

Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота

з дисципліни: «Інтеграційні програмні системи »

Виконали:
студенти 4 курсу
ФІОТ гр. ІО-41
Демчук Дмитро
Щур Вадим
Подзірей Ярослав
Чалій Андрій

Київ 2017

1. Короткий опис проекту

У даній роботі розроблено програму – систему KPI-sale (платформа для надання послуг та продажу товарів). Створена програма дозволяє виставити товари та послуги у відкритий доступ, для подальшого продажу. За допомогою цієї програми користувач може додати свій товар/послугу у відповідному вікні, але для цього потрібно попередньо зареєструватись на сайті. Після схвалення адміністратора, товар потрапляє на головну сторінку. Задля безпеки, для того щоб купити товар чи послугу також потрібно зареєструватись на ресурсі. Покупець може вибрати відповідний товар чи послугу яка його цікавить на головній сторінці. Також діє сортування за категоріями.

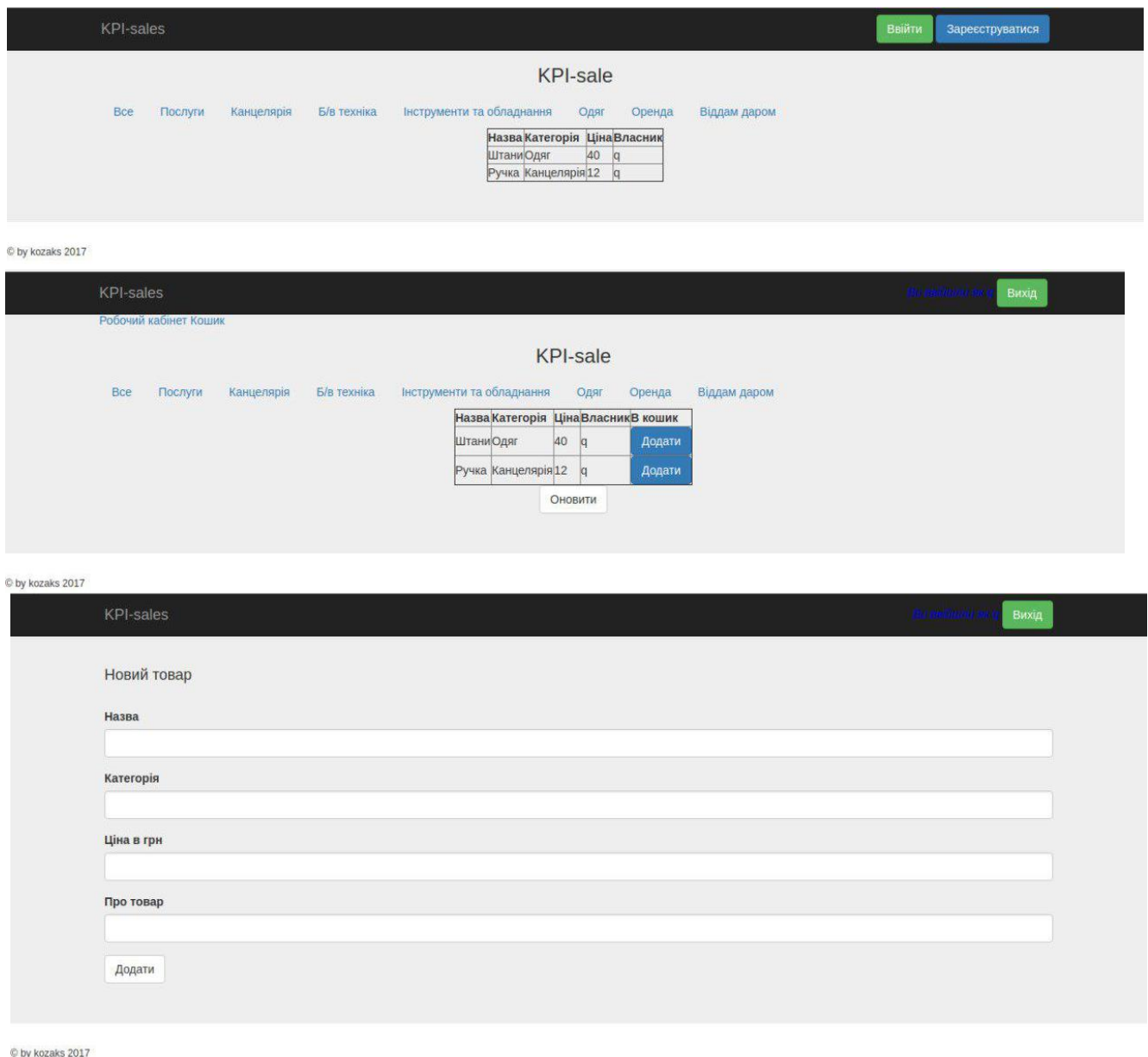


Рис.1 Скріншоти інтерфейсу додатку

Робота з сервером відбувається через Node.js за допомогою модулів ExpressJS, а для бази даних - mongoose.

Для авторизації використовується Passport, а для шифрування паролів Bcrypt. Для авторизації користувач повинен бути зареєстрований на сайті. Для реєстрації він повинен пройти класичне заповнення форм з іменем, email, паролем. У випадку успішної авторизації, при наступному запуску сайту користувачеві вже не потрібно вводити логін та пароль, доки не буде натиснута кнопка Logout, бо зашифовані логін і пароль зберігаються у cookie браузера.

Примітка: Проект реалізований не повністю. (немає стилів для оформлення, і також фото товарів/послуг, проте основний функціонал – працює. Вигляд в подальшому легко коректується після написання відповідним формам стилів).

