

PRACA DYPLOMOWA

WYDZIAŁ
BUDOWY MASZYN I INFORMATYKI

KIERUNEK: Informatyka

SPECJALNOŚĆ: TECHNIKI TWORZENIA OPROGRAMOWANIA

Maciej Tonderski

nr albumu: 62572

Praca magisterska

PLATFORMA DO AUTOMATYCZNEGO
WYKRYWANIA I NAPRAWY
PODATNOŚCI W KONTENERACH
DOCKEROWYCH

Kategoria pracy: projektowa

Promotor: dr inż. RUSLAN SHEVCHUK

Bielsko-Biała, 2025

Streszczenie

Tutaj umieść streszczenie swojej pracy magisterskiej.

Contents

1	Wprowadzenie	3
1.1	Cel pracy	3
1.2	Zakres pracy	3
2	Czym jest HomeLab oraz analiza istniejących rozwiązań	5
2.1	Definicja HomeLab oraz znaczenie	5
2.2	Technologie wykorzystywane w homelabach	5
2.3	Analiza istniejących systemów do zarządzania homelabem	6
3	Projekt Systemy Homelab	9
3.1	Wymagania funkcjonalne i нефункционалне	9
3.2	Architektura systemu	11
3.3	Technologie i narzędzia użyte w systemie	11
4	Implementacja systemu	14
4.1	Backend - API do zarządzania systemem	14
4.2	Frontend - Interfejs użytkownika	14
4.3	Automatyzacja Konfiguracji i wdrożenie	14
5	Testowanie i analiza systemu	18
5.1	Testy jednostkowe i integracyjne	18
5.2	Testy wydajnościowe i bezpieczeństwa	18
5.3	Opinie użytkowników i ewaluacja systemu	18
6	Podsumowanie i wnioski	19
6.1	Osiągnięcia i rezultaty pracy	19
6.2	Możliwości dalszego rozwoju	19

1. Wprowadzenie

Współczesna technologia informatyczna umoliwia pasjonatom IT, administratorom systemów oraz programistom budowanie i zarządzanie własnymi środowiskami testowymi oraz produkcyjnymi w domowych warunkach. Koncepcja HomeLab czyli prywatnego środowiska IT, zyskuje na popularności dzięki coraz szerszemu dostępowi do wydajnego sprzętu, technologii wirtualizacji oraz narzędzi do automatyzacji zarządzania infrastrukturą. Jednak dla wielu użytkowników proces konfiguracji i utrzymania takiego środowiska może być skomplikowany i czasochłonny.

1.1 Cel pracy

Celem niniejszej pracy magisterskiej jest zaprojektowanie i implementacja systemu HomeLab, który uprości proces budowy, konfiguracji oraz zarządzania własną infrastrukturą IT. System ten ma zapewnić użytkownikom intuicyjne narzędzia do zarządzania serwerami, maszynami wirtualnymi, kontenerami oraz sieci, a także umożliwić zdalny bezpieczny dostęp do zasobów. Kluczowym założeniem projektu jest maksymalna automatyzacja procesów, co pozwoli na minimalizację konieczności manualnej konfiguracji i zwiększy wygodę użytkownika.

1.2 Zakres pracy

W pracy zostaną omówione istotne aspekty techniczne związane z budową homelab, w tym wybór odpowiednich technologii, metod zarządzania infrastrukturą oraz zapewnienia jej bezpieczeństwa. Ponadto przedstawiona zostanie analiza istniejących rozwiązań oraz uzasadnienie wyboru implementowanych funkcjonalności. Efektem końcowym pracy będzie gotowy system, który może zostać wdrożony przez użytkowników chcących stworzyć własne homelab w sposób szybki i efektywny.

Niniejsza praca stanowi przyczynek do rozwoju narzędzi dedykowanych osobom zainteresowanym budową i zarządzaniem własnym środowiskiem IT, oferując innowacyjne podejście do automatyzacji i ułatwienia dostępu do homelab. W kolejnych rozdziałach zostaną szczegółowo omówione wszystkie kluczowe elementy systemu oraz proces

jego implementacji.

