ДОДАТОК А

Програмні коди

Лістинг модулей

```
     @Controller
     public class MainController {
         private final JacksonFactory jsonFactory =
JacksonFactory.getDefaultInstance();
         private final HttpTransport httpTransport = new
NetHttpTransport();
     @RequestMapping(value = "/", method = RequestMethod.GET)
     public ModelAndView index() throws Exception {
             ModelAndView view = new ModelAndView("index");
             getDropBoxFiles(view);
             getGoogleDriveFiles(view);
             return view;
     }

     @RequestMapping(value = "/downloadDbx/{fileId}", method =
RequestMethod.GET)
     public void downloadDbx(@PathVariable(value = "fileId") String
fileId, ttpServletResponse response) throws Exception {
     DbxRequestConfig config = new DbxRequestConfig("SpaceDrive",
"en_US");
     DbxClientV2 client = new DbxClientV2(config, ACCESS_TOKEN_Dbx);
DbxFiles.Metadata metadata = client.files.getMetadata(fileId);
     InputStream inputStream =
client.files.downloadBuilder(metadata.pathLower).start().body;
     String headerKey = "Content-Disposition";
     String headerValue = String.format("attachment; filename=\"%s\"",
metadata.name);
             response.setHeader(headerKey, headerValue);
             OutputStream outStream = response.getOutputStream();
             IOUtils.copy(inputStream, outStream);
         }
     @RequestMapping(value = "/upload", method = RequestMethod.POST)
         public String upload(@RequestParam("file") MultipartFile
multipartFile) throws Exception {
             GoogleCredential credential = new
GoogleCredential().setAccessToken(ACCESS_TOKEN_GDrive);
             Drive service = new Drive.Builder(httpTransport, jsonFactory,
credential).setApplicationName("SpaceDrive").build();
     DbxRequestConfig config = new DbxRequestConfig("SpaceDrive",
"en_US");
     DbxClientV2 client = new DbxClientV2(config, ACCESS_TOKEN_Dbx);
     if (((service.about().get().execute().getQuotaBytesTotal() -
service.about().get().execute().getQuotaBytesUsed())
                     +
(client.users.getSpaceUsage().allocation.getIndividual().allocated -
client.users.getSpaceUsage().used)) < multipartFile.getSize()) {
                 return "redirect:/";
             }

      InputStream inputStream = multipartFile.getInputStream();
     java.io.File file = new
java.io.File(multipartFile.getOriginalFilename());
```

```
     try (FileOutputStream out = new FileOutputStream(file)) {
                 IOUtils.copy(inputStream, out);
             }
             boolean movedToDx = false;
             int partCounter = 1;
             int sizeOfFiles = (int) (multipartFile.getSize() / 2);
             byte[] buffer = new byte[sizeOfFiles];
             try (BufferedInputStream bis = new BufferedInputStream(
                     new FileInputStream(file))) {
                 String name = file.getName();
                 int tmp = 0;
                 while ((tmp = bis.read(buffer)) > 0) {
           java.io.File newFile = new java.io.File(name + "."        +
String.format("%03d", partCounter++));
     try (FileOutputStream out = new FileOutputStream(newFile)) {
                         out.write(buffer, 0, tmp);
                     }
     if (!movedToDx) {
          client.files.uploadBuilder("/" + newFile.getName()).run(new
FileInputStream(newFile));
           movedToDx = true;
           } else {
              File body = new File();
              body.setTitle(newFile.getName());
              body.setMimeType(multipartFile.getContentType());
              FileContent mediaContent = new
