

Instrukcja uruchomieniowa i użytkowania:

W celu zbudowania i uruchomienia aplikacji lokalnie na swoim komputerze, należy:

Wejść w consoli 'Command Prompt' do folderu głównego aplikacji klienckiej, tj. folder o nazwie 'projekt'.

Następnie należy użyć dwóch poleceń gradle'owych:

```
gradlew.bat build
```

```
gradlew.bat bootRun
```

opis znaczenia powyższych komend (budowanie, uruchomienie) znajdziemy tutaj:

https://docs.gradle.org/current/userguide/gradle_wrapper.html

<https://medium.com/@itsromiljain/gradle-installation-and-gradle-springboot-web-application-611a845b011e>

W czasie wykonywania 2 komendy są między innymi wykonywane testy jednostkowe. Jako oczekiwany rezultat, w obu przypadkach powinniśmy otrzymać informację o sukcesie.

```
C:\Users\Administrator\Desktop\10\projekt>gradlew.bat build
BUILD SUCCESSFUL in 19s
5 actionable tasks: 5 up-to-date
C:\Users\Administrator\Desktop\10\projekt>_
```

```
2018-09-01 21:44:41.394 INFO 91748 --- [
    main] pl.edu.agh.project.ProjectApplication
    : Started ProjectApplication in 20.334 second
    s (JVM running for 24.162)
<=====-----> 75% EXECUTING [3m 8s]
> :bootRun
```

Możemy podglądnąć szczegóły wyników testów w oknie przeglądarki pod przekierowaniem:
file:///C:/[ścieżkaDoKatalogu]/projekt/build/reports/tests/test/index.html

Po zbudowaniu pliku .jar aplikacji serwerową możemy uruchamiać bezpośrednio poleceniem:

```
java -jar build\libs\projekt-0.0.1-SNAPSHOT.jar
```

Nasza aplikacja serwerowa już działa poprawnie. Zgodnie z przyjętymi wartościami, na lokalnym środowisku serwer jest ustawiony na port 5000.

Aplikacja kliencka

W celu uruchomienia aplikacji klienckiej, należy przejść do jej głównego folderu o nazwie "klientReactjs".

Następnie, w przypadku braku zainstalowanych plików związanych z node_modules, należy użyć komendy instalującej owe moduły, tj.:

```
npm install
```

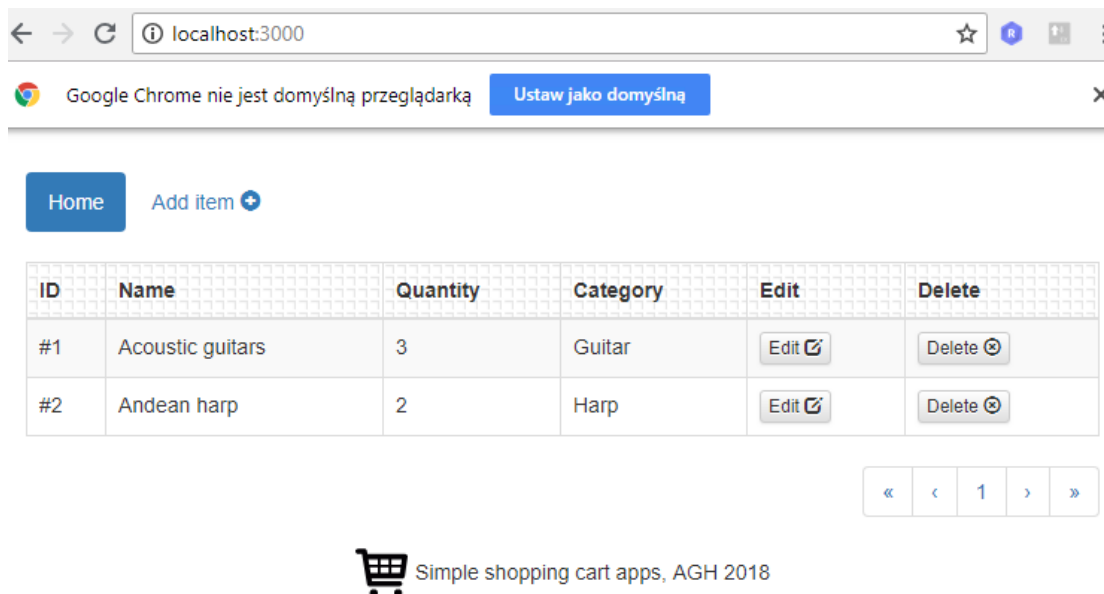
W kolejnym kroku uruchamiamy aplikację kliencką komendą:

```
npm start
```

W przypadku poprawnego wykonania uruchomienia oczekiwanym widokiem w oknie konsoli jest:

```
[1191] ./src/components/ItemEdit.js 8.1 kB {0} [built]
[1192] ./src/components/common/FormField.js 6.67 kB {0} [built]
[1193] ./src/components/common/FormSubmit.js 5.05 kB {0} [built]
[1194] ./src/components/NotFound.js 3.76 kB {0} [built]
webpack: Compiled successfully.
```

Po uruchomieniu aplikacji na porcie aplikacji klienta (domyślnie port ustawiłem na 3000 w pliku konfiguracyjnym), użytkownik w widoku przeglądarki zastaje 'crud-list'e. Aplikacja obsługuje wszystkie podstawowe metody CRUD.



