**Имена:**  Марио Ничев **фн:** 81167**Начална година:** *2014* **Програма: бакалавър, КН** **Курс: 4**   
**Тема: Million Dollar Homepage  
Дата: 2018-01-24 Предмет: wwwTech2017\_18\_9ed\_KN\_winter**

**имейл: marionitchev@gmail.com**

**преподавател:** доц. д-р Милен Петров

**Предаване:** Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използваниресурси**, архива да се казва 9999\_project\_final.zip. (Успех!). (Редактирайте маркираните зони в жълто с коректната информация)**

# ТЕМА: Million Dollar Homepage

## 1. Условие

Да се направи сайт подобен на сайта за „продаване“ на пиксели <http://milliondollarhomepage.com/>. Тъй като условито е малко общо, проектът наподобява функционалността на сайта <http://reddit.com/r/place> където резервирането на писксели се извършва с цената на време вместо пари.

## 2. Въведение

Приложението представлява прост HTML 5 Canvas върху който всеки потребител може да запазва пиксели с избран от тях цвят. В сайта [http://milliondollarhomepage.com](http://milliondollarhomepage.com/) потребителите плащат по един долар на пиксел и запазват банер реклами. Тъй като няма как потребителите да плащат за пикселите, този проект наподобява всъщност сайта <http://reddit.com/r/place> (който не е вече активен). В този сайт всеки потребител може да слага по един пиксел на споделен canvas на всеки 10 минути. За този подход обаче е нужен голям брой активни потребители и в тази имплементация потребителите могат да слагат неограничен брой пиксели, но пикселите се изтриват след фиксиран период от време.

## 3. Теория

За комуникация между клиента в браузъра и сървърва се използват HTTP заявки и конкретно сървърът придоставя REST API за заделяне на пиксели, създаване и удостоверяване на личността на потребители. Вътрешно сървърът запазва информацията за пикселите и потребителите в MySQL база данни. За да може потребителя бързо да нарисува всички пиксели в canvas-а, сървърът периодично създава .png изображение, което изпраща като отговор на GET /api/v1/pixels заявка.

За authentication сървърът хешира паролите на потребителите, след като ги регистрира. Когато един потребител влезе в профила си той получава authentication token който след това използва за да прави следващите си заявки към сървъра.

## 4. Използвани технологии

Сървърната част на проекта е реализирана на Java. Съответно използваните технологии са :

* Apache Tomcat 8 Server – сървърът върху който върви HTTP Java Servlet-ът.
* MySQL и JDBC connector – базата в която се запазва информацията за пикселите, потребителите и authentication token-ите.

Java библиотеките които са иползвани са

* Gson – библиотека за конвертиране на JSON обекти към Java обекти
* JAX-RS API – просто апи за улесняване на REST заявки на Java
* Guava – библиотека с много utility инструменти. Конкретно е използван AbstractScheduledService за периодично ъпдейтване на .png изображението.

За клиентската част са използвани просто CSS, Javascript и HTML без никакви специфични библиотеки.

## 5. Инсталация и настройки

Първа стъпка от инсталацията на проекта е да се построи самия java проект в .war архив. Тъй като е Maven проект това става с командата **mvn clean verify**, като .war файла ще бъде в създадената target директория. В предадения .zip архив вече има компилиран .war файл.

Втората стъпка е да се конфигурира Apache Tomcat 8 сървъра, като съответно той се разархивира на произволно място и .war файла се поставя в webapps поддиректорията. За да може да се иползва произволна MySQL база, проекта зависи от файла *configuration.json*, който трябва да се постави в root папката на Tomcat сървъра. Файлът има следния формат :

{  
 "driverClassName": "com.mysql.cj.jdbc.Driver",  
 "username": "user",  
 "password": "password",  
 "databaseName": "databaseName",  
 "databaseUrl": "jdbc:mysql://data/base/url",  
 "maxActiveConnections": 15,  
 "maxIdleConnections": 5,  
 "initialPoolSize": 5,  
 "validationQuery": "Select 1"  
}

Единствената зависимост към самата база данни е тя да има вече създадена база с име **databaseName**. За целта може да се използва скрипта *setup\_db.sql*, който единствено създава база с име Pixels.

След конфигурацията сървърът се стартира и спира съответно с нормалните Tomcat скриптове bin/startup.sh и bin/shutdown.sh.

## 6. Кратко ръководство на потребителя

REST API-то на сървъра отговаря на следните заявки :

**GET /api/v1/login**

Проверява дали потребител е подал правилни username и password.

REQUEST

Content-Type: application/json

Пример:

{  
 "username": "user",  
 "password": "password"  
}

RESPONSE

Content-type: text/plain

Пример:

4aa8c3e7-e3f4-4658-80d4-9e54b308b946

Status: 200 – Ако потребителя е дал правилни име и парола

Status: 400 – В противен случай

**POST /api/v1/user**

Регистрира нов потребител.

REQUEST

Content-Type: application/json

Пример:

{  
 "username": "user",  
 "password": "password"  
}

RESPONSE

Status: 200 – Ако потребителя се е създал успешно

Status: 400 – В противен случай

**POST /api/v1/pixels**

Маркира пиксели и ги регистрира на даден потреител. Потребителя трябва да сложи Autherization Header-ът и да сложи в него стойност Bearer <auth-token>.

