Uniwersytet
im. Adama Mickiewicza
w Poznaniu

Wydział Matematyki i Informatyki

Maciej Głowacki
Numer albumu: 434689

Filip Izydorczyk
Numer albumu: 434700

Marcin Woźniak
Numer albumu: 434812

Hubert Wrzesiński
Numer albumu: 434813

PlanNaPlan - Dokumentacja wdrożeniowa

Poznań, styczeń 2021

Spis treści

1 Instalacja 2
1.1 Backend 2
1.1.1 Kompilowanie aplikacji ze źródła 2
1.1.2 Binarna wersja aplikacji 4
1.2 Frontend 6
1.2.1 Budowanie strony ze źródła 6
2 Wdrożenie aplikacji 8
2.1 Pierwsze ręczne wdrożenie 8
2.1.1 Proxy 8
2.1.2 Backend 9
2.1.3 Frontend 9
2.1.4 Baza danych 9
2.2 Automatyzacja wdrożenia na produkcje 10
2.2.1 Backend 10
2.2.2 Frontend 12

1 Instalacja

1.1 Backend

1.1.1 Kompilowanie aplikacji ze źródła

- **Krok 0:** Wymagania:
  - System operacyjny: Linux
  - Java 14
  - Maven
  - Baza Danych (np. MariaDB)
  - Osobny user do uruchamiania aplikacji np. backend

- **Krok 1:** Pobranie repozytorium ze źródła:

```
% git clone git@git.plannaplan.pl:filipizydorczyk/backend.git
```

- **Krok 2:** Zmiana środowiska backendowego na produkcyjne:

```
% echo "spring.profiles.active=prod" > restservice/src/main/
    resources/application.properties
```

- **Krok 3:** Instalacja pluginów za pomocą `maven`:

```
% mvn clean
% mvn install
```

- **Krok 4:** Kompilowanie środowiska:

```
% cd restservice
% mvn clean package spring-boot:repackage
```

- **Krok 5:** Utworzenie folderu oraz skopiowanie aplikacji:

```
% mkdir -p /opt/plannaplan-backend/logs
% cp -rv restservice/target/*.jar /opt/plannaplan-backend/
    backend.jar
```

- **Krok 6:** Dodanie pustej bazy danych, a także użytkownika dla backend'u.
- **Krok 7:** Samo uruchomienie aplikacji - wiąże się z ustawieniem zmiennych środowiskowych. Polecamy stworzenie skryptu w Bashu aby to wszytsko było przetrzymywane tylko w tym skrypcie. Przykładowy skrypt 1.

Kod 1: Skrypt uruchamiający backend aplikacji

```
1  #!/bin/sh
2  SERVICE_NAME="PlanNaPlan Backend"
3  DIR="/opt/plannaplan-backend"
4
5  export PATH=$PATH:$JAVA_HOME/bin
6
7  export PLANNAPLAN_MYSQL_DB_HOST=""
8  export PLANNAPLAN_MYSQL_DB_PORT=""
9  export PLANNAPLAN_MYSQL_DB=""
10 export PLANNAPLAN_MYSQL_DB_USERNAME=""
11 export PLANNAPLAN_MYSQL_DB_PASSWORD=""
12 export PLANNAPLAN_EMAIL=""
13 export PLANNAPLAN_EMAIL_HOST=""
14 export PLANNAPLAN_EMAIL_PORT=""
15 export PLANNAPLAN_EMAIL_USERNAME=""
16 export PLANNAPLAN_EMAIL_PASSWORD=""
17 export PLANNAPLAN_CONSUMER_KEY=""
18 export PLANNAPLAN_CONSUMER_SECRET=""
19
20 java -jar $DIR/backend.jar >> $DIR/log-$(/usr/bin/date -I).log
       2>&1
21 echo $! > /tmp/sd-plananplan.pid
```

- **Krok 8:** Uzupełnienie skryptu (nr. 1) swoimi danymi produkcyjnymi.
- **Krok 9:** Utworzenie serwisu, aby backend startował przy uruchomieniu

serwera.