FileContent(multipartFile.getContentType(), newFile);
              service.files().insert(body, mediaContent).execute();
              }
     }
        }
        return "redirect:/";
     }
     @RequestMapping(value = "/moveDbx/{fileId}", method =
RequestMethod.GET)
     public String moveToDbx(@PathVariable("fileId") String fileId) throws
Exception {
             GoogleCredential credential = new
GoogleCredential().setAccessToken(ACCESS_TOKEN_GDrive);
             Drive service = new Drive.Builder(httpTransport, jsonFactory,
credential).setApplicationName("SpaceDrive").build();
             InputStream inputStream =
service.getRequestFactory().buildGetRequest(new
GenericUrl(service.files().get(fileId).execute().getDownloadUrl())).execute
().getContent();
     DbxRequestConfig config = new DbxRequestConfig("SpaceDrive",
"en_US");
             DbxClientV2 client = new DbxClientV2(config,
ACCESS_TOKEN_Dbx);
             client.files.uploadBuilder("/" +
service.files().get(fileId).execute().getOriginalFilename()).run(inputStrea
m);
             service.files().delete(fileId).execute();
             return "redirect:/";
         }
         @RequestMapping(value = "/deleteGDrive/{fileId}", method =
RequestMethod.GET)
         public String deleteGDrive(@PathVariable("fileId") String fileId)
```

```
throws Exception {
              GoogleCredential credential = new
GoogleCredential().setAccessToken(ACCESS_TOKEN_GDrive);
              Drive service = new Drive.Builder(httpTransport, jsonFactory,
credential).setApplicationName("SpaceDrive").build();
              service.files().delete(fileId).execute();
              return "redirect:/";
          }
          private void getDropBoxFiles(ModelAndView view) throws Exception
{
      DbxRequestConfig config = new DbxRequestConfig("SpaceDrive",
"en_US");
       DbxClientV2 client = new DbxClientV2(config, ACCESS_TOKEN_Dbx);
              view.addObject("DbxName", "Dropbox");
              Map<String, String> result = new HashMap<>();
              ArrayList<DbxFiles.Metadata> entries =
client.files.listFolder("").entries;
              for (DbxFiles.Metadata entry : entries) {
                  result.put(entry.name, ((DbxFiles.FileMetadata)
entry).id);
              }
              view.addObject("DbxFiles", result);
              view.addObject("DbxTotal",
client.users.getSpaceUsage().allocation.getIndividual().allocated / (1024 *
1024));
              view.addObject("DbxUsed", client.users.getSpaceUsage().used /
1024);
          }

          private void getGoogleDriveFiles(ModelAndView view) throws
Exception{
              GoogleCredential credential = new
GoogleCredential().setAccessToken(ACCESS_TOKEN_GDrive);
              Drive service = new Drive.Builder(httpTransport, jsonFactory,
credential).setApplicationName("SpaceDrive").build();
              Map<String, String> result = new HashMap<>();
              Drive.Files.List request = service.files().list();
              List<com.google.api.services.drive.model.File> list =
request.execute().getItems();
              for (File file : list) {
                  if (isNotCyrillic(file)) {
                      result.put(file.getTitle(), file.getId());
                  }
              }
              view.addObject("GDriveName", "Google Drive");
              view.addObject("GDriveFiles", result);
              view.addObject("GDriveTotal",
service.about().get().execute().getQuotaBytesTotal() / (1024 * 1024));
              view.addObject("GDriveUsed",
service.about().get().execute().getQuotaBytesUsed() / 1024);
          }
      }
```