Aby uruchomić testy aplikacji klienckiej, należy będą w folderze 'klientReactjs' uruchomić polecenie:

```
npm test
```

Oczekiwanym rezultatem jest pozytywny rezultat dla wszystkich utworzonych przypadków testowych.

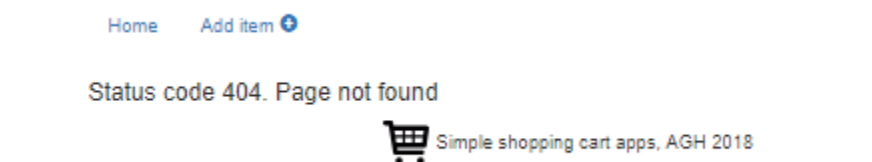
```

itemsFetchList()
  ✓ should return the ITEMS_LIST_SAVE action
itemsAddEdit() - add
  ✓ should return the ITEMS_ADD_SAVE action
itemsAddEdit() - edit
  ✓ should return the ITEMS_ADD_SAVE action
itemsDelete()
  ✓ should return the ITEMS_DELETE_SAVE action

18 passing (281ms)

```

W przypadku wprowadzenia nieobsługiwanego endpoint'u, użytkownik zostanie przekierowany do podstorny 'Page not found'.



Użytkownik ma możliwość zdefiniowania nowego elementu dla koszyka sklepowego. Zgodnie z zdefiniowanymi wartościami, pole Quantity obsługuje typ prosty 'integer'.

W celu stworzenia bardziej przyjaznego interfejsu dla użytkownika, dodane zostały proste walidatory

wypełnienia formularza.

The first screenshot shows a form with two fields: 'Name' and 'Quantity'. Both fields are empty and have a red border with a red 'x' icon in the top right corner. Below each field is a red error message: 'Name is required' and 'Quantity is required'.

The second screenshot shows the same form with the 'Name' field containing the text 'name' and the 'Quantity' field containing the text 'quantity'. Both fields now have a green border and a green checkmark icon in the top right corner.

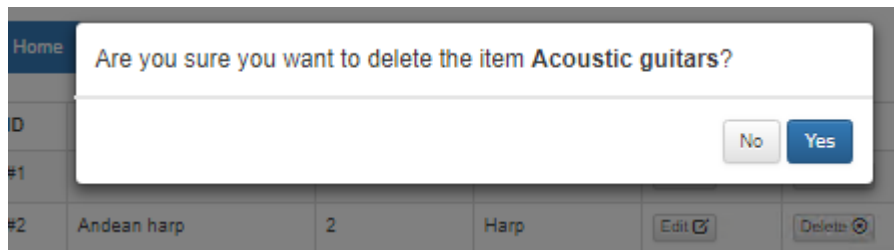
Użytkownik ma wybór kategorii przedmioty z rozwijanego menu listy wyboru.

The screenshot shows a form with a 'Category' label and a 'Save' button. To the right of the 'Category' label is a dropdown menu. The dropdown menu is open, showing a list of categories: 'Percussion', 'Guitar', 'Percussion', 'Harp', 'Keyboard', 'Horn', 'Trombone', and 'Piano'. The 'Keyboard' category is highlighted with a blue background.

W przypadku dodania więcej niż 10 przedmiotów wysokość koszyka nie jest powiększana, lecz kolejne rekordy są ukrywane. Możemy przewijać liste z użyciem paginacji.

The screenshot shows a pagination control with a series of buttons: '<<', '<', '1', '2', '>', and '>>'. The '1' button is highlighted with a blue background.

Aplikacja wykorzystuje 'pop-up' podczas próby usunięcia przedmiotu, prosząc użytkownika o potwierdzenie.



W celu zbudowania projektu do pliku bundle.js pod docelowe środowisko , należy uruchomić komendę:

```
npm run build-dev
```

Dzięki temu zostanie utworzony plik bundle.js, który w następnym kroku wdrożymy na chmurę IBM.