Kod 2: Deamon uruchamiający backend aplikacji.

```
1  [Unit]
2  Description=PlanNaPlan Backend
3  After=network.target
4
5  [Service]
6  Type=simple
7  ExecStart="/opt/plannaplan-backend/plannaplan-backend.sh"
8  WorkingDirectory=/opt/plannaplan-backend
9  User=backend #WYBRANY USER W KROKU 0
10 Group=backend #WYBRANY USER W KROKU 0
11 Environment=PATH=$PATH:$JAVA_HOME/bin
12 StartLimitInterval=30
13
14 [Install]
15 WantedBy=multi-user.target
```

- **Krok 10:** Przeładownie `systemctl`:

```
% systemctl daemon-reload
```

- **Krok 11:** Włączenie aby aplikacja uruchamiała się przy starcie systemu oraz pierwsze jej uruchomienie, a także status:

```
% systemctl enable plannaplan-backend
% systemctl start plannaplan-backend
% systemctl status plannaplan-backend
```

1.1.2 Binarna wersja aplikacji

- **Krok 0:** Wymagania:
  - System operacyjny: Linux
  - Java 14

- Baza Danych (np. MariaDB)
- Osobny user do uruchamiania aplikacji np. backend

- **Krok 1:** Pobranie binarnej.
  Wszytskie wersje binarane aplikacji znajdują się `https://git.plannaplan.pl/filipizydorczyk/backend`

- **Krok 2:** Utworzenie folderu oraz skopiowanie binarnej wersji aplikacji do folderu `/opt/plannaplan-backend/`

- **Krok 3:** Dodanie pustej bazy danych, a także użytkownika dla backend'u.

- **Krok 4:** Samo uruchomienie aplikacji - wiąże się z ustawieniem zmiennych środowiskowych. Polecamy stworzenie skryptu w Bashu aby to wszytsko było przetrzymywane tylko w tym skrypcie. Przykładowy skrypt 1.

- **Krok 5:** Uzupełnienie skryptu (nr. 1) swoimi danymi produkcyjnymi.

- **Krok 6:** Utworzenie serwisu, aby backend startował przy uruchomieniu serwera. Przykładowy kod nr. 2.

- **Krok 7:** Przeładownie `systemctl`:

```
% systemctl daemon-reload
```

- **Krok 8:** Włączenie aby aplikacja uruchamiała się przy starcie systemu oraz pierwsze jej uruchomienie, a także status:

```
% systemctl enable plannaplan-backend
% systemctl start plannaplan-backend
% systemctl status plannaplan-backend
```

1.2 Frontend

1.2.1 Budowanie strony ze źródła

- **Krok 0:** Wymagania:
  - System operacyjny: Linux
  - Yarn
  - Httpd
  – Zainstalowany certyfikat SSL (np. Let's Encrypt [1])
  - Osobny user do uruchamiania aplikacji np. frontend

- **Krok 1:** Pobranie repozytorium ze źródła:

```
% git clone http://git.plannaplan.pl/y0rune/frontend.git
```

- **Krok 2:** Dodanie zmiennej środowiskowej razem z adresem URL aplikacji backend:

```
% echo "REACT_APP_API_URL=https://wmi-backend.plannaplan.pl" > .
    env
```

- **Krok 3:** Zainstalowanie pluginów za pomocą `Yarn`:

```
% yarn
```

- **Krok 4:** Zbudowanie statycznej strony:

```
% yarn run build
```

- **Krok 5:** Skopiowanie plików z folderu `build/` do lokalizacji strony w `Httpd` (np. `/var/www/plannaplan.pl`)

- **Krok 6:** Przykładowa konfiguracja (Kod nr. 3) `Httpd` dla statycznej strony w katalogu `/var/www/plannaplan.pl`.

[1] Strona Let's Encrypt: `https://letsencrypt.org/`

Kod 3: Konfiguracja statycznej strony w Httpd.

```
<VirtualHost *:80>
   RewriteEngine on
   RewriteCond %{SERVER_NAME} = #URL DO STRONY
   RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=
       permanent]
</VirtualHost>

<VirtualHost *:443>
        SSLEngine on
        ServerName #URL DO STRONY :443
        DocumentRoot /var/www/plannaplan.pl/
        ServerAdmin #EMAIL ADMINISTRATORA
        SSLCertificateFile /etc/letsencrypt/live/full.plannaplan.
            pl/fullchain.pem
        SSLCertificateKeyFile /etc/letsencrypt/live/full.
            plannaplan.pl/privkey.pem
        Include /etc/letsencrypt/options-ssl-apache.conf

        <Directory /var/www/plannaplan.pl/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Allow from all
                Options -Indexes
        </Directory>
</VirtualHost>
```