ДОДАТОК Б

Слайди презентації

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження методів впровадження технологічних стеків при розробці веб-додатків

Керівник:
проф. Шубін І. Ю.

Виконала:
ст. гр. ІПЗмзд-18-1 Радіонова А.М.

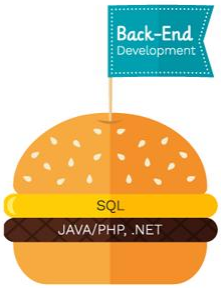
1

Мета роботи

- Дослідження принципів розробки сучасних веб-застосунків на основі «технологічного стеку»
- Анализ и разработка возможностей применения их в платформах облачных хранилищ
- Разработка интеграционной платформы для облачных хранилищ
- Дослідження способів і алгоритмів, що використовуються при інтеграції хмарних сховищ
- Дослідження можливостей їхнього використання в інтеграційній платформі
- Розробка інтеграційної платформи для хмарних сховищ

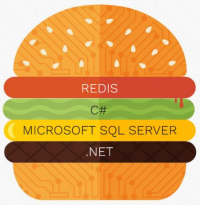
2

Основні визначення і цілі створення технологічних стеків



Back-end

Full-stack



Технологічні стеки

3

Використання хмарних сховищ

- Синхронізація файлів між декількома пристроями
- Створення резервних копій файлів
- Доступ до файлів з будь-якого місця, де є підключення до мережі інтернет
- Обмін файлами між користувачами

4

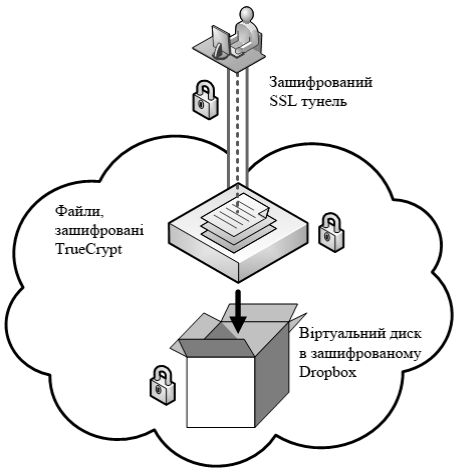
Переваги хмарних сховищ

- Можливість використання на малопотужних ПК
- Мінімізація витрат і збільшення продуктивності IT інфраструктури
- Легкість і простота в обслуговуванні
- Зниження витрат на придбання ПЗ
- Можливість роботи при наявності будь-якої операційної системи

