

Politechnika Świętokrzyska w Kielcach

Wydział Elektroniki, Automatyki i Informatyki

Projekt: **Programowanie w Java**

Grupa: 3ID11A	Temat: Codebin – System uruchamiania repozytoriów w chmurze obliczeniowej	Skład grupy: Michał Młodawski Piotr Piasecki Tobiasz Nartowski Kacper Obara
Rok studiów: 3		

Codebin – System uruchamiania repozytoriów w chmurze obliczeniowej

Opracowanie i sprawozdanie projektu Codebin

Czerwiec 2019

Spis treści

1. Opis projektu	1
2. Wykorzystane technologie	1
1. Użyte języki programowania oraz formaty tekstu	1
2. Wykorzystane oprogramowanie przy projektowaniu i wdrażaniu projektu	1
3. Użyte biblioteki w projekcie	2
1. Biblioteki zewnętrzne	2
2. Biblioteki wewnętrzne	2
4. Funkcjonalność projektu	3
5. Obsługa projektu i sposób jego uruchomienia	3
1. Budowanie projektu	3
1. Proces kompilacji aplikacji	3
6. Wygląd interfejsu webowego	5
1. Strona główna	5
2. Wygląd strony logowania	6
3. Wygląd strony z głównego panelu	7
4. Wygląd strony z informacjami o repozytoriach	8
5. Strona z tworzeniem kontenera	9
6. Strona z uruchomionym kontenerem	10
7. Strona uruchomionym podglądem	11
8. Przykłady operacji na kontenerach	12
9. Strona z procesem płatności	13
10. Strona z włączonym sharelink do repozytorium	14
7. Opis poszczególnych klas, metod i zapytań REST	15
1. Opis interfejsów	15
1. Interfejs LinkClientInterface	15
2. Interfejs ContainersRepository	15
3. Interfejs ImagesRepository	16
4. Interfejs UsersRepository	16
3. Opis klas	17
1. Klasa HomeController	17
2. Klasa GithubClient	17
3. Klasa GithubRestController	18
4. Klasa PostLogin	19
5. Klasa LinkClient	19
6. Klasa LinkController	19

7.	Klasy Containers, Images i Users	19
8.	Klasa CodebinApplication	19
9.	Klasa SecurityConfig	20
4.	Opis zapytań REST	20
8.	Podsumowanie projektu	21

1. Opis projektu

Projekt Codebin powstał w celu wykorzystania możliwości nowych technologii połączonych razem z możliwością uruchamiania systemów linuxowych w chmurze za sprawą technologii OpenStack i Docker. Dodatkowym założeniem projektu było napisanie aplikacji webowej, która będzie komunikować się przy serwerem za pomocą komunikatów REST, które są szyfrowane za pomocą standardu TLS 1.2.

2. Wykorzystane technologie

W naszym projekcie wykorzystaliśmy najnowocześniejsze technologie dla osiągnięcia jak najlepszych walorów estetycznych aplikacji webowej i szybkości działania chłodzenia wodnego.

1. Użyte języki programowania oraz formaty tekstu

1. Apache Maven jako narzędzie do automatyzacji budowy projektu.
2. HTML z wykorzystaniem frameworka Bootstrap, JavaScript z frameworkiem jQuery oraz AngularJS.
3. Java z rozszerzeniem Spring Boot po stronie serwera.
4. JSON jako format tekstowy dla przejrzystego uporządkowania danych w pliku konfiguracyjnym.
5. Bash do konfiguracji systemów operacyjnych.
6. AlpineLinux jako podstawa do zbudowania własnego systemu operacyjnego.

2. Wykorzystane oprogramowanie przy projektowaniu i wdrażaniu projektu

1. Adobe XD CC w celu szybkiego prototypowanie interfejsu graficznego.
2. IntelliJ IDEA jako główne IDE do programowanie serwera.
3. JavaDoc do prowadzenia dokumentacji kodu.
4. JetBrains WebStorm jako IDE do interfejsu graficznego.
5. Putty do komunikacji z serwerem.

3. Użyte biblioteki w projekcie

1. Biblioteki zewnętrzne

LP.	Nazwa biblioteki	Licencja	Opis
1.	Apache Tomcat	Apache License 2.0	Kontener aplikacji webowych.
2.	JSON.simple	Apache License 2.0	Prosty zestaw narzędzi do obsługi plików JSON.
3.	log4j	Apache License 2.0	Biblioteka służąca do tworzenia logów podczas działania aplikacji.
4.	Spring Boot	GNU Lesser General Public	Framework Java dodający możliwość tworzenia webowych aplikacji.
5.	Angular.js	MIT	Framework wspomagający tworzenie i rozwój aplikacji internetowych na pojedynczej stronie.
6.	Bootstrap	MIT	Framework CSS dodający przydatne elementy do projektowania stron.
7.	Chart.js	MIT	Biblioteka dodaje możliwość sterowania diodami LED.
8.	Jquery	MIT	Biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu.
9.	Popper.js	MIT	Biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z modali.
10.	Font Awesome	Licencja wewnętrzna: https://fontawesome.com/license	Zestaw piktogramów przedstawionych w formacie czcionki.
11.	junit5	Eclipse Public License 2.0	Narzędzie służące do tworzenia powtarzalnych testów jednostkowych.
12.	Docker	MIT	System do wirtualizacji poszczególnych kontenerów
13.	OpenStack	MIT	System nadrzędnej wirtualizacji

Tabela 1 Biblioteki zewnętrzne wykorzystane w projekcie.

2. Biblioteki wewnętrzne

1. cookie.js – zarządzanie ciasteczkami.
2. fast.js – szybkie odczytywanie plików testowych z poziomu przeglądarki.

4. Funkcjonalność projektu

Serwerowa część opiera się na przekazywaniu zapytań REST za pomocą socketu unixowego do aplikacji Docker, natomiast część webowa zarządza kontenerami.

