apiary.apib

FORMAT: 1A
HOST: http://localhost/api

cbos-comunicator api

Toto je sablona pra dalsi budouci mikrosluzby.

Status [/status]

Tato metoda je určena pro overeni, zda sluzba bezi

Status [GET]

+ Request (application/json)

+ Response 200 (application/json; charset=utf-8)
    + Attributes(Status, fixed-type)

Data Structures

Status
+ app_name: bootstrap-oracle-api (string)
+ status: running (string)
+ version: 1.0.0 (string)

.npmrc

registry=https://pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/

always-auth=true

; begin auth token
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:username=myocto
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:_password=NXR4N212bWdhd2pscnZvY2loaWhsenlsbmx2NTU1eGZlM2hrZnZtcXZyazcyYTZtNjRpYQ==
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:email=npm requires email to be set but doesn't use the value
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:username=myocto
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:_password=NXR4N212bWdhd2pscnZvY2loaWhsenlsbmx2NTU1eGZlM2hrZnZtcXZyazcyYTZtNjRpYQ==
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:email=npm requires email to be set but doesn't use the value
; end auth token

package.json

```
{
    "name": "cbos-comunicator-api",
```

```
    "scripts": {
        "dev": "cross-env-shell DEBUG=postgres-utils LOG_LEVEL=debug
LOG_FORMAT=pretty_json node ./src/index.js",
        "start": "node src/index.js"
    },
    "dependencies": {
        "app-utils": "^1.19.11",
        "body-parser": "^1.20.2",
        "cookie-parser": "^1.4.6",
        "cross-env": "^7.0.3",
        "dotenv": "^8.6.0",
        "express": "^4.19.2",
        "express-validator": "^6.15.0",
        "postgres-utils": "^1.4.2"
    }
}
```

bootstrap-postgres.yml

```yaml
version: "3.3"
services:
  bootstrap-postgres:
    image: ogwhub.azurecr.io/libs/bootstrap-postgres:develop-latest
    environment:
      PG_HOST: "192.168.130.238"
      PG_PORT: 5432
      PG_USER: "pumpa"
      PG_PASSWORD: "pumpa"
      PG_DATABASE: "centrum"
      TZ: "Europe/Prague"
    ports:
      - 1111:3000
    networks:
      - default
    logging:
      driver: json-file
    deploy:
      restart_policy:
        condition: on-failure
```

set_env_variables.sh

```
#!/bin/sh
```

```
: ${ENV_SECRETS_DIR:=/run/secrets}

env_secret_debug()
{
    if [ ! -z "$ENV_SECRETS_DEBUG" ]; then
        echo -e "\033[1m$@\033[0m"
    fi
}

# usage: env_secret_expand VAR
#    ie: env_secret_expand 'XYZ_DB_PASSWORD'
# (will check for "$XYZ_DB_PASSWORD" variable value for a placeholder that defines the
#  name of the docker secret to use instead of the original value. For example:
# XYZ_DB_PASSWORD={{DOCKER-SECRET:my-db.secret}}
env_secret_expand() {
    var="$1"
    eval val=\$$var
    if secret_name=$(expr match "$val" "DOCKER-SECRET->\([^}]\+\)$"); then
        secret="${ENV_SECRETS_DIR}/${secret_name}"
        env_secret_debug "Secret file for $var: $secret"
        if [ -f "$secret" ]; then
            val=$(cat "${secret}")
            export "$var"="$val"
            env_secret_debug "Expanded variable: $var=$val"
        else
            env_secret_debug "Secret file does not exist! $secret"
        fi
    fi
}

env_secrets_expand() {
    for env_var in $(printenv | cut -f1 -d"=")
    do
        env_secret_expand $env_var
    done

    if [ ! -z "$ENV_SECRETS_DEBUG" ]; then
        echo -e "\n\033[1mExpanded environment variables\033[0m"
        printenv
    fi
}

env_secrets_expand

exec "$@"
```

start.sh

```
#!/bin/bash

# cd service || exit

# npm start &

# cd ..

cd api || exit

npm start
```

.npmrc

registry=https://pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/

always-auth=true

; begin auth token
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:username=myocto
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:_password=NXR4N212bWdhd2pscnZvY2loaWhsenlsbmx2NTU1eGZlM2hrZnZtcXZyazcyYTZtNjRpYQ==
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/registry/:email=npm requires email to be set but doesn't use the value

//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:username=myocto
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:_password=NXR4N212bWdhd2pscnZvY2loaWhsenlsbmx2NTU1eGZlM2hrZnZtcXZyazcyYTZtNjRpYQ==
//pkgs.dev.azure.com/myocto/_packaging/myocto/npm/:email=npm requires email to be set but doesn't use the value
; end auth token