5

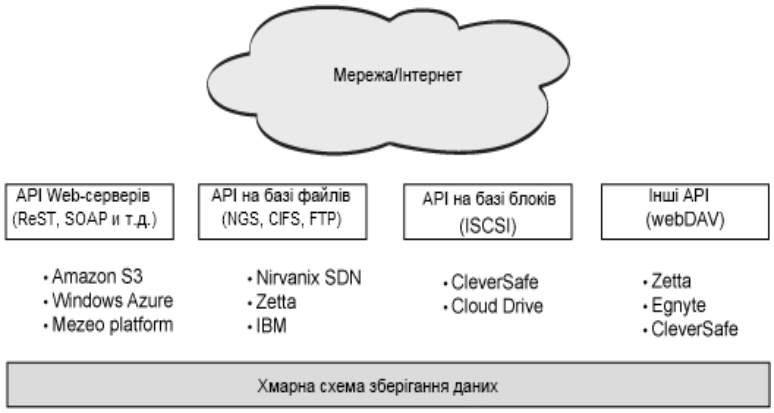
Ризики використання

- Конфіденційність
- Доступність
- Цілісність



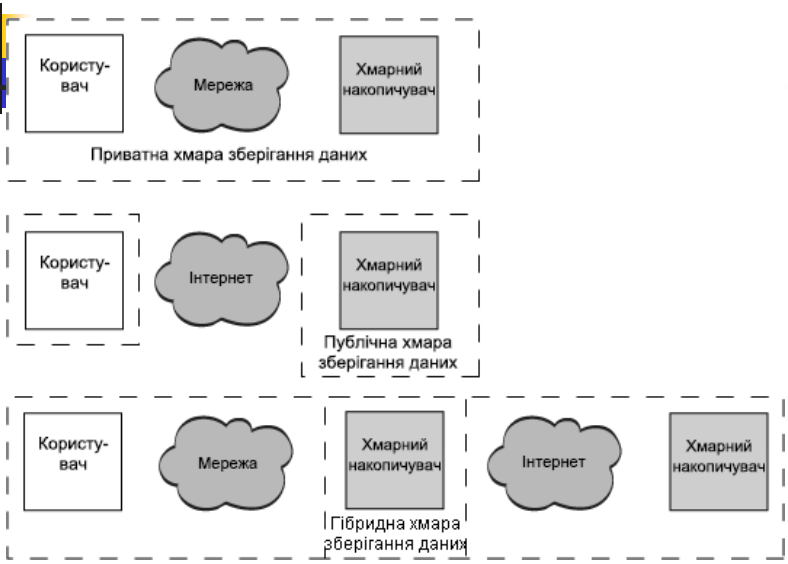
6

Методи доступу до хмарних систем зберігання даних



7

Хмарні моделі зберігання данних



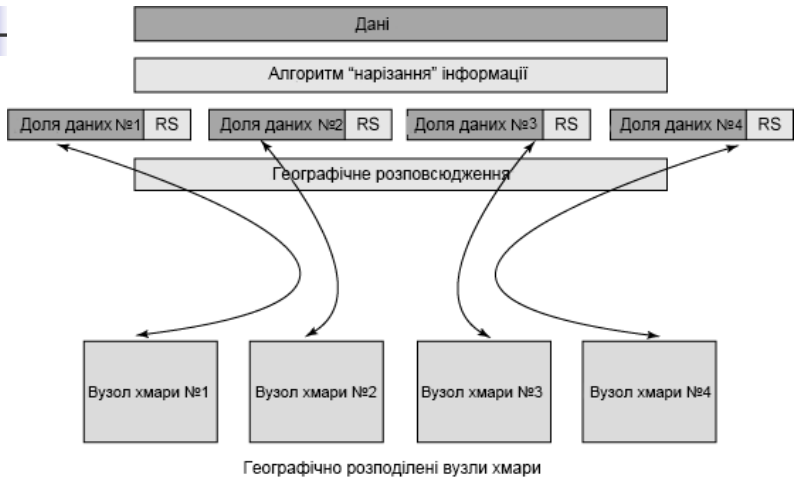
8

Переваги інтеграційної платформи

- «Гармонізація» даних - синхронізація, реплікація і інші операції над даними в різних системах
- інструменти для забезпечення захищеного доступу до даних в будь-якому місці
- гнучке масштабування ресурсів
- зниження сукупної вартості володіння

9

Підхід Cleversafe до забезпечення високої готовності даних



10

Алгоритм балансування навантаження в хмарному сховищі

- Для оптимізації механізмів доступу до даних розроблено загальну модель доступу до даних системи зберігання.
- R = (U,M,Q),
  де U = {u1,u2,…} – множина користувачів;
  M = {m1,m2,…} – множина унікальних елементів даних, що розміщені на пристроях зберігання.

  При цьому мінімальною одиницею даних m_i вважаємо файл, що має обов'язкову властивість h – розмір.
- Отже, функція розподілення елементів даних по пристроям зберігання має вигляд

P: Mc → D.

11

Функція цілі

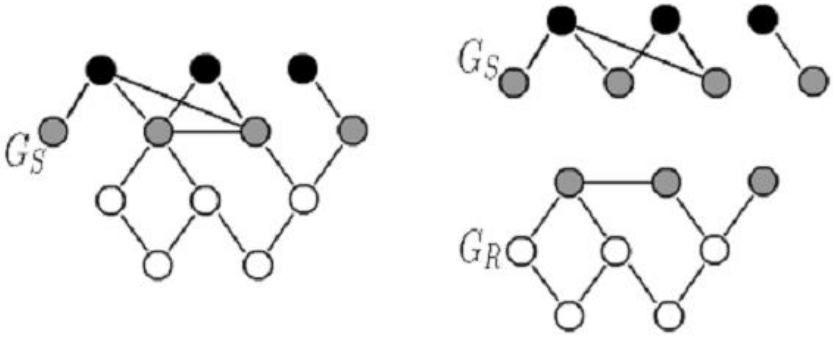
$$\sum_{i=1}^{N} Pi(t) \rightarrow \min$$

$$\sum_{i=1}^{N} LiPi(t)Ri(t) \rightarrow \max$$

- де N – загальна кількість заявок, що надійшли в систему за інтервал часу ΔT.