2. Система автоматизації збірки

У якості системи для автоматизації збірки проекту використовується Gulp – інструмент збірки веб-додатку, що дозволяє автоматизувати задачі, що повторюються. В нашому випадку – це запуск тестів, виклик JS і CSS файлів. Для прикладу ми запускаємо таск для app.js, інші файли у нашому випадку запускати недоцільно. Програмне забезпечення використовує командну строку для запуску задач, визначених у файлі Gulpfile.

```
dima@dima-Aspire-E1-570G ~/rgr $ git add .
dima@dima-Aspire-E1-570G ~/rgr $ git commit -m "add tests1"
[master 6d2d703] add tests1
1 file changed, 1 insertion(+), 1 deletion(-)
dima@dima-Aspire-E1-570G ~/rgr $ git push origin master
Username for 'https://github.com': loneagle
Password for 'https://loneagle@github.com':
Підрахунок об'єктів: 3, виконано.
Delta compression using up to 4 threads.
Стиснення об'єктів: 100% (3/3), виконано.
Запис об'єктів: 100% (3/3), 283 bytes | 0 bytes/s, виконано.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com:loneagle/rgr.git
 a7929d4..6d2d703 master -> master
dima@dima-Aspire-E1-570G ~/rgr $ gulp minify
[11:11:11] Using gulpfile ~/rgr/gulpfile.js
[11:11:11] Starting 'minify'...
[11:11:11] Finished 'minify' after 13 ms
dima@dima-Aspire-E1-570G ~/rgr $
```

Рис.2 Збірка проекту інструментом Gulp

Взаємодія між частинами програми реалізується через оператор .pipe (), виконуючи по одній задачі за раз, не торкаючись вихідних файлів, до кінця процедури. Це дає можливість комбінації плагінів в будь-якій послідовності та кількості.