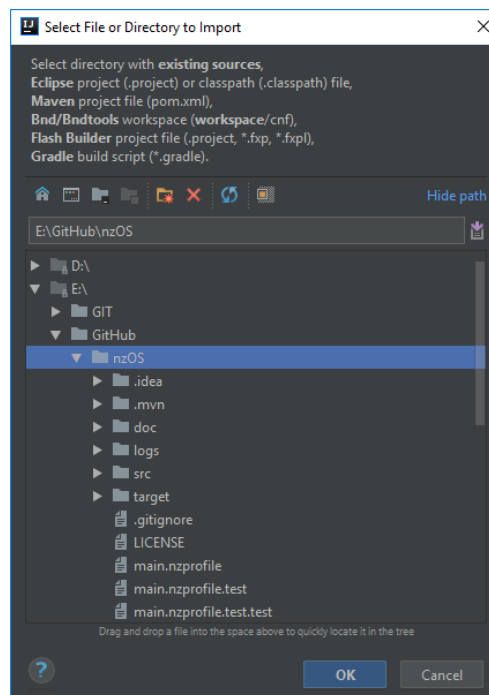
5. Obsługa projektu i sposób jego uruchomienia

1. Budowanie projektu

Aby móc wybudować projekt należy posiadać zintegrowane środowisko programistyczne np. IntelliJ IDEA.

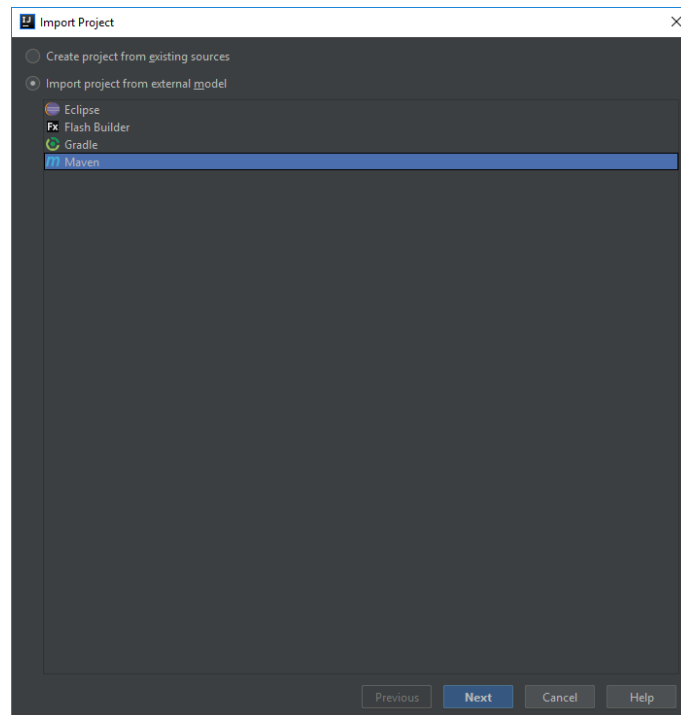
1. Proces kompilacji aplikacji

Pierwszym krokiem jest importowanie projektu z istniejących źródeł:

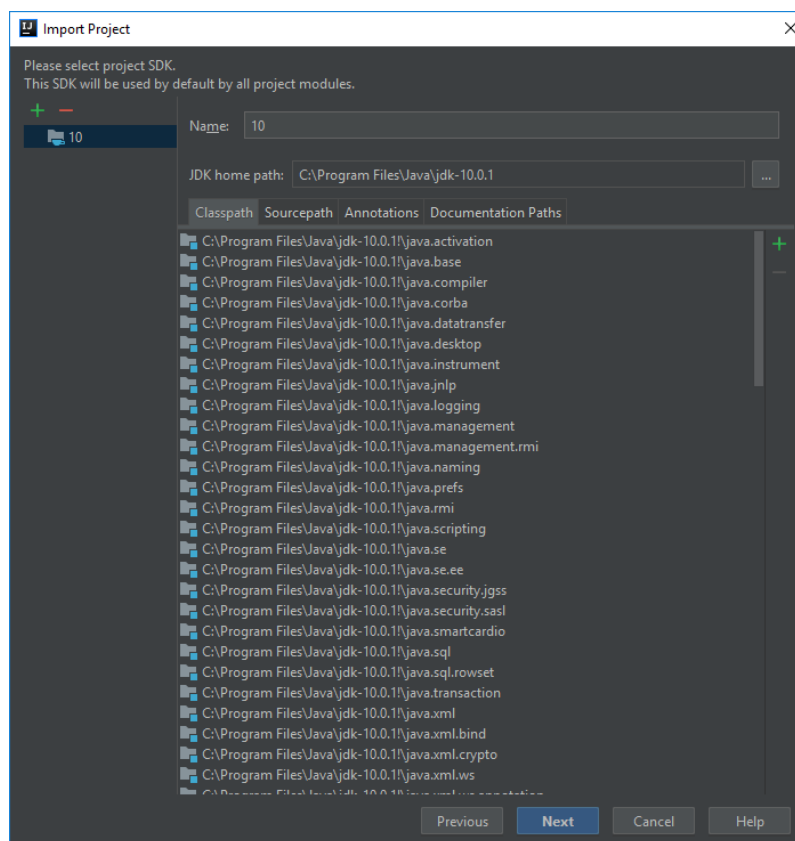


Rysunek 1 Poprawne wybranie folderu z istniejących źródeł.

Po skompilowaniu projektu należy użyć narzędzia **Maven**. Służy do zautomatyzowania procesu tworzenia zależności wymaganych do uruchomienia. Tworzy piaskownicę dla środowiska wykonawczego systemu Windows lub drzewo instalacji dla aplikacji pulpitu systemu Windows, które można łatwo dołączyć do pakietu instalacyjnego.



Rysunek 2 Wybranie narzędzia do tworzenia zależności.