12

Розбиття графа G хмарного сховища, що масштабується, на два подграфи: ***Gs*** і ***Gr***



13

Оцінювання ефективності

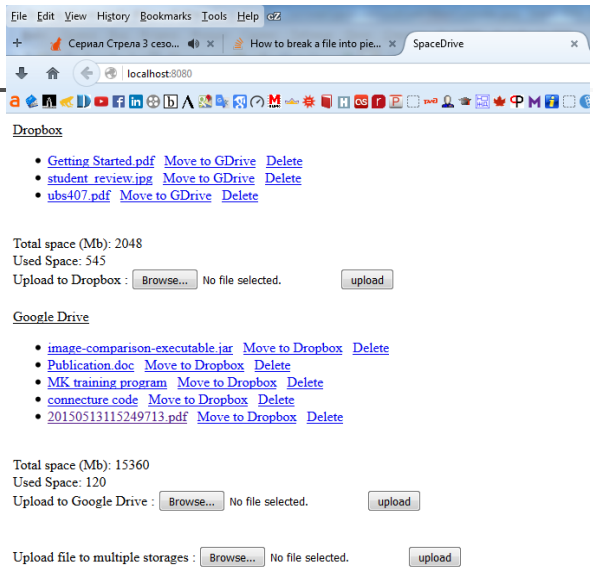
- Запропоновано два алгоритму розв'язання задачі: переборний алгоритм і апроксимаційний алгоритм поліноміальної обчислювальної складності
  max {O (| A | log Δ), O (| A | (| V | + δ))},

  де A - множина ребер; V - множина вершин;
  Δ - максимальна ступінь вершин; δ - константа.

- Алгоритми дають оптимальний результат за критерієм мінімізації часу масштабування.
- Оптимальна швидкість масштабування дозволить хмарним сховищ гранично швидко орендувати і вивільняти пристрої зберігання даних.

14

Розробка програми



15

Висновки

- Обоснована актуальність досліджень використання технологічних стеків при проетуванні веб-застосунків.
- Проаналізовано основні поняття і концепції предметної галузі – технологічних стеків
- Обґрунтовано актуальність дослідження застосування «хмарних сховищ в інтеграційній платформі.
- Описано основні поняття і концепції предметної області - інтеграція хмарних сховищ.
- Проаналізовано існуючі підходи і алгоритми інтеграції хмарних сховищ, оцінені їх недоліки та переваги.
- Запропоновано варіант платформи для хмарної інтеграції.
- Обґрунтовано вибір програмного інструментарію для розробки програмного забезпечення, в ході чого була вибрана мова Java.
- Зроблено опис об'єктної і алгоритмічної моделі системи “Платформа для інтеграції хмарних сховищ”

16

ДОДАТОК В

Апробація результатів роботи

The Methods of Developing Web-applications using Technology Stack

Radionova A., Shubin I.

The automated testing of Web-based software can drastically improve the quality and reduce the costs of software development and testing. However, it requires careful planning and significant initial investment. To make it more efficient we should analyze the habits and preferences of end-users.

Here we analyze the known approaches to this problem and offer methods to improve the users behavior reflection

Modern-day Web-based software, as any user-centric software in general, is characterized by the increasing complexity and growing demands of end-users for convenience, speed, reliability and attention to special needs. New tools, such as context-sensitive search, syndication, and tagging [1] are being invented to provide Web designers, developers, and quality specialists with relevant tools for comprehending and eliminating such challenges. Let us leave their job to designers and developers and focus on the last and consequently most important stage of the Web solution creation – testing. No doubts that any solution we produce should be of high quality. As Humble and Farley correctly mentioned, "building quality also means constantly working to improve your testing strategy". Building the proper strategy earlier and implementing it into efficient automated tests through the entire system lifetime can save us a tremendous effort and resources that could be put to activities that are more creative. We should also take into account the future beyond the release date. It is clear that any system striving to survive over next couple of years should already be "built to be rebuilt". Efficient automated test coverage and a comprehensible user behavior model could help us make these rebuilds fitting within the budgets and ensure the requested quality levels.