2. Czym jest HomeLab oraz analiza istniejących rozwiązań

2.1 Definicja HomeLab oraz znaczenie

Homelab jest prywatnym środowiskiem IT, dzięki któremu entuzjaści nowych technologii, administratorzy systemów oraz programiści mogą w lokalnym - domowym środowisku testować, rozwijać oraz zarządzać własną infrastrukturą IT. Jego głównym zamierzeniem jest stworzenie realistycznego środowiska do eksperymentowania z technologiami cloudowymi, wirtualizacją, konteneryzacją oraz narzędziami DevOps. Własny system Homelab to również metoda na rezygnację z komercyjnych subskrypcji, takich jak Google Drive, DropBox czy OneDrive, co pozwala na w pełni kontrolowanie kto ma dostęp do naszych prywatnych danych. Dzięki niemu zwiększa się prywatność poprzez wyeliminowanie potrzeby przechowywania zdjęć w usługach chmurowych takich jak Google Photos. Homelaby znajdują zastosowanie w wielu obszarach, w tym:

- nauka administracji serwerami i sieciami,
- testowaniu nowych technologii przed użyciem jej w środowisku produkcyjnym,
- budowaniu prywatnej chmury oraz rozwiązań do przechowywania danych,
- analizie bezpieczeństwa i przeprowadzaniu testów penetracyjnych,
- tworzenie autoamtyzacji dla infrastruktury IT,
- uniezależnieniu się od komercyjnych dostawców chmury w celu zwiększenia kontroli nad własnymi danymi.

2.2 Technologie wykorzystywane w homelabach

Homelab może składać się z różnych komponentów, od dedykowanych serwerów fizycznych po rozwiązania chmurowe i kontenerowe. Kluczowe technologie wykorzystywane w homelabach obejmują:

2.2.1 Wirtualizacja i konteneryzacja

- Proxmox VE - platforma do zarządzania maszynami wirtualnymi i kontenerami.
- VMware ESXi - profesjonalne narzędzie do wirtualizacji serwerów.
- Hyper-V - narzędzie do wirtualizacji dostarczane przez Microsoft wraz z systemem Windows.
- Docker i Kubernetes - technologie konteneryzacji, pozwalające na elastyczne zarządzanie aplikacjami i zasobami.

2.2.2 Automatyzacja i zarządzanie konfiguracją

- Ansible, Terraform, Puppet, Chef - narzędzia do automatyzacji wdrażania i zarządzania infrastrukturą.

2.2.3 Monitoring i analiza

- Prometheus i grafana - rozwiązanie do monitorowania wydajności i wizualizacji danych.
- Zabbix - platforma do monitorowania infrastruktury IT.

2.3 Analiza istniejących systemów do zarządzania homelabem

2.3.1 Przegląd dostępnych rozwiązań

Na rynku istnieje kilka systemów umożliwiających zarządzanie homelabem. Do najpopularniejszych należą:

- Proxmox VE - rozbudowany, open-source rozwiązanie do zarządzania maszynami wirtualnymi i kontenerami, oferujące integrację z Ceph i wysoką dostępność.
- Unraid - popularne rozwiązanie NAS z obsługą wirtualizacji i kontenerów, cenione za łatwość obsługi ale ograniczone zastosowanie korporacyjne.
- OpenStack - potężna platforma chmurowa, która może być używana do zarządzania homelabem, ale jej skomplikowana konfiguracja sprawia, że nie jest przyjazna dla początkujących użytkowników.
- TrueNAS - rozbudowane oprogramowanie do zarządzania przestrzenią dyskową, które umożliwia tworzenie prywatnych chmur danych

- Docker + Kubernetes - stosowane w bardziej zaawansowanych wdrożeniach do zarządzania kontenerami, ale wymagające większej wiedzy technicznej.