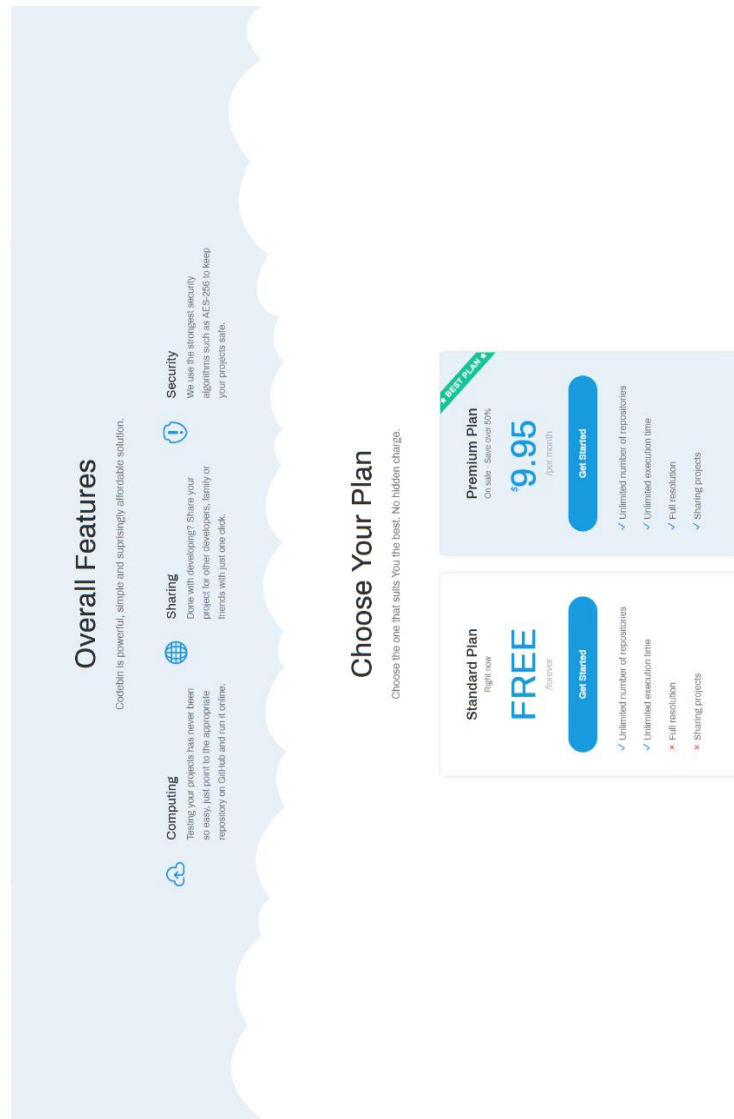


Rysunek 3 Wybór SDK dla projektu

Po tych trzech krokach powinniśmy uzyskać w pełni skompilowaną i gotową do uruchomienia aplikację.

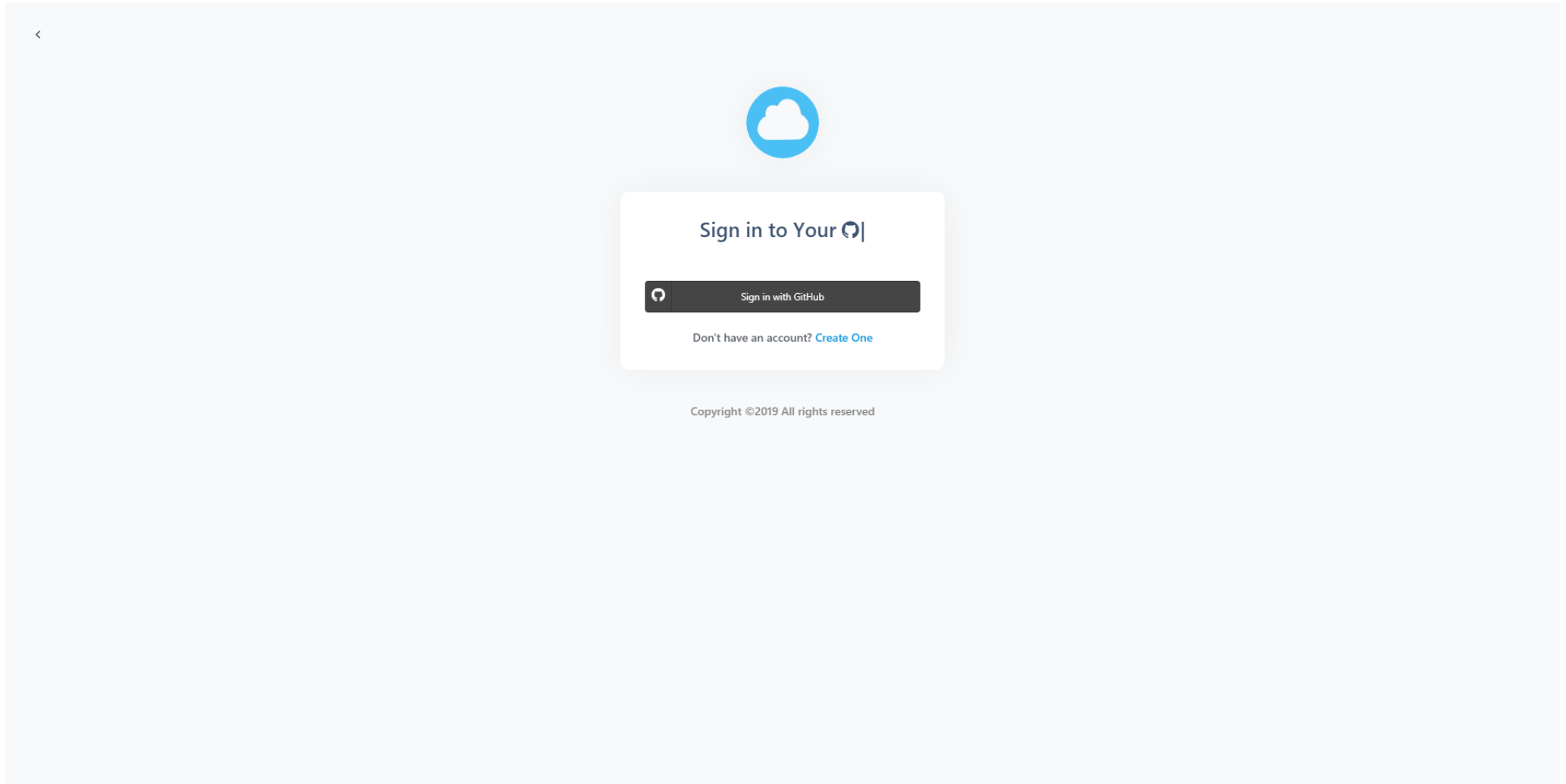


Codebin is a cloud platform capable of running over 14 leading technologies.