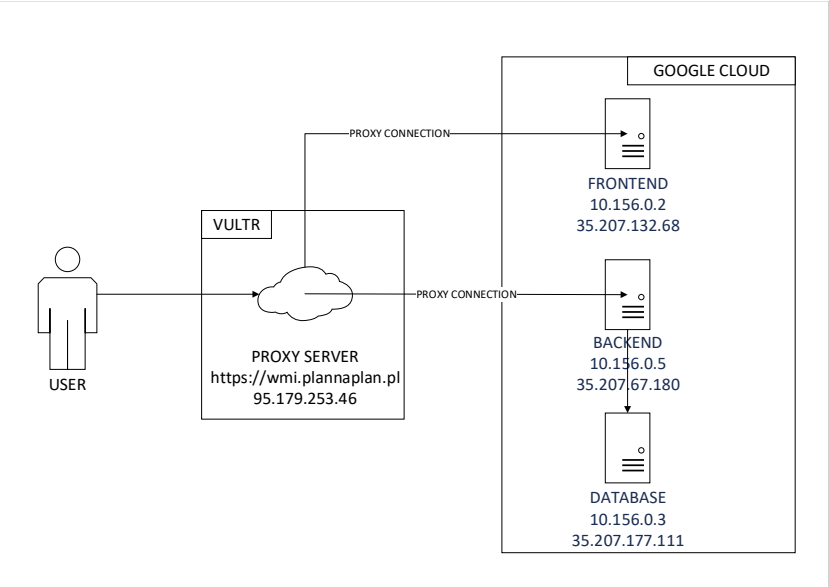
- **Krok 7:** Restart `Httpd` za pomocą `systemctl`

```
% systemctl httpd restart
```

2 Wdrożenie aplikacji

2.1 Pierwsze ręczne wdrożenie

Każdy z naszych serwisów czyli Backend, Frontend, Baza Danych zostały wdrożone do serwerów, które znajdują się w chmurze Google Cloud Platform[2]. Dodatkowo ze względów bezpieczeństwa ruch jest przekierowany przez serwer Proxy.



Rysunek 1: Infrastruktura naszej aplikacji
Źródło: Opracowanie własne

2.1.1 Proxy

Serwer znajduje się w chmurze Vultr[3]. Pełni on role serwera proxy [4].

Informacje o serwerze:

- 1vCPU

[2]Źródło: `https://cloud.google.com`
[3]Chmura Vultr: `https://www.vultr.com/`
[4]Definicja serwera proxy. Link: `https://en.wikipedia.org/wiki/Proxy_server`

- 1 GB pamięci RAM
- 25 GB pojemność dysku
- System Operacyjny: Gentoo Linux [5]
- Lokalizacja: Frankfurt, Niemcy

2.1.2 Backend

Wdrożenie backendu aplikacji odbywało się podobnie jak to zostało opisane w podrozdziale 1.1.2.

Informacje o serwerze:

- 1vCPU
- 1.7 GB pamięci RAM
- 20 GB pojemność dysku
- System Operacyjny: CentOS 8
- Lokalizacja: Frankfurt, Niemcy

2.1.3 Frontend

Wdrożenie frontendu aplikacji odbywało się podobnie jak to zostało opisane w podrozdziale 1.2.1.

Informacje o serwerze:

- 1vCPU
- 600 MB pamięci RAM
- 20 GB pojemność dysku
- System Operacyjny: CentOS 8
- Lokalizacja: Frankfurt, Niemcy

2.1.4 Baza danych

Wdrożenie bazy danych (MariaDB[6] w naszym przypadku) polegało na zainstalowaniu jej na osobnej maszynie.