Wdrożenie aplikacji do chmury IBM

W celu wdrożenia aplikacji do chmury IBM, należy wykonać kolejne kroki zgodnie z instrukcją (pobrać i zainstalować IBM® Cloud developer tools z repozytorium github'a) :

<https://console.bluemix.net/docs/cli/index.html>

```
ibmcloud dev help
```

```
PS C:\Users\Administrator> ibmcloud dev help
NAME:
  C:\Program Files\IBM\Cloud\bin\ibmcloud.exe dev - A CLI plugi
USAGE:
  C:\Program Files\IBM\Cloud\bin\ibmcloud.exe dev command [argu
VERSION:
  2.1.4
COMMANDS:
  build      Build the application in a local container
  code       Download the code from an application
  console    Opens the IBM Cloud console for an applica
```

Następnie tworzymy na IBM Cloud nowy serwer dla naszej aplikacji serwerowej z użyciem Liberty for Java

<https://console.bluemix.net/catalog/starters/liberty-for-java>

W kolejnym kroku zakładamy nowy serwer dla naszej aplikacji klienckiej z użyciem SDK for Node.js

<https://console.bluemix.net/catalog/starters/sdk-for-nodejs>

ibmcloud api <https://api.eu-gb.bluemix.net>

Należy zwrócić uwagę na to, gdzie znajduje się serwer i wybrać prawidłową zmienną (w tym przypadku eu-gb -> oznacza Wielką Brytanię).

```
PS C:\Users\Administrator> ibmcloud api https://api.eu-gb.ibmcloud.net
Setting api endpoint to https://api.eu-gb.ibmcloud.net...
OK
API endpoint: https://api.eu-gb.ibmcloud.net
Not logged in. Use 'C:\Program Files\IBM\Cloud\bin\ibmcloud.exe login'
```

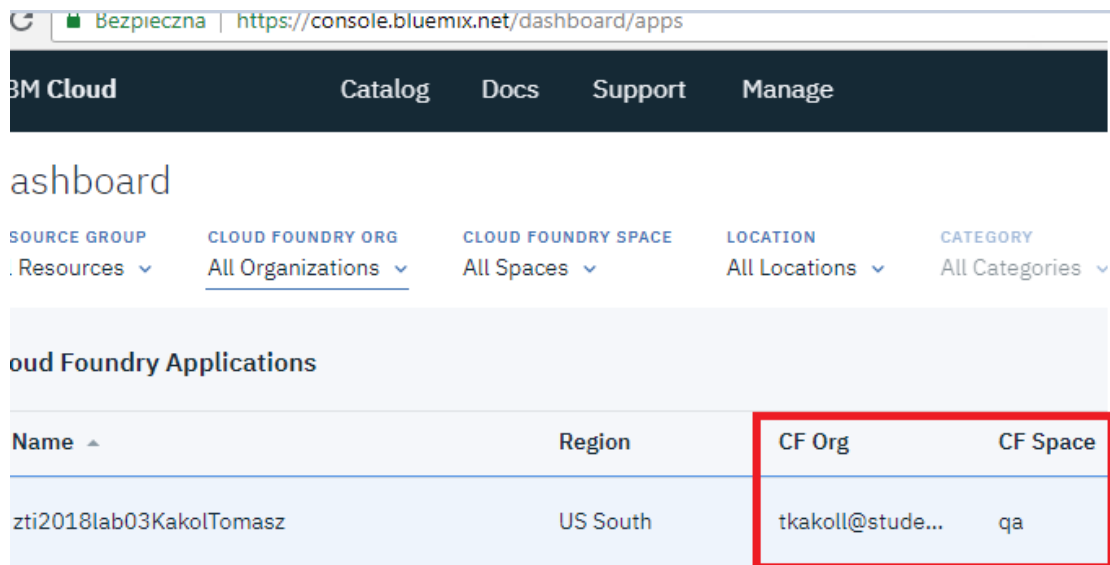
W celu połączenia i zalogowania się na IBM Cloud, musimy użyć następującej komendy:

```
bluemix login -u tkakoll@student.agh.edu.pl -o org_name -s space_name
```

W moim przypadku było to:

```
bluemix login -u tkakoll@student.agh.edu.pl -o tkakoll@student.agh.edu.pl -s qa
```

Jeżeli nie wiemy jaką ustawiliśmy nazwę organizacji i przestrzeni dla naszego zasobu, możemy to sprawdzić również z poziomu przeglądarki.



The screenshot shows the IBM Cloud console dashboard. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', and 'Manage'. Below the navigation bar, the 'dashboard' is displayed. There are several filter tabs: 'SOURCE GROUP' (Resources), 'CLOUD FOUNDRY ORG' (All Organizations), 'CLOUD FOUNDRY SPACE' (All Spaces), 'LOCATION' (All Locations), and 'CATEGORY' (All Categories). The main content area is titled 'Cloud Foundry Applications'. It contains a table with the following data:

Name	Region	CF Org	CF Space
zti2018lab03KakolTomasz	US South	tkakoll@stude...	qa

The 'CF Org' and 'CF Space' columns are highlighted with a red box.