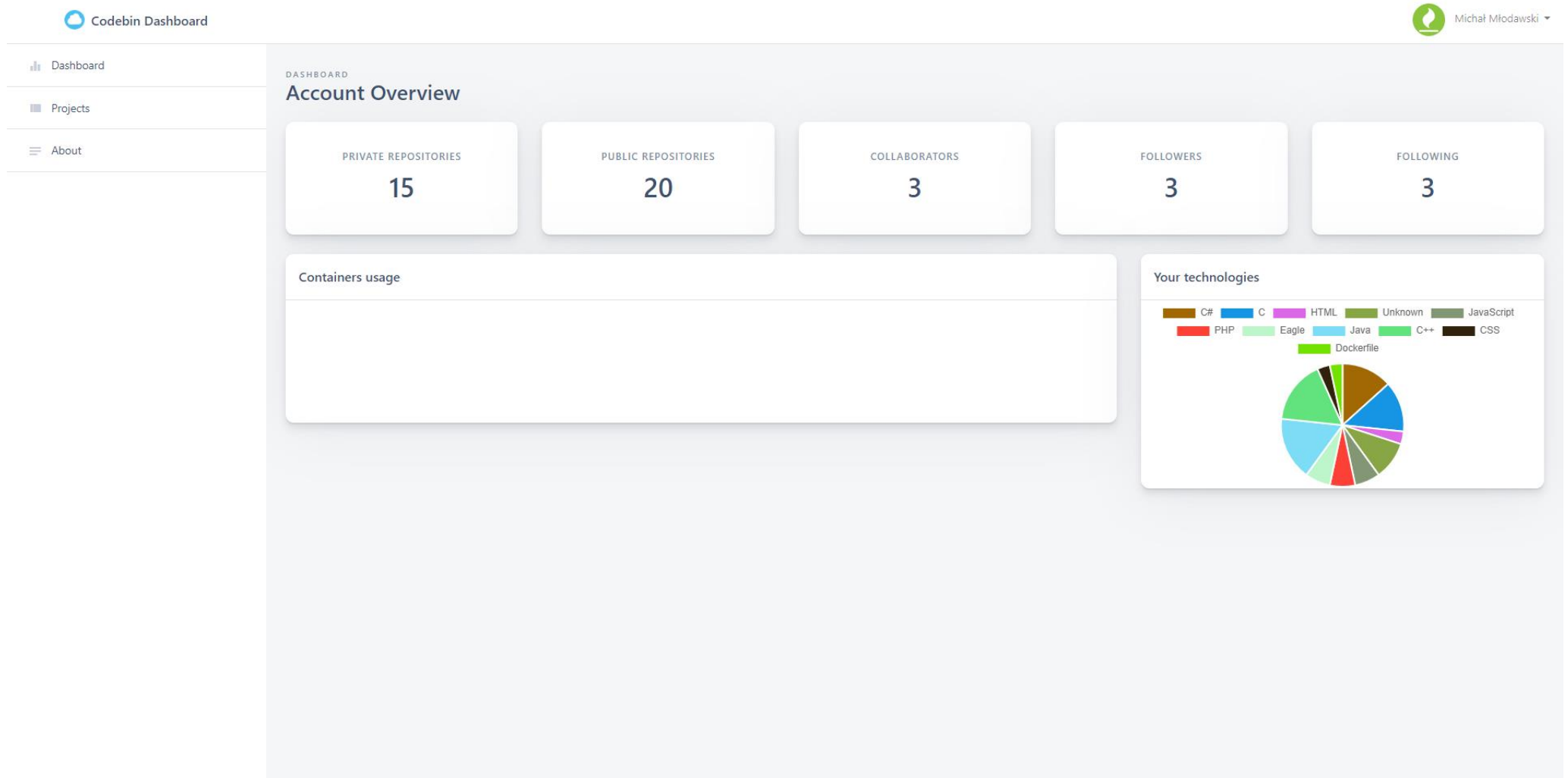
Rysunek 4 Wygląd strony głównej.

2. Wygląd strony logowania.



Rysunek 5 Wygląd strony z logowaniem do serwisu Github.

3. Wygląd strony z głównego panelu.



Rysunek 6 Wygląd strony z głównego panelu..

Codebin Dashboard

Dashboard

Projects

About

PROJECTS

Your GitHub projects

OnionEngine

Just a school project

C++

2018-12-12T08:13:42Z

Open

Alpine-noVNC

Part of the STV project, the basic files needed to run an Alpine system with Openbox and noVNC

Dockerfile

2018-04-20T13:48:02Z

Unresponsive

Bazy-Danych-1-Klient

Simple CRUD made in AngularJS and PHP

PHP

2018-01-30T10:00:52Z

Unresponsive

codebin-html

CSS

2018-05-20T11:54:18Z

Open

dinca

Repository with text of DMCA takedown notices as received. GitHub does not endorse or adopt any assertion contained in the following notices. Users identified in the notices are presumed innocent until proven guilty. Additional information about our DMCA policy can be found at

2018-07-05T13:17:02Z

Unresponsive

Fallout4-PipBoy

Library to use the TCP communication mechanism/protocol of the PipBoy Android/iOS app with .NET

C#

2018-10-07T19:28:18Z

Unresponsive

Heapport

Implementation of the heapport algorithm and methods enabling the use of the algorithm.

Java

2018-04-10T17:09:16Z

Open

IoT-Smart-Health

Web interface to the IoT Smart Health application

JavaScript

2018-03-07T13:18:10Z

Open

libjpeg

Independent JPEG Group's JPEG software

C

2018-07-08T18:11:05Z

Open

noVNC

VNC client using HTML5 (WebSockets, Canvas)

JavaScript

2018-04-19T08:10:42Z

Open

ncOS

Alternative cooling management system.

Java

2018-05-04T13:37:32Z

Open

Project-PC-master-control-center

Management of LED lighting from the PC application via UART. Written in C++ (QT) with Steam integration.

C#

2018-03-10T18:33:57Z

Unresponsive

Project-Titan-main-game

A simple multiplayer game using the Unity engine

C#

2017-04-20T13:23:02Z

Unresponsive

Project-Titan-User-identification

Application that allows to create a unique user ID

C#

2017-08-16T14:42:02Z

Unresponsive

Project-Titan-Web-Components

Titan project server APIs


PHP


2017-08-16T14:23:42Z


Unresponsive


Rysunek 7 Wygląd strony z informacjami o repozytoriach.


5. Strona z tworzeniem kontenera

 Codebin Dashboard

 Michał Młodawski ▾

 Dashboard

 Projects

 About

CONTAINERS

Creation of a new container

Heapsort

Implementation of the heapsort algorithm and methods enabling the use of the algorithm.