[5]Strona główna dystrybucji: `https://gentoo.org`
[6]Strona projektu MariaDB `https://mariadb.org/`

Informacje o serwerze:

- 1vCPU
- 600 MB pamięci RAM
- 20 GB pojemność dysku
- System Operacyjny: CentOS 8
- Lokalizacja: Frankfurt, Niemcy

2.2 Automatyzacja wdrożenia na produkcje

Z powodu dość czasochłonnej każdorazowej zmiany na produkcję, doszliśmy do wniosku, że automatyzacja jest czymś koniecznym w naszym projekcie. Gdy odbywają się prace, które mają na celu poprawę aplikacji, każda taka zmiana odbywa się na osobnej gałęzi (branchu). Następnie, osoba wykonująca zmianę prosi o zmianę (w Pull Request) a kolejna osoba sprawdza czy wszystkie testy przechodzą na lokalnej maszynie użytkownika. W dalszym kroku akceptuje zmiany a następnie repozytorium klonowane jest do kilku instancji (GitLab, GitHub, Gitea) w celu zapasowej oraz przeprowadzenia kompilacji i wdrożenie jej na wybrany serwis.

2.2.1 Backend

Automatyzacja backendu odbywa się w kilku krokach:

- build - aplikacja jest kompilowana, jeżeli wszytsko przejdzie zgodnie z planem CI/CD uruchamia kolejny krok.
- deploy_production - binarny plik jest wysyłany do serwera, serwis `plannaplan-backend` jest restartowny. W celu przeładowania aplikacji.

Kod 4: Konfiguracja CI/CD backendu

```
1 stages:
2   - build
3   - deploy
```

```
4
5  build:
6    stage: build
7    image: maven
8    script:
9      - echo "Start building App"
10     - echo "spring.profiles.active=prod" > restservice/src/main/
         resources/application.properties
11     - mvn clean
12     - mvn install
13     - cd restservice
14     - mvn clean package spring-boot:repackage
15     - echo "Build successfully!"
16   artifacts:
17     expire_in: 1 hour
18     paths:
19       - restservice/target/
20   only:
21     - master
22
23 deploy_production:
24   stage: deploy
25   before_script:
26     - apt-get update
27     - apt-get --yes --force-yes install rsync
28   script:
29     - 'which ssh-agent || ( apt-get update -y && apt-get install
         openssh-client -y )'
30     - eval $(ssh-agent -s)
31     - ssh-add <(echo "$SSH_PRIVATE_KEY")
32     - mkdir -p ~/.ssh
33     - '[[ -f /.dockerenv ]] && echo -e "Host *\n\
         tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
34     - echo "Deploying to server"
35     - ssh backend@wmi-backend-gc.plannaplan.pl -t "sudo systemctl
```

```
        stop plannaplan-backend"
36    - ssh backend@wmi-backend-gc.plannaplan.pl -t "rm -rf /opt/
        plannaplan-backend/backend.jar"
37    - rsync --progress restservice/target/*.jar backend@wmi-backend-
        gc.plannaplan.pl:/opt/plannaplan-backend/backend.jar
38    - sleep 5
39    - ssh backend@wmi-backend-gc.plannaplan.pl -t "sudo systemctl
        start plannaplan-backend"
40    - echo "Deployed"
41  only:
42    - master
```

2.2.2 Frontend

- build - aplikacja jest kompilowana, jeżeli wszytsko przejdzie zgodnie z planem CI/CD uruchamia kolejny krok.
- deploy_production - statyczna strona wysyłana do serwera, serwis `httpd` jest restartowny. W celu przeładowania aplikacji.

Kod 5: Konfiguracja CI/CD frontendu

```
1  stages:
2    - build
3    - deploy
4
5  build:
6    stage: build
7    image: node
8    script:
9      - echo "Start building App"
10     - echo "REACT_APP_API_URL=https://wmi-backend.plannaplan.pl" > .
         env
11     - yarn
12     - CI= yarn run build
```

```
13      - echo "Build successfully!"
14    artifacts:
15      expire_in: 1 hour
16      paths:
17        - build
18        - node_modules/
19
20  deploy_production:
21    stage: deploy
22    before_script:
23      - apt-get update
24      - apt-get --yes --force-yes install rsync
25    script:
26      - 'which ssh-agent || ( apt-get update -y && apt-get install
            openssh-client -y )'
27      - eval $(ssh-agent -s)
28      - ssh-add <(echo "$SSH_PRIVATE_KEY")
29      - mkdir -p ~/.ssh
30      - '[[ -f /.dockerenv ]] && echo -e "Host *\n\
            tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
31      - echo "Deploying to server"
32      - rsync --progress -r build/* --delete website@wmi-frontend.
            plannaplan.pl:/var/www/plannaplan.pl
33      - echo "Deployed"
34    only:
35      - master
```