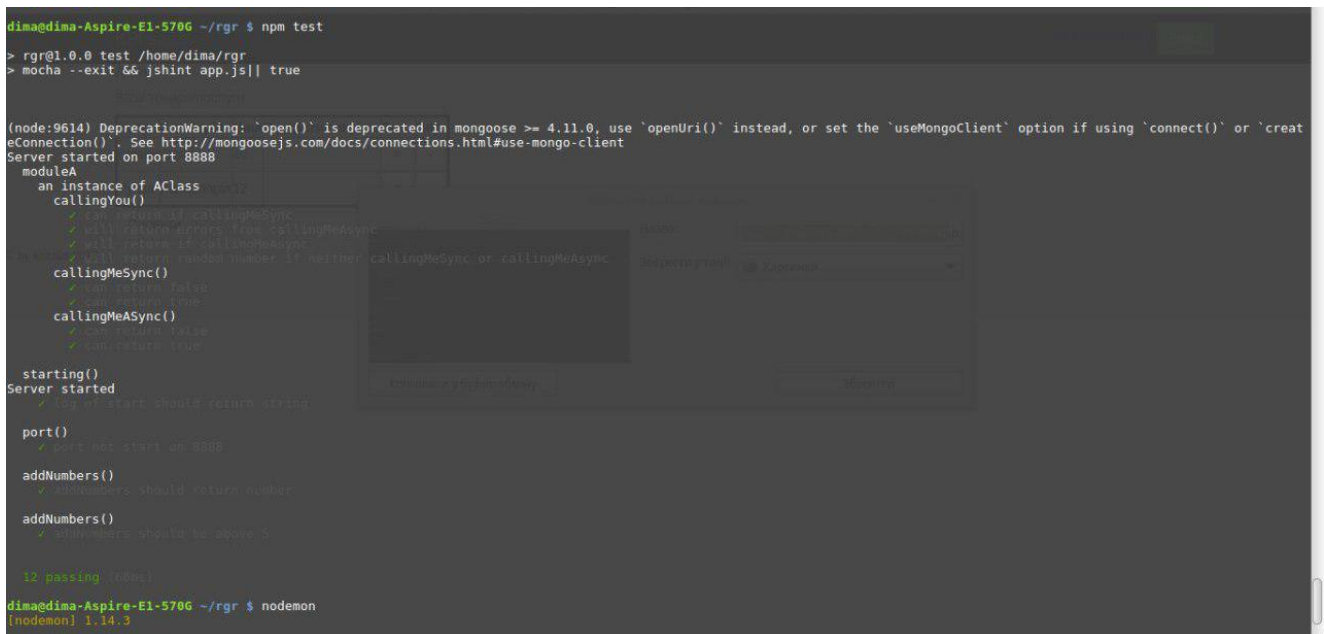
Так само в Gulp удосконалюється система збірки. Це означає, що, крім запуску задач, можна також копіювати файли з місця на місце, скопіювати і

розгортати проект у новому оточенні.

3. Безперервна інтеграція

Безперервна інтеграція - це практика розробки програмного забезпечення, яка полягає в об'єднанні робочих копій у спільну основну гілку розробки та виконанні автоматизованих збірок проектів для швидкого виявлення потенційних дефектів та вирішення інтеграційних проблем. Перехід до безперервної інтеграції дозволяє знизити трудомісткість інтеграції та зробити її більш прогнозованою за рахунок швидкого виявлення та усунення помилок і, але основною перевагою є скорочення вартості виправлення дефектів за рахунок раннього їх виявлення.

Для безперервної інтеграції створено тести для перевірки роботи навігації на стороні клієнта та відображення інформації.



```
dima@dima-Aspire-E1-570G ~/rgr $ npm test
> rgr@1.0.0 test /home/dima/rgr
> mocha --exit && jshint app.js|| true

(node:9614) DeprecationWarning: 'open()' is deprecated in mongoose >= 4.11.0, use 'openUri()' instead, or set the 'useMongoClient' option if using 'connect()' or 'createConnection()' - See http://mongoosejs.com/docs/connections.html#use-mongo-client
Server started on port 8888
  moduleA
    an instance of AClass
      callingYou()
        ✓ can return id callingMeSync
        ✓ will return errors from callingMeAsync
        ✓ will return if callingMeSync
        ✓ will return id callingMeSync or callingMeAsync
      callingMeSync()
        ✓ can return false
        ✓ can return true
      callingMeAsync()
        ✓ can return false
        ✓ can return true
  starting()
  Server started
    ✓ log of start should return string
  port()
    ✓ port has start on 8888
  addNumbers()
    ✓ addNumbers should return number
  addNumbers()
    ✓ addNumbers should be even 5
  12 passing (100s)

dima@dima-Aspire-E1-570G ~/rgr $ nodemon
[nodemon] 1.14.3
```

Рис.3 Тестування коду

Також для реалізації безперервної інтеграції використовувався розподілений веб-сервіс Travis CI.

