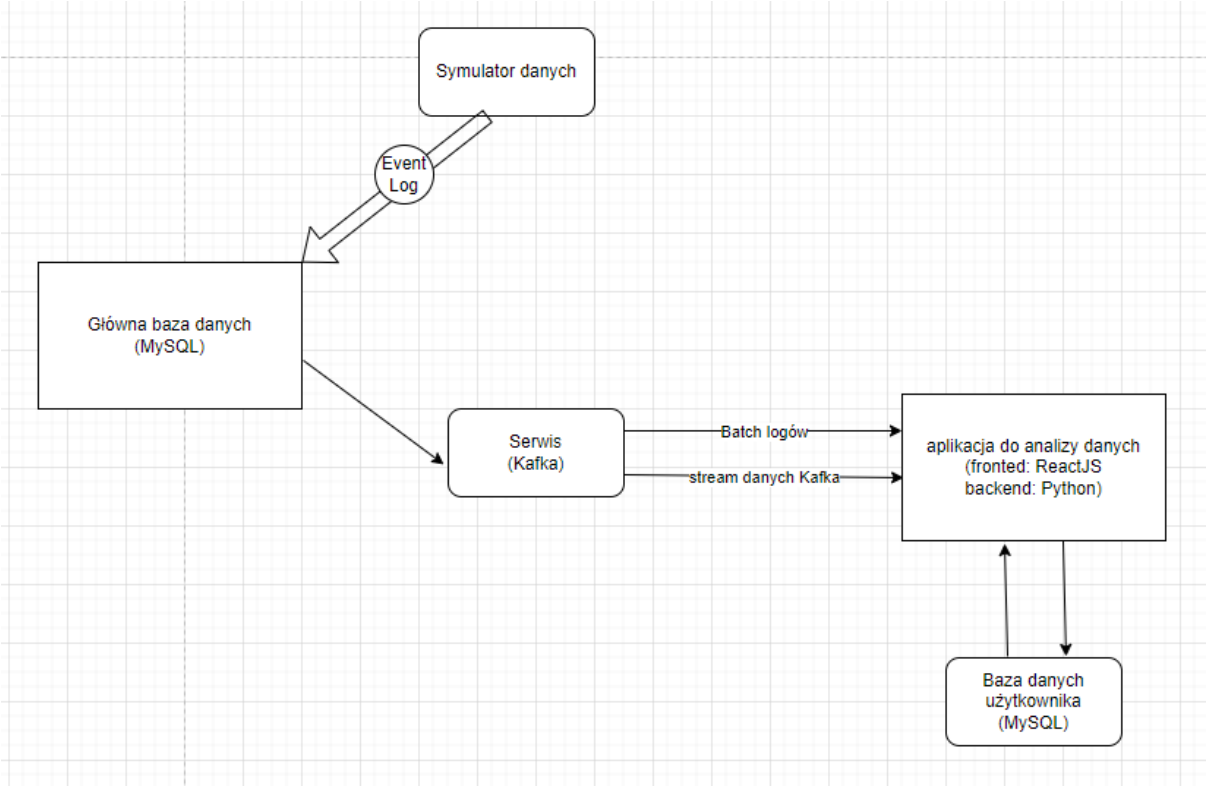
Środowisko wdrożeniowe: Microsoft Azure

CI/CD: Jenkins

Zarządzanie zależnościami: pip (Python).
Wersjonowanie kodu:
Git z integracją GitHub.

Zarządzanie projektem:
JIRA

**Wstępny projekt / architektura**




Stworzony projekt system składa się z symulatora danych generującego logi zdarzeń, które są ładowane do głównej bazy danych MySQL. Dane te są przetwarzane w trybie batch przez usługę Kafka i w trybie strumieniowym, co na elastyczność w obsłudze danych. Kafka służy także do przetwarzania logów batchowych. System zawiera również aplikację do analizy danych z frontendem w ReactJS i backendem w Pythonie, która umożliwia analizę i prezentację danych zebranych z głównej bazy danych. Istnieje również oddzielna baza danych użytkownika MySQL, która służy do przechowywania danych specyficznych dla warstwy logiki biznesowej. Architektura rozdziela warstwy logiki biznesowej od danych procesów.

**Materiały szkoleniowe**

- JavaScript:
  - https://www.javascripttutorial.net
  - https://youtu.be/PkZNo7MFNFg?si=SM63CnKqfR0iZcT6
  - https://www.chartjs.org/docs/latest/
- Python:
  - https://docs.python.org/3/tutorial/index.html
  - https://flask.palletsprojects.com/en/3.0.x/
- MySQL:
  - https://youtu.be/7S_tz1z_5bA?si=Da6MHrlSziHvFyEC
  - https://www.mysqltutorial.org
- Jenkins:
  - https://youtu.be/FX322RVNGj4?si=t69Ah9lUGRl_MNp8
- Kafka:
  - https://kafka.apache.org/quickstart