REQUEST

Content-Type: application/json

Пример:

{  
 "username": "user",  
 "pixels": [{  
 "x": 10,  
 "y": 20,  
 "red": 255,  
 "green": 0,  
 "blue": 10  
 },  
 {  
 "x": 25,  
 "y": 15,  
 "red": 167,  
 "green": 127,  
 "blue": 0  
 }]  
}

RESPONSE

Status: 200

**GET /api/v1/pixels**

Връща текущия .png файл, който се ъпдейтва от базата на всеки 5 секунди.

RESPONSE

Content-type: image/png

Status: 200

Обикновения потребител трябва да използва html страниците за да използва сайта, но REST endpoint-ите са открити и могат да се викат от всички.

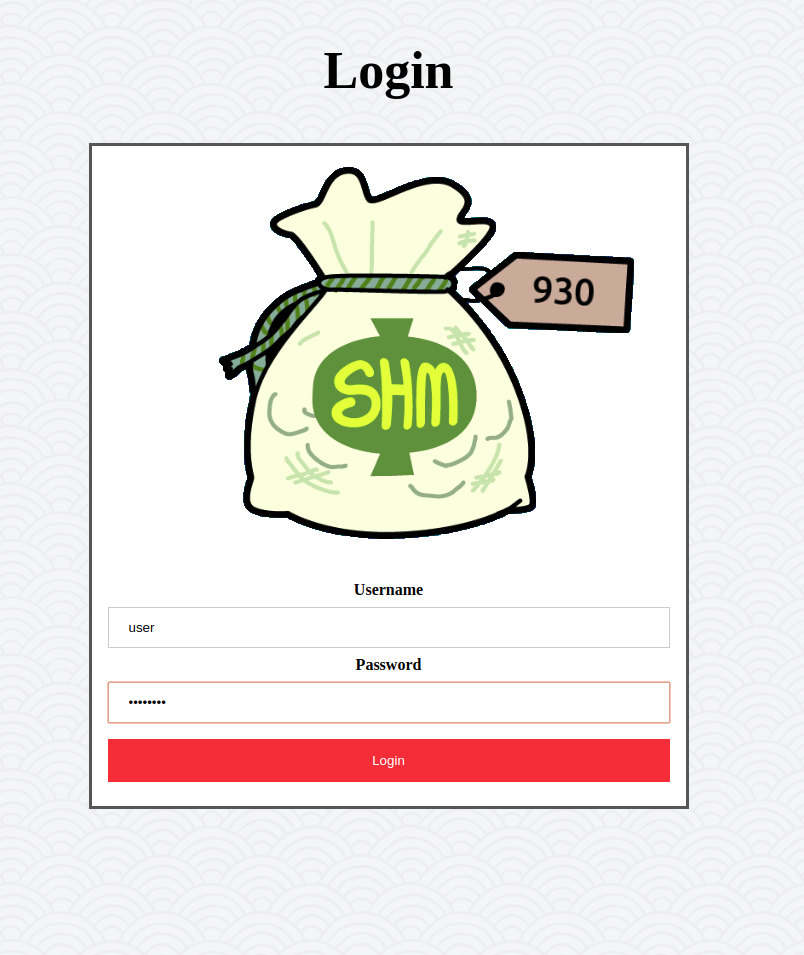
## 7. Примерни данни

Сървърът има три html страници с които потребителя може да взаимодейства със сайта:

* register.html – формата за регистрация на нови потребители
* login.html – формата за логване на потребители. Тази форма също така запазва 2 cookie-та след логване: Autherization header-a и username на потребителя.

## 

## **Фиг. 1** Регистрационна форма



## **Фиг. 2** Логин форма

## **Фиг. 3** Главната страница на сайта

## 8. Описание на програмния код

Архитектурата на проекта се състои от един Servlet обект който приема HTTP заявки и един ApplicationController който изпълнява тези заявки. В ApplicationController-а има два DataAccessObjects, които директно говорят с базата използвайки JDBC. Още един обект ImageUpdateScheduler периодично (през 5 секунди) създава .png изображение от пикселите в базата, което след това се връща при REST заявка.

## 9. Приноси на студента, ограничения и възможности за бъдещо разширение

Проектът може да бъде разширен, като се добави възможността да се добавят директно изображения като отделни пиксели в главната html страница. Освен това би било полезно, ако при hoover-ване на мишката върху вече зает пиксел, да се изобразява името на потребителя, който го е заел.

## 10. Какво научих

Научих да да работя бързо, като разделям модуларно работата си. Да използвам нови технологии за изграждане на сайтове, с което не съм имал много опит. Проектът също ме научи как да създавам REST API, което е доста гъвкав подход за създаване на уеб приложения.

## 11. Използвани източници

1. <http://cxf.apache.org/>- Apache CXF Servlet
2. <http://tomcat.apache.org/>- Apache Tomcat
3. <https://dev.mysql.com/doc/>- MySQL
4. <http://www.restapitutorial.com/> - REST API Tutorial

Предал: ………………………….

/*фн, имена, спец., група*/

Приел: ………………………….

/доц. *Милен Петров*/