zure-pipelines.yml

```yaml
trigger:
    branches:
        include:
            - refs/heads/*

variables:
    - name: dockerRegistryServiceConnection
      value: ogwhub
    - name: vmImageName
      value: 'ubuntu-latest'

stages:
    - stage: Build
      displayName: Build and push image
      condition: or(
          eq(variables['build.sourceBranch'], 'refs/heads/master'),
          eq(variables['build.sourceBranch'], 'refs/heads/pre-release'),
          eq(variables['build.sourceBranch'], 'refs/heads/develop'),
          eq(variables['build.sourceBranch'], 'refs/heads/test'),
          contains(variables['build.sourceBranch'], 'refs/heads/hotfix')
          )
      jobs:
          - job: set_variables
            displayName: Set variables
```

```
    pool:
        vmImage: $(vmImageName)
    steps:
        - bash: |
            version=`node -p "const p = require('./package.json'); p.version;"`
            echo "##vso[task.setvariable variable=version;isOutput=true]$version"
          name: setvarStep
  - job: Build
    displayName: Build
    dependsOn: set_variables
    condition: or(
        eq(variables['build.sourceBranch'], 'refs/heads/pre-release'),
        eq(variables['build.sourceBranch'], 'refs/heads/develop'),
        eq(variables['build.sourceBranch'], 'refs/heads/test')
        )
    variables:
        tag: $[format('{0}-latest', replace(variables['build.sourceBranch'],
'refs/heads/', ''))]
        projectName: $[lower(variables['system.teamProject'])]
    pool:
        vmImage: $(vmImageName)
    steps:
        - checkout: self
          persistCredentials: true
        - task: Docker@2
          displayName: Build develop and push an image to container registry
          inputs:
            command: buildAndPush
            repository: $(projectName)/$(Build.Repository.Name)
            dockerfile: ./Dockerfile
            containerRegistry: $(dockerRegistryServiceConnection)
            tags: |
                $(tag)
  - job: build_release
    displayName: Build release
    dependsOn: set_variables
    condition: eq(variables['build.sourceBranch'], 'refs/heads/master')
    variables:
        tag: $[dependencies.set_variables.outputs['setvarStep.version']]
        projectName: $[lower(variables['system.teamProject'])]
    pool:
        vmImage: $(vmImageName)
    steps:
        - checkout: self
          persistCredentials: true
        - task: Docker@2
          displayName: Build develop and push an image to container registry
          inputs:
            command: buildAndPush
```

```
                    repository: $(projectName)/$(Build.Repository.Name)
                    dockerfile: ./Dockerfile
                    containerRegistry: $(dockerRegistryServiceConnection)
                    tags: |
                        latest
                        $(tag)
                - script: |
                    git config --global user.name "BuildService"
                    git config --global user.email "autobuild@ys.cz"
                    git tag -d $(tag)
                    git push origin :refs/tags/$(tag)
                    git tag -a $(tag) -m "Tag $(tag)"
                    git push origin $(tag)
                  displayName: Git tag
        - job: build_hotfix
          displayName: Build hotfix
          dependsOn: set_variables
          condition: contains(variables['build.sourceBranch'], 'refs/heads/hotfix')
          variables:
              tag: $[dependencies.set_variables.outputs['setvarStep.version']]
              projectName: $[lower(variables['system.teamProject'])]
          pool:
              vmImage: $(vmImageName)
          steps:
              - checkout: self
                persistCredentials: true
              - task: Docker@2
                displayName: Build hotfix and push an image to container registry
                inputs:
                    command: buildAndPush
                    repository: $(projectName)/$(Build.Repository.Name)
                    dockerfile: ./Dockerfile
                    containerRegistry: $(dockerRegistryServiceConnection)
                    tags: |
                        $(tag)
              - script: |
                    git config --global user.name "BuildService"
                    git config --global user.email "autobuild@ys.cz"
                    git tag -d $(tag)
                    git push origin :refs/tags/$(tag)
                    git tag -a $(tag) -m "Tag $(tag)"
                    git push origin $(tag)
                displayName: Git tag
```

Dockerfile

```
FROM node:18.14-alpine

ENV API_FOLDER=api
ENV SERVICE_FOLDER=service

RUN apk add --update bash tzdata

ADD ./deploy ./deploy

ADD package.json .

ADD $API_FOLDER/src ./$API_FOLDER/src

ADD $API_FOLDER/.npmrc ./$API_FOLDER
ADD $API_FOLDER/package.json ./$API_FOLDER
ADD $API_FOLDER/package-lock.json ./$API_FOLDER

RUN cd $API_FOLDER &&  npm ci --omit=dev && cd ..

# ADD $SERVICE_FOLDER/src ./$SERVICE_FOLDER/src

# ADD $SERVICE_FOLDER/.npmrc ./$SERVICE_FOLDER
# ADD $SERVICE_FOLDER/package.json ./$SERVICE_FOLDER
# ADD $SERVICE_FOLDER/package-lock.json ./$SERVICE_FOLDER

# RUN cd $SERVICE_FOLDER && npm ci --omit=dev && cd ..

RUN chmod a+x ./deploy/set_env_variables.sh
RUN chmod a+x ./deploy/start.sh

ENV ORA_SDTZ "Europe/Prague"

ENTRYPOINT ["./deploy/set_env_variables.sh"]
CMD ["./deploy/start.sh"]
```

package.json

```
{
    "version": "4.0.0"
}
```

README.md

```
# Popis cbos-comunicator

-   Jedná se o šablonu služby připojené k postgres

## ENDPOINT

-   GET `/api/status`
    -   Vrací status aplikace a kontroluje připojení k POSTGRES (v případě neúspěšného připojení vypíná aplikaci)

## ~~SERVICE~~

-   ~~Service obsahuje scheduler, který pravidelně kontroluje napojení na POSTGRES (defaultně každých 5 min)~~

# Nastavení (ENV PROMENNE)

| Název       | Popis                                                          | Defaultní hodnota | Příklad           |
| ----------- | -------------------------------------------------------------- | ----------------- | ----------------- |
| PG_HOST     | IP adtesa serveru, kde je nainstalovananá datzbáze POSTGRES    |
| '192.168.130.235' |
| PG_PORT     | Číso portu na kterém je dostupná databáze                      |
| 5432              |
| PG_USER     | Postgres uživatelské jméno                                     |
| 'pumpa'           |
| PG_PASSWORD | Postgres uživatelské heslo                                     |
| 'pumpa'           |
| PG_DATABASE | Název databáze, do které se chceme připojit                    |
| 'centrum'         |
| PORT        | Číslo portu, na které jsou dostupné EP pro API                 | 3000              |
|                   |
```