На Travis CI виконуються можуть виконуватись наступні скрипти:

"start": "node app.js",

"test": "mocha --exit && jshint app.js"— запускає усі тести проекту; запускає статичний аналізатор коду.

4. Експоненціальна витримка

Експоненціальна витримка створена в компоненті `app.js` та використано для запиту при реєстрації на сервері. На графіку відображено залежність від номеру невдалого запиту та інтервалу затримки

```
function(router,url,data,options,max,delay,callback)
{ let t = this;
  router.post(url,data,options).then(response =>{
    callback(response);
  },err =>{
    if (max > 0) {
      console.log('Problem with
connect'); setTimeout(function() {
        t.exponentialBackoff(router,url,data,options, --max, delay * 2, callback);
      }, delay);
    }else {
      console.err('Error');
    }
  })
}
```

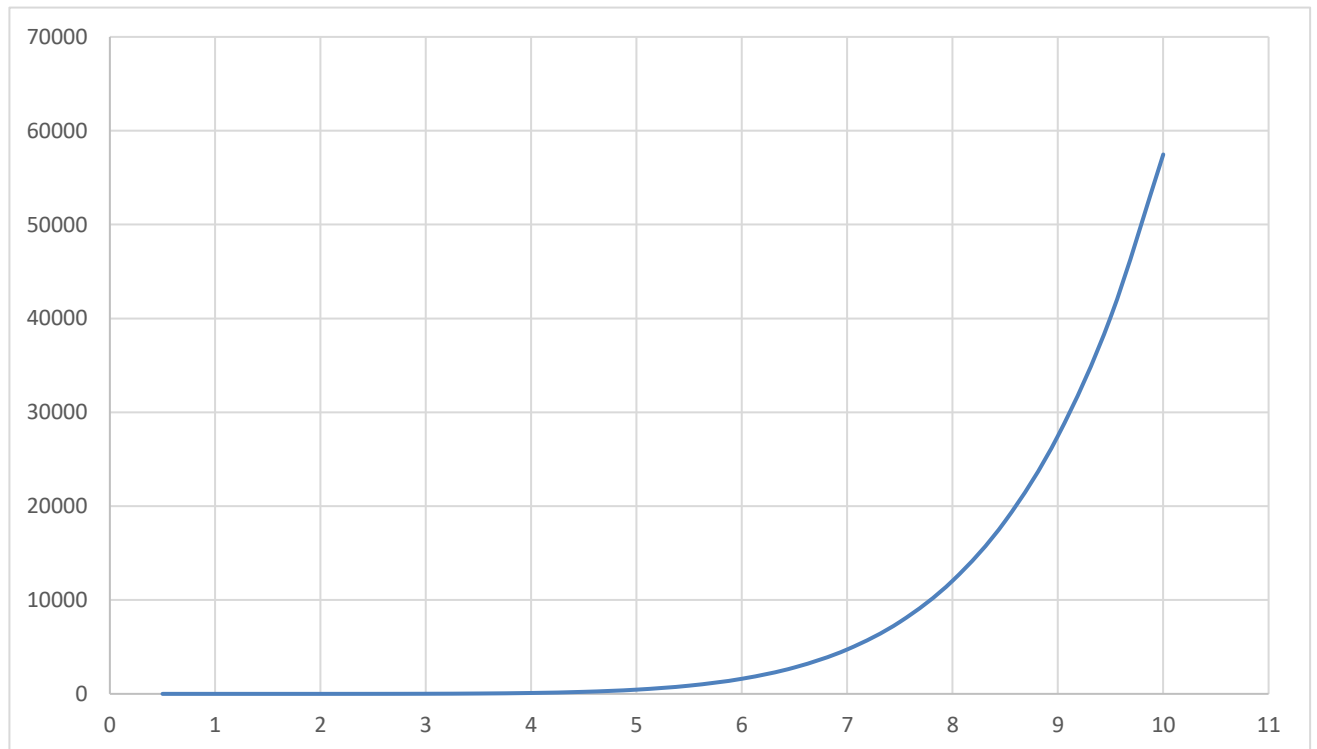


Рис.4. Графік експоненціальної витримки