2.3.2 Zalety i ograniczenia konkurencyjnych systemów

Proxmox VE

Zalety

- Darmowa wersja open-source.
- Wsparcie dla maszyn wirtualnych (KVM) i kontenerów (LXC).
- Możliwość tworzenia klastrów wysokiej dostępności.

Wady

- Brak pełnej automatyzacji wdrożeń.
- Stosunkowo wysoki próg wejścia dla początkujących użytkowników.

Unraid

Zalety

- Intuicyjny interfejs użytkownika.
- Łatwa obsługa pamięci masowej i kontrolerów.

Wady

- Model licencyjny oparty na opłacie jednorazowej.
- Ograniczona integracja z systemami chmurowymi.

OpenStack

Zalety

- Zaawansowane funkcje chmurowe.
- Skalowalność i modularność.

Wady

- Bardzo wysoka trudność wdrożenia.
- Wymaga dużej ilości zasobów sprzętowych.

TrueNAS

Zalety

- Silne wsparcie dla przechowywania danych.
- Wbudowana replikacja i ochrona RAID

Wady

- Skupione głównie na funkcjach NAS.
- Brak natywnego wsparcia dla maszyn wirtualnych.

Docker + Kubernetes

Zalety

- Elastyczność w zarządzaniu aplikacjami kontenerowymi.
- Łatwe skalowanie infrastruktury.

Wady

- Wymaga dużej wiedzy technicznej.
- Brak wsparcia dla maszyn wirtualnych.

2.3.3 Identyfikacja luki technologicznej

Analiza powyższego porównania dostępnych systemów pokazuje, że żadne z obecnych rozwiązań nie zapewnia jednocześnie:

- Pełnej integracji zarządzania maszynami wirtualnymi, kontenerami i przestrzenią dyskową w jednym ekosystemie.
- Prostego i intuicyjnego interfejsu dla użytkowników niebędących ekspertami w zarządzaniu infrastrukturą IT.
- Natychmiastowej automatyzacji wdrażania, bez konieczności skomplikowanej konfiguracji narzędzi DevOps.
- Wbudowanej funkcjonalności związanej z bezpieczeństwem i prywatnością, eliminującej konieczność korzystania z komercyjnych rozwiązań chmurowych.

Proponowany system HomeLab ma na celu uzupełnienie tej luki poprzez stworzenie intuicyjnego narzędzia do zarządzania domową infrastrukturą IT, które zapewni łatwą obsługę, pełną automatyzację oraz zwiększoną prywatność użytkowników.

3. Projekt Systemy Homelab

3.1 Wymagania funkcjonalne i нефunkcjonalne

3.1.1 Wymagania funkcjonalne

1. **Zarządzanie infrastrukturą** - możliwość konfiguracji i zarządzania maszynami wirtualnymi oraz kontenerami Docker.
2. **Panel administracyjny** - intuicyjny interfejs użytkownika stworzony przy pomocy systemu AppSmith, do zarządzania zasobami systemu.
3. **Baza danych** - przechowywanie informacji o konfiguracji systemu i użytkownikach w MongoDB.
4. **Bezpieczny dostęp zdalny** - integracja z Tailscale umożliwiająca dostęp z dowolnego miejsca na ziemi.
5. **Automatyzacja wdrożeń** - wsparcia dla CI/CD za pomocą GitHub Pipelines.
6. **Obsługa domeny** - integracja z DuckDNS w celu dynamicznego zarządzania domeną.
7. **Monitorowanie zasobów** - mechanizmy zbierania informacji o wykorzystaniu CPU, pamięci RAM oraz przestrzeni dyskowej.
8. **Wsparcie dla rozszerzeń** - możliwość dodawania nowych funkcji poprzez kontenery Dockera.
9. **Łatwe wdrażanie aplikacji** - opcja uruchamiania własnych usług w kontenerach bez konieczności zaawansowanej konfiguracji.
10. **Bezpieczne uwierzytelnianie i automatyzacja** - mechanizm logowania oparty na OAuth2 i zarządzanie rolami użytkowników.