What should the efficient testing strategy be based on? According to [2], "the design of a testing strategy is primarily a process of identifying and prioritizing project risks and deciding actions … to mitigate them". The main quality risks for the Web applications and user experience are mistakes made by users and breaking their social habits and convenience, leading to dissatisfaction and denial of cooperation.

In this paper, we will focus on the known and prospective techniques for such habits detection and reflection in the modern automated tests implementation.

Let us look at the Web user habits and activities as an important component of general social behavior.

According to OFcom [3], today British teenagers spend more than 27 hours a week browsing the Internet, with average adult's 20 hours a week online is not far from it. It means that online activities are gradually replacing the usual offline communication and cooperation and have become a crucial part of lifestyle, heavily influencing satisfaction, mood, and even mental health of the society members. Willing to take maximum advantage of the users' good mood and whatever it takes avoiding their distraction, Web content owners and producers have no choice but to carefully study, collect and process the user behavior statistics to produce viable prediction models.

Due to the importance of the issue and huge variety of study subjects, there is a plenitude of approaches and proposed categories for analysis of the Web audience behavior. A good example of successful user behavior analysis based on a large-scale predefined survey is presented in the Alberta University report. It shows, among others, the growing importance of mobile devices and technologies in the Web user experience, significant differences in the perception of different information channels and formats by different age and professional groups (students/parents/employees). Moreover, it discovers a growing demand for personalized communications such as direct mailing from the management versus just publishing data on the Web site.

Along with the static user behavior modelling, in our test strategy it is also important to account for the changing nature of user behavior that can be reflected in the dynamic models. The following dynamic parameters are considered: varied probability of wrong user actions performed, such as mouse-clicks within a certain radius around the target object boundaries causes by the muscular tiredness consequences such as carpal tunnel syndrome (CTS); varied probability of keyboard typing mistakes caused both by tiredness and loss of focusing due to various environmental distractors; decreasing rate of visual information perception caused by excessive eyes exertion and excessive or deficient illumination, leading to the necessity to call for the same information again; mMomentary contextual environment influence, often stressful, such as driving the car in a traffic jam, or looking after the children playing in the kitchen.

During the first implementation stage the qualitative parameters operationalization methods described in [1] will be used. The exact formulas for calculating the output parameters will be improved using semi-automated learning during the pilot exploitation of the system. The pilot users will be able to provide their feedback regarding the perceived deviations of the values of output parameters. This feedback will be analyzed to detect the flaws in calculations, and the model will be updated accordingly. For the implementation of the described model into software solutions, we have chosen the Service-Oriented Architecture (SOA) approach integrated with Selenium WebDriver test automation Suite. Selenium and specialized solutions based on it are the most popular freeware Web test automation platform today, supported on all popular browsers and mobile architectures. The WebDriver protocol becomes de-facto a standard for communication between the components of the UI test automation software.

This approach will allow the maximum reuse of browser integration capabilities of Selenium, preserve the platform and language flexibility, and focus our development on injecting the user behavior-specific logic into the automated tests.

1. Shubin I., The Methods of Adaptation in Computer-Based Training Systems// Proceeding of 2015 Information Technologies in Information Business Conference (ITIB) 7 – 9 October, 2015

2. J. Humble, D. Farley, Continuous Delivery, Pearson Education, 2011.

3. Media use and attitudes report by OFcom, May 2015, Retrieved August 1st, 2015 .