# CUSTOMER RELATIONSHIP MANAGEMENT – system zarządzania relacjami z klientami


**Uniwersytet w Siedlcach**

Instytut Informatyki

Serhii Humennyi

Nr albomu: 88455

Informatyka III rok


2024/2025

Prowadzący : Stanisław Ambroszkiewicz

# 1. Wprowadzenie

Celem pracy jest zaprojektowanie i stworzenie funkcjonalnego systemu CRM (Customer Relationship Management), umożliwiającego małym i średnim przedsiębiorstwom zarządzanie relacjami z klientami oraz procesami związanymi z obsługą klienta.

Rozwiązanie ma zostać zaimplementowane w technologii Spring Boot, z wykorzystaniem konteneryzacji i orkiestracji w środowisku Kubernetes, co zapewni skalowalność, niezawodność oraz nowoczesny sposób wdrażania aplikacji.

---

# 2. Uzasadnienie wyboru tematu

Dynamiczny rozwój firm oraz rosnąca konkurencja na rynku powodują konieczność efektywnego zarządzania relacjami z klientami.
Wiele przedsiębiorstw poszukuje systemów CRM dostosowanych do własnych procesów, które jednocześnie są tanie w utrzymaniu i skalowalne.

Technologie Spring Boot oraz Kubernetes należą do najpopularniejszych narzędzi wykorzystywanych we współczesnej inżynierii oprogramowania.
Praca łączy aspekty aplikacyjne (funkcjonalności CRM, GUI) z praktycznymi rozwiązaniami DevOps (Docker, Kubernetes), co czyni ją wartościową zarówno edukacyjnie, jak i zawodowo.

---

# 3. Cel pracy

Głównym celem pracy inżynierskiej jest zaprojektowanie, zaimplementowanie i wdrożenie kompletnego systemu CRM, który umożliwi:

- zarządzanie danymi klientów,
- obsługę ofert handlowych,
- zarządzanie zadaniami i aktywnościami,
- logowanie oraz kontrolę dostępu użytkowników,
- prezentację danych w formie czytelnego GUI,
- uruchamianie aplikacji w środowisku kontenerowym (Docker + Kubernetes),
- zapewnienie skalowalności i niezawodności działania.

Wynikiem pracy będzie działająca aplikacja wraz z dokumentacją techniczną i opisową.

---

# 4. Zakres pracy

### 4.1. Analiza problemu i wymagań

- analiza potrzeb biznesowych systemów CRM
- identyfikacja głównych modułów funkcjonalnych
- tworzenie specyfikacji wymagań funkcjonalnych i niefunkcjonalnych

### 4.2. Projekt systemu

- projekt architektury aplikacji opartej o Spring Boot
- model bazy danych
- projekt REST API
- projekt GUI aplikacji (np. w technologii Thymeleaf lub SPA)
- projekt wdrożenia na środowisku Kubernetes

### 4.3. Implementacja funkcjonalności CRM

- moduły: Klienci, Oferty, Zadania i aktywności, Autoryzacja,
- GUI: formularze, tabele, widoki szczegółowe, walidacje,
- migracje danych (Liquibase),
- konteneryzacja aplikacji (Docker),
- wdrożenie na klastrze Kubernetes.

### 4.4. Warstwa prezentacji (GUI)

- interfejs użytkownika dla pracowników firmy
- widoki list, formularzy i szczegółowych danych
- walidacja danych
- ergonomia i przejrzystość obsługi

### 4.5. Integracja i wdrożenie

- konteneryzacja aplikacji (Docker)
- przygotowanie manifestów Kubernetes (Deployment, Service, ConfigMap, Secrets)
- wdrożenie aplikacji na lokalnym lub chmurowym klastrze Kubernetes
- testy poprawnego działania

---

# 5. Zakres funkcjonalny

## Moduł Klienci

- dodawanie, edycja, usuwanie klientów,
- przegląd listy klientów,
- filtrowanie i wyszukiwanie,
- historia kontaktu.

## Moduł Oferty

- tworzenie i edycja ofert,
- przypisywanie ofert klientom,
- zmiana statusów ofert.