Java

2018-04-11T17:09:15Z

Basic information

User plan: Free
Exposed port: 8443
Ram memory: 1 GB
Disk space: 1 GB

Installation process:

Go to the container

CONTAINERS

11c8cd35ddf17775dc6b78b74594083ba974a6da90fa34fd6424eb59284695

General view

- Status: Free
- Visibility: Local access
- Creation time: 12-05-2019 11:19:33

Remove container

Execute commands

Restart container

Advertising

Current processes

UID	PID	STIME	TIME	CMD
root	23513	05:19	00:00:00	/usr/bin/python2 /usr/bin/supervisord -c /etc/supervisord.conf.d/supervisord.conf
root	23566	05:19	00:00:08	/usr/lib/firefox/firefox https://github.com/SimpleMethod
root	23567	05:19	00:00:00	xfce4-terminal -e bash -c 'ps aux; bash'
root	23568	05:19	00:00:00	bash /root/.nvm/nc/urls/launch.sh --nvc localhost:5900 --listen 8443 --cert /etc/cert.pem
root	23578	05:19	00:00:00	socket tcp-listen:6100 reuseaddr;fork unix:/tmp/X11-unix/0
root	23580	05:19	00:00:00	Xvfb :0 -screen 0 1280x720x24
root	23586	05:19	00:00:00	python /root/.nvm/nc/urls/websocket/run --web /root/.nvm/nc/urls/_/ --cert /etc/cert.pem 8443 localhost:5900
root	23596	05:19	00:00:00	dbus-launch --autolaunch=835551a132a617207c4e5c4cf887 --binary-syntax --close-stderr
root	23598	05:19	00:00:00	/usr/bin/dbus-daemon --fork --print-pid 5 --print-address 7 --session
root	23608	05:19	00:00:00	/usr/libexec/at-spi-bus-launcher
root	23612	05:19	00:00:00	/usr/bin/dbus-daemon --config-file=/usr/share/default/at-spi2/accessibility.conf --nofork --print-address 3
root	23614	05:19	00:00:00	/usr/libexec/at-spi2-registrid -use-gnome-session
root	23624	05:19	00:00:00	/usr/lib/xrdb/xrdbd/xtconf/xtconfd
root	23639	05:19	00:00:00	timd2
root	23640	05:19	00:00:00	/usr/bin/x11vnc -xkb --no-record --no-frees --no-frees --nondamage --display :0 --now --wait 50 --shared --permitFileTransfer --tightVnc
root	23641	05:19	00:00:00	/usr/bin/openssl --startup /usr/libexec/openssl/openssl.autostart OPENBOX
root	23644	05:19	00:00:00	bash -c ps aux; bash
root	23648	05:19	00:00:00	bash
root	23679	05:19	00:00:02	/usr/lib/firefox/firefox --contentproc -childID 1 -isOfBrowsers -bootstrap 3030 -stringPrefs 289-36:ff3d0d11-3623-4461-adca-d0a2e4287971 -schedulerPrefs 00012 -greennil /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -appdir /usr/lib/firefox/browser 8 tab
root	23756	05:19	00:00:01	/usr/lib/firefox/firefox --contentproc -childID 2 -isOfBrowsers -bootstrap 3030 -stringPrefs 289-36:ff3d0d11-3623-4461-adca-d0a2e4287971 -schedulerPrefs 00012 -greennil /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -appdir /usr/lib/firefox/browser 8 tab
root	23814	05:19	00:00:00	/usr/lib/firefox/firefox --contentproc -childID 3 -isOfBrowsers -bootstrap 3030 -stringPrefs 289-36:ff3d0d11-3623-4461-adca-d0a2e4287971 -schedulerPrefs 00012 -greennil /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -appdir /usr/lib/firefox/browser 8 tab
root	23946	05:20	00:00:00	bash -c java -jar /root/project/java/target/kopowanie-1.0-SNAPSHOT.jar; bash
root	23947	05:20	00:00:00	java -jar /root/project/java/target/kopowanie-1.0-SNAPSHOT.jar

Logs from the machine

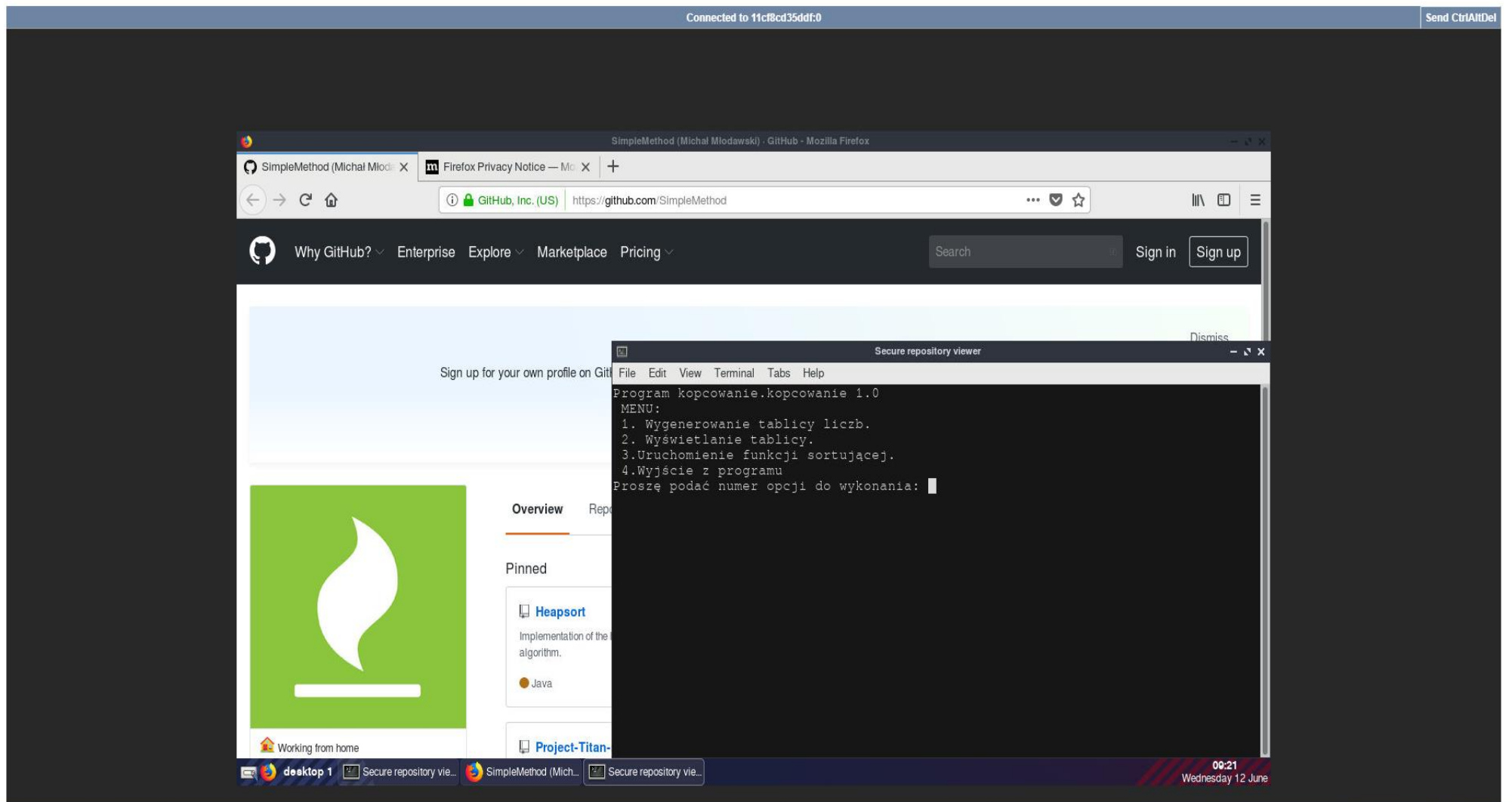
```
000000 2019-05-12 09:19:33:05:07: Super not running at root. Privileges were not dropped because no user is specified in the config file. If you intend to run at root, you can set user=root in the config file to avoid this message.
000000 2019-05-12 09:19:33:17: INFO Super not started with pod 1
000000 2019-05-12 09:19:34:00: INFO spawned: 'vnc' with pod 7
000000 2019-05-12 09:19:34:08: INFO spawned: 'firefox' with pod 8
000000 2019-05-12 09:19:34:32: INFO spawned: 'xfce4-terminal' with pod 9
000000 2019-05-12 09:19:34:20: INFO spawned: 'nvc' with pod 10
000000 2019-05-12 09:19:34:28: INFO spawned: 'X11vnc' with pod 11
000000 2019-05-12 09:19:34:27: INFO spawned: 'openssl' with pod 12
000000 2019-05-12 09:19:34:29: INFO spawned: 'ls' with pod 13
000000 2019-05-12 09:19:34:20: INFO spawned: 'ls' with pod 15
```