3.1.2 Wymagania niefunkcjonalne

1. **Niski pobór energii** - system wdrażany na Raspberry Pi 5, co zapewni efektywność energetyczną.
2. **Wysoka dostępność** - redundancja i odporność na awarię dzięki Docker oraz Integracji z VPN.
3. **Łatwość w utrzymaniu** - system powinien umożliwiać łatwe aktualizację i rekonfigurację w razie potrzeby ręcznej interwencji.
4. **Skalowalność** - możliwość rozszerzenia o nowe komponenty i usługi.
5. **Bezpieczeństwo** - szyfrowanie komunikacji oraz kontrola dostępu do zasobów.
6. **Modularność** - podział systemu na niezależne komponenty działające w kontenerach Docker.
7. **Integracja z open-source** - Wsparcie dla narzędzi i technologii dostępnych na licencji open-source.
8. **Minimalizacja kosztów** - niskie koszty sprzętowe i utrzymanie dzięki Raspberry Pi i rozwiązaniom chmurowym typu DuckDNS.
9. **Wydaźność** - optymalizacja aplikacji pod Raspberry Pi, aby zapewnić płynne działanie
10. **Łatwość wdrożenia** - uproszczona konfiguracja pozwalająca na szybkie uruchomienie systemu.

3.2 Architektura systemu

System HomeLab składa się z kilku kluczowych komponentów:

3.2.1 Backend (FastAPI + MongoDB)

- FastAPI [2] odpowiada za obsługę logiki biznesowej i API Backendu.
- MongoDB [1] przechowuje dane użytkowników, konfigurację systemową i rejestr operacji.

3.2.2 Frontend (AppSmith)

- niskokodowa platforma pozwalająca na łatwe budowanie interfejsu użytkownika.
- Integracja z backendem poprzez REST API

3.2.3 Warstwa sieciowa

- Połączenia umożliwiające i zabezpieczone poprzez Tailscale (VPN).
- DuckDNS zapewniający możliwość dostępu do systemu za pomocą domeny, zamiast adresu IP.

3.2.4 Środowisko kontenerowe

- Docker wykorzystywany do zarządzania usługami systemu.
- Możliwość łatwego wdrażania i skalowania aplikacji poprzez kontenery.

3.2.5 Automatyzacja CI/CD

- GitHub Actions zarządza automatyzacją wdrożeń i aktualizacji systemu
- Każda zmiana w kodzie uruchamia testy oraz wdrożenie nowej wersji aplikacji.

3.2.6 Urządzenie docelowe

Raspberry Pi 5 jako główny host systemu, zapewniający niskie zużycie energii i optymalizację kosztów.

3.3 Technologie i narzędzia użyte w systemie

System HomeLab wykorzystuje następujące technologie:

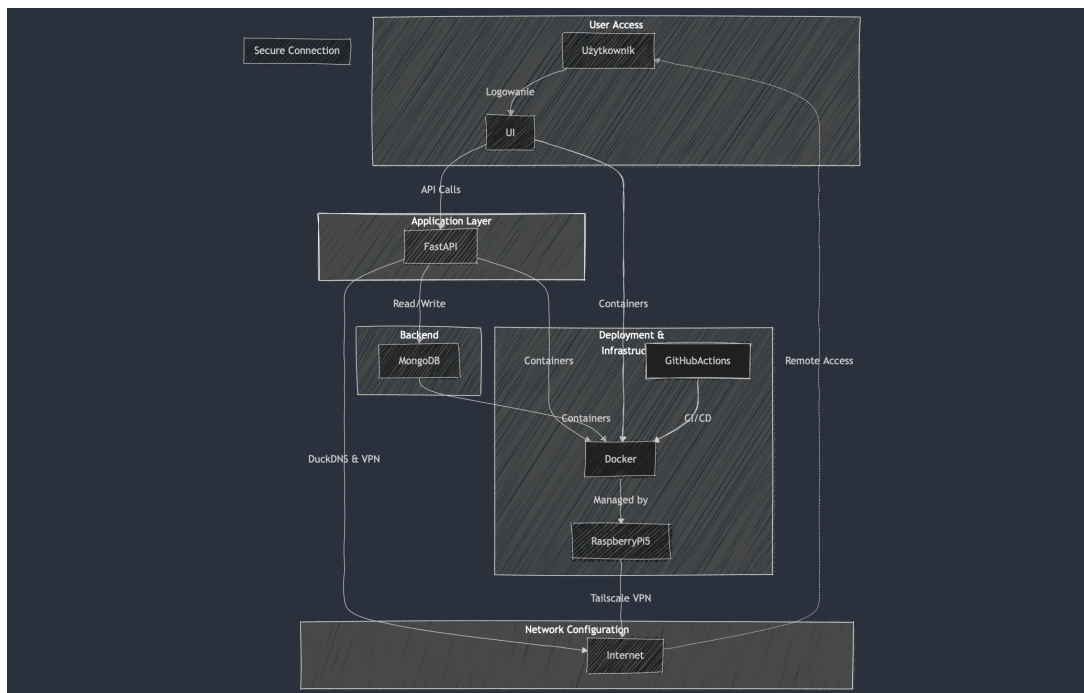


Figure 3.1: Schemat Architektury

3.3.1 Backend

- FastAPI - szybki i nowoczesny framework do tworzenia API w pythonie
- MongoDB - baza danych NoSQL przechowująca konfigurację i dane użytkowników

3.3.2 Frontend

- AppSmith - niskokodowe narzędzia do budowy interfejsu użytkownika.
- RestAPI - wykorzystywane do komunikacji między frontendem a backendem.

3.3.3 Warstwa Sieciowa

- Tailscale - VPN do bezpiecznego zapewnienia zdalnego dostępu do systemu, bez konieczności posiadania stałego adresu IP.
- DuckDNS - dynamiczny system zarządzania domeną umożliwiający łatwy dostęp do systemu.

3.3.4 Środowisko uruchomieniowe

- Docker - używany do konteneryzacji aplikacji i zarządzania zależnościami.
- Raspberry Pi 5 - host systemu zapewniający energooszczędność i niski koszt.

3.3.5 Automatyzacja CI/CD

- GitHub Actions - narzędzie do automatyzacji wdrożeń i testowania kodu.
- Pipeline CI/CD - automatyczne testowanie, budowanie i wdrażanie aplikacji

Dzięki zastosowaniu powyższych technologii system Homelab będzie nowoczesnym, skalowalnym i energooszczędnym rozwiązaniem dla użytkowników domowych.

4. Implementacja systemu

4.1 Backend - API do zarządzania systemem

4.1.1 Struktura i kluczowe endpointy API

4.1.2 Obsługa uwierzytelniania i autoryzacji

4.2 Frontend - Interfejs użytkownika

4.2.1 Projekt UI/UX

4.2.2 Implementacja aplikacji webowej

4.3 Automatyzacja Konfiguracji i wdrożenie

4.3.1 Integracja z narzędziami CI/CD

Współczesne systemy informatyczne wymagają nie tylko solidnej implementacji, ale również efektywnego zarządzania cyklem życia oprogramowania. W tym kontekście, integracja narzędzi CI/CD (Continuous Integration / Continuous Deployment) odgrywa kluczową rolę w automatyzacji procesów budowania, testowania i wdrażania aplikacji.

Zastosowanie self-hosted runnera

W celu zapewnienia kompatybilności architektury uruchomieniowej systemu, zdecydowano się na użycie **self-hosted runnera** zamiast domyślnych runnerów GitHub Actions. Domyślne maszyny CI/CD oferowane przez GitHub działają wyłącznie na **architekturze x86**, co ogranicza możliwość testowania i wdrażania aplikacji na innych platformach, takich jak **ARM** (np. Raspberry Pi, serwery oparte na ARM64).