W celu utworzenia redeploy'a naszej aplikacji, należy użyć poniższej komendy (znajdując się równocześnie w ścieżce katalogu projektu):

```
bluemix app push tomasz-kakol-server-zti -p ./build/libs/projekt-0.0.1-SNAPSHOT.jar
```

Po uruchomieniu oczekiwanym rezultatem jest prawidłowe uruchomienie aplikacji serwerowej (Tutaj przeznaczyłem więcej miejsca dla zasobu niż było to potrzebne. Nie wiedziałem ile projekt będzie potrzebował miejsca. Później zmniejszyłem -> ale już z poziomu przeglądarki):

```

start command: $PWD/.java/jre/bin/java -Xtune:virtualized -Dcom.ibm.jsse2
               -Xdump:heap:defaults:file=../dumps/heapdump.%Y%m%d.%H%M%S.p
               -Xdump:java:defaults:file=../dumps/javacore.%Y%m%d.%H%M%S.p
               -Xdump:snap:defaults:file=../dumps/Snap.%Y%m%d.%H%M%S.%p
               -Xdump:tool:events=systhrow,filter=java/lang/OutOfMemoryEr
               $JVM_ARGS org.springframework.boot.loader.JarLauncher --se

state    since    cpu    memory    disk
#0  running  2018-09-02T11:51:04Z  77.9%  155.8M of 512M  133.8M of 1G

PS C:\Users\Administrator\Desktop\10\projekt>

```

Uruchomienie aplikacji klienckiej.

Analogicznie jak w przypadku serwera, tutaj również wykonujemy te same kroki. Tym razem, z powodu tego, że aplikacji klienckiej napisanej w języku Javascript z wykorzystaniem frameworka ReactJS, nie skorzystamy z 'Libery for Java'.

Wszystkie komendy wykonujemy w takim samym porządku jak w przypadku serwera, podmieniając jedynie nazwę aplikacji klienckiej.

Z powodu zbudowania pliku .js komenda 'push' przyjmuje inne parametry (- c).

```
bluemix app push tomasz-kakol-clientJS-zti -c "node server.js"
```

Należy się upewnić, czy wcześniej wybudowaliśmy aplikację FE, czyli plik bundle.js z użyciem komendy 'npm start build-dev' na lokalnym komputerze.

```

- bower_components (nothing to cache)
-----> Build succeeded!
├─ jsdoc-react@1.0.0
├─ react@15.6.2
├─ react-bootstrap@0.30.10
├─ react-dom@15.6.2
├─ react-redux@5.0.7
├─ react-router@3.2.1
├─ react-router-bootstrap@0.23.3
├─ react-router-redux@4.0.8
├─ redux@3.7.2
├─ redux-form@6.4.3
└─ redux-saga@0.14.8

```

```

Waiting for app to start...

name:          tomasz-kakol-clientJS-zti
requested state: started
instances:     1/1
usage:         256M x 1 instances
routes:        tomasz-kakol-clientJS-zti.mybluemix.net
last uploaded: Mon 03 Sep 23:20:44 CEST 2018
stack:         cflinuxfs2
buildpack:     SDK for Node.js(TM) (ibm-node.js-6.14.3, buildpack
start command: node server.js

state    since    cpu    memory    disk
#0  running  2018-09-03T21:23:20Z  0.0%  1M of 256M  1.3M of 1G

PS C:\Users\Administrator\Desktop\10\klientReactjs>

```

Po ukończeniu wdrażania aplikacji serwerowej i klienckiej na chmurę IBM, aplikacja kliencka jest dostępna pod linkiem:


<https://tomasz-kakol-clientjs-zti.mybluemix.net/>

<https://tomasz-kakol-clientjs-zti.mybluemix.net>

Home

Add item ➕

ID	Name	Quantity
#1	Acoustic guitars	3
#2	Andean harp	2



Podsumowanie:

Aplikacja pod względem ilości funkcjonalności i jakości kodu nie jest bardzo dobra (w dużej mierze posiłkowałem się dostępną treścią z internetu), lecz główną zaletą jest zrealizowany sposób wdrażania aplikacji klient-serwer na chmurowisko. Przedstawiony sposób umożliwia stosunkowo łatwe przeskalowanie projektu do większych rozmiarów dzięki zastosowanemu rozwiązaniu. Tak wdrażany projekt jest wstępem do większych zaawansowanych projektów 'mikroserwisowych'.