6. Strona z uruchomionym kontenerem

7. Strona uruchomionym podglądem

Local access

×



8. Przykłady operacji na kontenerach

Warning! ×

Are you sure?

Remove container

Close

Execute commands ×

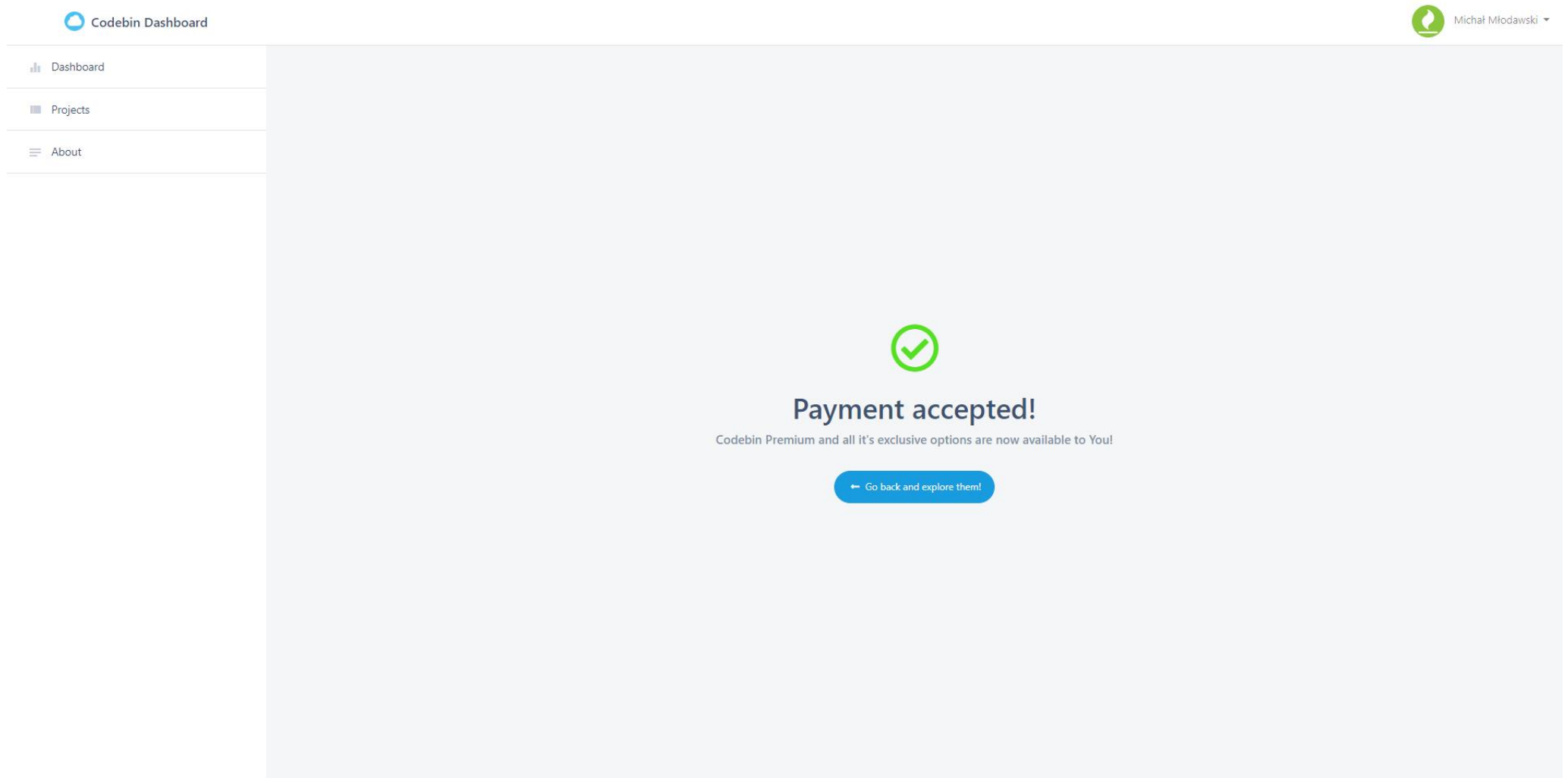
Path

Arguments

Execute

Close

9. Strona z procesem płatności



10. Strona z włączonym sharelink do repozytorium

Codebin | Container

Niebezpieczona | <https://127.0.0.1/share/Xg9HdclO+OowyQ3dJcX2dQ==>

Connected to 9e6eeat12fbb7:0

Secure repository viewer

```
File Edit View Terminal Tabs Help
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  19.0  0.2  19380 17764 ?        Ss   09:21   0:00 /usr/bin/python
root         8  32.0  0.9 1265144 75268 ?        Rl   09:21   0:00 /usr/lib/firefo
```