Zastosowanie własnego runnera pozwala na:

- Uruchamianie testów i budowanie obrazów Docker na architekturze zgodnej z docelowym środowiskiem produkcyjnym.

- Pełną kontrolę nad zasobami sprzętowymi wykorzystywanymi w procesie CI/CD.
- Możliwość integracji z lokalnym registry dla przechowywania obrazów Docker.

Workflow GitHub Actions dla testowania

name: Run Tests

on:

push:

branches:

- main
- dev

pull_request:

jobs:

test:

runs-on: self-hosted

services:

mongo:

image: mongo:latest

ports:

- 27017:27017

steps:

- name: Checkout Repository

uses: actions/checkout@v4

- name: Set Up Python

uses: actions/setup-python@v4

with:

python-version: "3.13"

- name: Install Dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

pip install pytest pytest-asyncio httpx


```
- name: Run Pytest
  run: pytest -v
```

Workflow GitHub Actions dla budowania i wersjonowania obrazów Docker

Po przejściu testów jednostkowych obraz Docker jest budowany i pushowany do lokalnego rejestru uruchomionego na `localhost:5000`.

```
name: Build and Push Docker Image
```

```
on:
```

```
  push:
```

```
    branches:
```

```
      - main
      - dev
```

```
  pull_request:
```

```
    branches: [dev, main]
```

```
jobs:
```

```
  build_and_push:
```

```
    runs-on: self-hosted
```

```
    steps:
```

```
      - name: Checkout Repository
        uses: actions/checkout@v4
```

```
      - name: Extract Version
```

```
        run: echo "VERSION=$(date +%Y%m%d')-$(git rev-parse --short HEAD)" >> $GITHUB_ENV
```

```
      - name: Build Docker Image
```

```
        run: |
```

```
          docker build -t localhost:5000/myapp:${{ env.VERSION }} .
```

```
          docker tag localhost:5000/myapp:${{ env.VERSION }} localhost:5000/myapp:latest
```

```
      - name: Push Docker Image to Local Registry
```

```
        run: |
```

```
          docker push localhost:5000/myapp:${{ env.VERSION }}
```

```
          docker push localhost:5000/myapp:latest
```

Korzyści z bezpośredniego pushowania obrazu do registry

W przeciwieństwie do wcześniejszej konfiguracji, w której obraz był zapisywany lokalnie za pomocą `docker save`, obecnie jest on od razu przesyłany do prywatnego rejestru. Takie podejście:

- Umożliwia natychmiastowe wykorzystanie obrazu w środowisku produkcyjnym bez potrzeby ręcznego jego ładowania.
- Pozwala na prostsze zarządzanie wersjami obrazów w registry.
- Minimalizuje czas między budowaniem a wdrożeniem.

Publikacja obrazu do lokalnego registry

Wszystkie wersje aplikacji są automatycznie przechowywane w lokalnym rejestrze **Docker Registry**, a najnowsza wersja jest oznaczana jako **latest**. Dzięki temu wdrożenie nowej wersji sprowadza się do uruchomienia nowego kontenera:

```
docker pull localhost:5000/myapp:latest
docker run -d --name myapp localhost:5000/myapp:latest
```

Dzięki temu każda wersja aplikacji jest jednoznacznie identyfikowana, a **latest** wskazuje na najnowszą stabilną wersję.

5. Testowanie i analiza systemu

5.1 Testy jednostkowe i integracyjne

5.2 Testy wydajnościowe i bezpieczeństwa

5.3 Opinie użytkowników i ewaluacja systemu

6. Podsumowanie i wnioski

6.1 Osiągnięcia i rezultaty pracy

6.2 Możliwości dalszego rozwoju

Bibliography

- [1] MongoDB. *MongoDB Docs*. URL: <https://www.mongodb.com/docs>. (accessed: 02.02.2025).
- [2] Tiangolo. *FastAPI*. URL: <https://fastapi.tiangolo.com>. (accessed: 02.02.2025).