## Moduł Zadania i Aktywności

- rejestracja zadań i aktywności,
- przypisywanie aktywności do klientów,
- oznaczanie statusów realizacji.

## Moduł Logowania i Autoryzacji

- logowanie użytkowników,
- role: administrator, pracownik, kierownik,
- kontrola dostępu do funkcji systemu.

## GUI

- widoki tabelaryczne, formularze,
- walidacja danych,
- responsywny układ.

---

# 6. Zakres niefunkcjonalny

## Wydajność

- szybkie działanie API oraz warstwy GUI,
- optymalne zapytania do bazy danych.

### Skalowalność

- wdrożenie w klastrze Kubernetes,
- obsługa replikacji i balansowania ruchu.

### Bezpieczeństwo

- Spring Security, szyfrowanie haseł,
- ochrona danych użytkowników.

### Niezawodność

- odporność na restart kontenerów,
- automatyczny restart aplikacji w Kubernetes.

### Utrzymywalność

- przejrzysta architektura,
- migracje bazy (Liquibase),
- wersjonowanie w GitHub,
- możliwość łatwego wdrażania aktualizacji.

---

## 7. Planowane technologie i narzędzia

### Backend

- Spring Boot
- Spring Web
- Spring Data JPA
- Spring Security
- Liquibase
- MySQL (prod) / H2 (dev)

### Frontend / GUI

- Thymeleaf

### DevOps

- Docker
- Kubernetes (minikube / k3d / k3s / kind)
- GitHub + GitHub Actions

---

# 8. Charakterystyka użytkowników / aktorów

## Administrator

- zarządza użytkownikami systemu,
- kontroluje role i uprawnienia,
- nadzoruje poprawność danych.

## Pracownik

- obsługuje klientów, tworzy oferty i zadania,
- korzysta z GUI w codziennej pracy.

## Kierownik

- analizuje wyniki pracy zespołu,
- generuje zestawienia i raporty,
- podejmuje decyzje biznesowe.

---

# 9. Metody realizacji pracy

Analiza wymagań i projekt systemu – przygotowanie modeli UML, opisów przypadków użycia, diagramów architektury.

Implementacja backendu – stworzenie API oraz logiki biznesowej.

Projekt i implementacja GUI – widoki list, formularzy oraz stron szczegółowych.

Testy funkcjonalne – weryfikacja poprawności działania systemu.

Konteneryzacja – przygotowanie obrazu Docker.

Orkiestracja – wdrożenie do Kubernetes i testy wysokiej dostępności.

Dokumentacja – opis architektury, API, procesu wdrażania i instrukcji użytkowania.

## 10. Harmonogram

| Etap | Zadania | Termin |
|---|---|---|
| 1 | Analiza wymagań i przypadki użycia | Tydzień 1–2 |
| 2 | Projekt architektury systemu i bazy danych | Tydzień 3–4 |
| 3 | Implementacja backendu | Tydzień 5–8 |
| 4 | Implementacja GUI | Tydzień 9–11 |
| 5 | Konteneryzacja i Kubernetes | Tydzień 12–13 |
| 6 | Testy systemu | Tydzień 14 |
| 7 | Dokumentacja końcowa | Tydzień 15 |

## 11. Bibliografia wstępna


[1] C. Walls, *Spring in Action*. Manning, 2018.
[2] S. Newman, *Building Microservices*. O'Reilly, 2021.
[3] Kubernetes Documentation, *kubernetes.io*, dostęp: 2025.
[4] Spring Boot Documentation, *spring.io*, dostęp: 2025.
[5] Liquibase Documentation, *liquibase.org*, dostęp: 2025.
[6] J. Turnbull, *The Docker Book*, 2014.
[7] B. Burns et al., "Borg, Omega, and Kubernetes," *Communications of the ACM*, 2016.
[8] MySQL Documentation, *mysql.com*, dostęp: 2025.
[9] OWASP Foundation, *Security Guidelines*, 2025.
[10] G. Kim, *The Phoenix Project*, IT Revolution, 2013.