Smart Health - Mozilla Firefox

Smart Health | Firefox Privacy Notice — Mo | Smart Health

localhost:8080/doctor

dr Jan Nowak

SMART HEALTH

- Strona pacjenta
- Strona lekarza

Strona lekarza

Informacja o pacjencie

IP:

Pacjent:
Piotr Piasecki

Temperatura ciała: 0

Tętno: 0

Dawkowanie

Brak tabletek

Jedna tabletek

Dwie tabletki

Wykres funkcji życiowych

1

7.Opis poszczególnych klas, metod i zapytań REST

1. Opis interfejsów

1. Interfejs LinkClientInterface

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	String encrypt(String encryptText);	String encryptText – Wiadomość do zaszyfrowania	String – Wiadomość zaszyfrowana	Metoda służy do szyfrowania wiadomości
2.	String decrypt(String decryptText);	String decryptText – Wiadomość do rozszyfrowania	String – Wiadomość rozszyfrowana	Metoda służy do rozszyfrowania wiadomości

2. Interfejs ContainersRepository

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	List<Containers> removeByIdDocker(String name);	String name – nazwa kontenera do usunięcia	List<Containers> Zwraca listę kontenerów	Metoda służy do usunięcia kontenera po nazwie
2.	List<Containers> removeAllByIdDocker(String name);	String name – nazwa kontenerów do usunięcia	List<Containers> Zwraca listę kontenerów	Metoda służy do usunięcia wszystkich kontenerów po nazwie
3.	Containers getFirstById(Long id);	String name – nazwa kontenera do usunięcia	Containers – Zwraca znaleziony kontener	Metoda służy do znalezienia kontenera po ID
4.	Containers getFirstByIdDocker(String dockerId);	String dockerId – ID kontenera do usunięcia	Containers – Zwraca znaleziony kontener	Metoda służy do znalezienia kontenera po ID kontenera
5.	Containers getFirstByShareUrl(String shareURL);	String shareURL – Link do kontenera	Containers – Zwraca znaleziony kontener	Metoda służy do znalezienia kontenera po linku
6.	Containers getFirstByName(String name);	String name – nazwa kontenera	Containers – Zwraca znaleziony kontener	Metoda służy do znalezienia kontenera po nazwie
7.	List<Containers> getByHostPorts(Integer ports);	Integer ports – Numer portu	List<Containers> Zwraca listę znalezionych kontenerów	Metoda służy do znalezienia kontenerów po porcie
8.	List<Containers> getByIdDocker(String dockerId);	String dockerId – ID kontenera	List<Containers> Zwraca listę znalezionych kontenerów	Metoda służy do znalezienia kontenerów po ID kontenera

3. Interfejs ImagesRepository

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	Images getFirstById(Integer id);	Integer id – ID obrazu	Images – Zwraca obiekt obrazu	Metoda służy do wyszukiwania obrazu po ID
2.	Images getFirstByType(String type);	String type – Typ obrazu	Images – Zwraca obiekt obrazu	Metoda służy do wyszukiwania obrazu po typie
3.	List<Images> getById(Integer id);	Integer id – ID obrazu	List<Images> - Zwraca listę obiektów obrazu	Metoda służy do wyszukiwania obrazów po ID
4.	List<Images> getName(String name);	String name – Nazwa obrazu	List<Images> - Zwraca listę obiektów obrazu	Metoda służy do wyszukiwania obrazów po nazwie
5.	Images getFirstByName(String name);	String name – Nazwa obrazu	Images – Zwraca obiekt obrazu	Metoda służy do wyszukiwania obrazu po nazwie
6.	List<Images> getByType(String type);	String type – Typ obrazu	List<Images> - Zwraca listę obiektów obrazu	Metoda służy do wyszukiwania obrazów po typie

4. Interfejs UsersRepository

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	Users getFirstByGithub(Integer github);	Integer github – ID użytkownika githuba	Users – Obiekt użytkownika	Metoda służy do odnajdywania użytkownika po ID githuba
2.	Users getFirstById(Integer id);	Integer github – ID użytkownika	Users – Obiekt użytkownika	Metoda służy do odnajdywania użytkownika po ID
3.	Users getFirstByToken(String token);	Integer github – token użytkownika	Users – Obiekt użytkownika	Metoda służy do odnajdywania użytkownika po tokenie
4.	Users getFirstBySubscription(String subscription);	String subscription – subskrypcja użytkownika	Users – Obiekt użytkownika	Metoda służy do odnajdywania użytkownika po subskrypcji
5.	Users findById(String id);	String id – ID kontenera dockera	Users – Obiekt użytkownika	Metoda służy do odnajdywania kontenera dockera po ID

3. Opis klas

1. Klasa HomeController

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public String index()	brak	String – zwraca konkretny ciąg znaków	Metoda zwraca „index.html”
2.	public String errorPage()	brak	String – zwraca konkretny ciąg znaków	Metoda zwraca „404”
3.	public String dashboard()	brak	String – zwraca konkretny ciąg znaków	Metoda zwraca „dashboard”

2. Klasa GithubClient

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	protected org.json.JSONObject getCloneReposMinimalInfo(String token, String username, String reposName);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda służy do odebrania minimalnej ilości informacji o repozytorium
2.	protected org.json.JSONObject getReposContributorsStatistics(String token, String username, String reposName);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda służy do pobierania informacji o kontrybutorach repozytorium
3.	protected org.json.JSONObject getReposInfo(String token, String username, String reposName);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda służy do odebrania wszystkich informacji o repozytorium
4.	protected String getUserRepos(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda pobiera listę repozytoriów użytkownika
5.	protected org.json.JSONObject getUserInfo(String token, String username);	String token – Token do autoryzacji String username – Nazwa użytkownika	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda zwraca informacje o użytkowniku
6.	protected org.json.JSONObject getUserInfo(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda zwraca informacje o użytkowniku
7.	protected org.json.JSONObject getAccessToken(String code);	String code – kod otrzymany z Github api	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda tworzy token autoryzacyjny

3. Klasa GithubRestController

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public ResponseEntity getReposCloneWithMinimalInfo (String token, String username, String repos);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda zwraca potrzebne informacje o repozytorium
2.	public ResponseEntity getReposInfo(String token, String username, String repos);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda zwraca wszystkie informacje o repozytorium
3.	public ResponseEntity getReposContributorsStatistics(String token, String username, String repos);	String token – Token do autoryzacji String username – Nazwa użytkownika String reposName – Nazwa repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda służy do pobierania informacji o kontrybutorach repozytorium
4.	public ResponseEntity repos(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca listę repozytoriów z informacjami na ich temat
5.	public ResponseEntity getInfoAboutOwner(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca plik JSON z informacjami o użytkowniku
6.	public ResponseEntity checkUserPremium(String id);	String id – identyfikator użytkownika	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca informację z bazy danych o użytkowniku
7.	public ResponseEntity getInfoAboutUser(String token, String username);	String token – Token do autoryzacji String username – Nazwa użytkownika	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca plik JSON z informacjami o użytkowniku
8.	public ResponseEntity checkToken(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Metoda weryfikuje, czy profil istnieje
9.	public ResponseEntity getpublicrepos(String token);	String token – Token do autoryzacji	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca informacje na temat wszystkich repozytoriów zalogowanego użytkownika
10.	public ResponseEntity getPublicRepo(String token, String repold);	String token – Token do autoryzacji String repold – Identyfikator repozytorium	org.json.JSONObject – Obiekt z danymi w formacie JSON	Zwraca informację na temat pojedynczego repozytorium zalogowanego użytkownika

4. Klasa PostLogin

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public ResponseEntity postlogin(HttpServletRequest response response, String code);	HttpServletResponse response – odpowiedź służąca zapisowi ciasteczka String code – Kod autoryzacji otrzymany z GitHuba	ResponseEntity – Kod HTML	Metoda autoryzuje i zapisuje token w ciasteczku

5. Klasa LinkClient

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public String encrypt(String encryptText);	String encryptText – Wiadomość do zaszyfrowania	String – wiadomość zaszyfrowana	Metoda szyfruje podaną wiadomość
2.	public String decrypt(String decryptText);	String decryptText – Wiadomość do rozszyfrowania	String – wiadomość rozszyfrowana	Metoda rozszyfrowuje podaną wiadomość

6. Klasa LinkController

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public String getDecryptText(String decryptText, Model model);	String decryptText - wiadomość do rozszyfrowania model - ??	String – zwraca ciąg znaków „share”	??

7. Klasy Containers, Images i Users

Wszystkie metody w tych klasach to settery i gettery. Klasy posiadają też przeciążone funkcje ToString(), które zwraca stringi w formacie JSON.

8. Klasa CodebinApplication

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	public LinkClient linkClient();	brak	LinkClient – nowy obiekt typu LinkClient	Metoda tworzy i zwraca nowy obiekt
2.	public SrvClient srvClient();	brak	SrvClient – nowy obiekt typu SrvClient	Metoda tworzy i zwraca nowy obiekt
3.	public GithubClient githubClient();	brak	GithubClient – nowy obiekt typu GithubClient	Metoda tworzy i zwraca nowy obiekt
4.	public String string();	brak	String – pusty string	Metoda zwraca „”

9. Klasa SecurityConfig

LP.	Metoda:	Argument przyjmujący:	Argument zwracany:	Opis:
1.	protected void configure(HttpSecurity http);	HttpSecurity http - Konfiguracja zabezpieczeń	brak	Metoda służy do ustawiania zabezpieczeń HTTP
2.	private AuthorizationRequestRepository< OAuth2AuthorizationRequest> authorizationRequestRepository();	brak	AuthorizationRequestRepos itory<OAuth2Authorization Request> - ??	Metoda tworzy i zwraca obiekt typu HttpSessionOAU th2Authorizati onRequestRepos itory

4. Opis zapytań REST

LP.	Nazwa:	Opis	Odpowiedzi
1.	List containers	Zwraca listę kontenerów	200 – brak błędu 400 – błąd parametru 500 – błąd serwera
2.	Inspect a container	Zwraca informację niskiego poziomu o kontenerze	200 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
3.	Get container logs	Zwraca logi stdout i stderr z kontenera	101 – logi odebranu jako stream 200 – logi odebranu jako response body 404 – brak określonego kontenera 500 – błąd serwera
4.	List processes running inside a container	Odpowiednik komendy ps dla systemów Unix	200 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
5.	Create a container	Tworzy kontener	201 – kontener utworzony pomyślnie 400 – zły parametr 404 – brak określonego kontenera 409 - konflikt 500 – błąd serwera
6.	Remove a container	Usuwa kontener	204 – kontener usunięty pomyślnie 400 – zły parametr 404 – brak określonego kontenera 409 - konflikt 500 – błąd serwera
7.	Pause a container	Pauzuje wszystkie procesy wewnątrz kontenera	204 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
8.	Unpause a container	Wznawia pracę zapauzowanego kontenera	204 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
9.	Restart a container	Restartuje kontener	204 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
10.	Kill a container	Wysyła sygnał POSIX do kontenera, niszcząc go	204 – brak błędu 404 – brak określonego kontenera 409 – kontener nie pracuje 500 – błąd serwera

11.	Start a container	Uruchamia kontener	204 – brak błędu 304 – kontener już pracuje 404 – brak określonego kontenera 500 – błąd serwera
12.	Get container stats based on resource usage	Zwraca informacje o zużyciu zasobów przez kontener	204 – brak błędu 404 – brak określonego kontenera 500 – błąd serwera
13.	Delete stopped containers	Usuwa zapauzowane kontenery	204 – brak błędu 500 – błąd serwera
14.	List Images	Zwraca listę obrazów na serwerze	200 – brak błędu 500 – błąd serwera
15.	Create an exec instance	Wykonuje polecenie wewnątrz pracującego kontenera	201 – brak błędu 404 – brak określonego kontenera 409 – kontener jest zapauzowany 500 – błąd serwera
16.	Start an exec instance	Rozpoczyna zbiór wcześniej ustalonych poleceń	200 – brak błędu 404 – brak określonego kontenera 409 – kontener nie pracuje

8. Podsumowanie projektu

Podczas prac nad projektem natknęliśmy się na kilka problemów, takich jak trudność połączeniem technologii wirtualizacji z aplikacją w Javie, czy przeprowadzenie testów jednostkowych. Pracowaliśmy razem nad większością elementów projektu. Chciałbym podziękować społeczności Stack Overflow.