МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра «стратегічне управління»

ЗВІТ

з лабораторної роботи №6

з дисципліни

«ПРОГРАМУВАННЯ ВЕБ-ЗАСТОСУВАНЬ java ee»

ВИКОНАВ

Студент групи КН-317А

ПЕРЕВІРИла

ас. каф. СУ

Луценко С. Ю.

Харків 2020

**ДОСЛІДЖЕННЯ КОМПОНЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ АУТЕНТИФІКАЦІЇ ТА СУПРОВОДЖЕННЯ СЕСІЙ**

ДОСЛІДЖЕННЯ КОМПОНЕНТІВ JAVA ДЛЯ ОРГАНІЗАЦІЇ EMAIL-РОЗСИЛКИ

Мета роботи

Закріпити навички щодо особливостей застосування классів ServletFilter, Session, Cookies. Набути навики розробки додатків для реалізації аутентифікації та супроводження сесій.

**Відправлення пошти**

Закріпити та поглибити знання з розробки застосувань Java для відправлення e-mail. Набути навички розробки додатків для e-mail-розсилки, набути навички налаштування поштового сервера.

Завдання роботи

**Завдання 1.** Додати до функціоналу веб-застосування реалізацію прецеденту «Аутентифікація». Даний функціонал має забезпечити перевірку логіну та паролю користувача на початку роботи із системою, а також контроль часу сеансу роботи користувача.

**Завдання 2.** Додати до функціоналу веб-застосування функцію відправлення даних одному або декільком адресатам на e-mail.

Даний функціонал у мінімальній реалізації має дозволити:

визначити адресата (список адресатів) для відправлення листа;

указати тему листа, що буде відправлено;

формувати для відправлення вкладення (attachments) у вигляді файла (наприклад, xml – як у ЛР №2), у який зберігти вміст будь-якої таблиці БД.

Хід виконання

1 Додати до функціоналу веб-застосування реалізацію прецеденту «Аутентифікація».

До проекту були додані такі сервлети як LoginServlet.java, LogoutServlet.java, RegisterServlet.java. Їх коди можна побачити у лістинг ах 1-3.

Лістинг 1 - LoginServlet.java

|  |
| --- |
| package Servlets;  imports …  @WebServlet("/Login")  public class LoginServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private static DbOperations dbHelper = null;    public static void setDbHelper(DbOperations DbHelper) {  dbHelper = DbHelper;  }  public LoginServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html; charset=UTF-8");  request.getRequestDispatcher("WEB-INF/Views/Account/Login.jsp").forward(request, response);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String username = request.getParameter("username");  String pwd = request.getParameter("password");  User user = null;  try{  user = dbHelper.getUsers().stream().  filter(el -> el.getEmail().equals(username) && el.getPassword().equals(pwd)).findFirst().get();  }  catch(NoSuchElementException ex) {  request.setAttribute("ErrorMessage", "Пользователь не найден");  request.getRequestDispatcher("WEB-INF/Views/Account/Login.jsp").forward(request, response);  }  request.getSession().setAttribute("CURRENT\_USER", user);  response.sendRedirect(request.getContextPath() + "/Products");  }  } |

Лістинг 2 - LogoutServlet.java

|  |
| --- |
| package Servlets;  imports …  @WebServlet("/Logout")  public class Logout extends HttpServlet {  private static final long serialVersionUID = 1L;  public Logout() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getSession().invalidate();  response.sendRedirect(request.getContextPath() + "/Products");  }  } |

Лістинг 3 - RegisterServlet.java

|  |
| --- |
| package Servlets;  imports…  @WebServlet("/Signup")  public class RegisterServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private static DbOperations dbHelper = null;  public static void setDbHelper(DbOperations DbHelper) {  dbHelper = DbHelper;  }  public RegisterServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html; charset=UTF-8");  request.getRequestDispatcher("WEB-INF/Views/Account/SignUp.jsp").forward(request, response);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String username = request.getParameter("username");  String pwd = request.getParameter("password");  String confirm\_pwd = request.getParameter("password");  User user = null;  try{  user = dbHelper.getUsers().stream().  filter(el -> el.getEmail().equals(username)).findFirst().get();  if (user != null) {  request.setAttribute("ErrorMessage", "Пользователь с таким именем уже существует");  request.getRequestDispatcher("WEB-INF/Views/Account/Signup.jsp").forward(request, response);  }  }  catch(NoSuchElementException ex) {  if (pwd.equals(confirm\_pwd) &&  pwd.matches("(?=.\*\\d)(?=.\*[a-z])(?=.\*[A-Z]).{4,}") &&  username.matches("^([a-zA-Z0-9\_\\-\\.]+)@([a-zA-Z0-9\_\\-\\.]+)\\.([a-zA-Z]{2,5})$")) {  user = new User();  user.setEmail(username);  user.setPassword(pwd);  dbHelper.addUser(user);  }  else {  request.setAttribute("ErrorMessage", "Введите корректные данные");  request.getRequestDispatcher("WEB-INF/Views/Account/Signup.jsp").forward(request, response);  }  request.getSession().setAttribute("CURRENT\_USER", user);  response.sendRedirect(request.getContextPath() + "/Products");  }  }  } |

Також був доданий веб-слухач (web-listener), що відповідає за розвиток подій коли створюється нова сесія. Функція setMaxInactiveInterval() визначає інтервал у секундах через який, при умові бездіяльності користувача, слід здійснити вихід для нього. У цьому випадку користувач автоматично виходить, якщо протягом 15 хвилин він не виконує ніякі запити. Код представлено у лістингу 4.

Лістинг 4 – MyHttpSessionListener.java

|  |
| --- |
| package Listeners;  import javax.servlet.annotation.WebListener;  import javax.servlet.http.HttpSessionEvent;  import javax.servlet.http.HttpSessionListener;  @WebListener  public class MyHttpSessionListener implements HttpSessionListener {  public void sessionCreated(HttpSessionEvent event){  event.getSession().setMaxInactiveInterval(15\*60); // in seconds  }  } |

Був доданий фільтр, що відстежує параметр CURRENT\_USER сесії і якщо він не null (тобто користувач увійшов до системи) – дозволяє перегляд усіх сторінок. Якщо користувач не у системі, то при переході на різні сторінки сайту здійснюється його пере направлення на сторінку з входом до сайту. Лістинг фільтру наведено нижче.

Лістинг 5 – Auth.java

|  |
| --- |
| package Filters;  imports…  @WebFilter(urlPatterns = {"/Elements", "/Products", "/CreateUpdateElement",  "/CreateUpdateProduct", "/DeleteElement", "/DeleteProduct", "/SendMail",  "/CreateUpdateGroup", "/Groups", "/DeleteGroup"} )  public class Auth implements Filter {  public Auth() {  }  public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws IOException, ServletException {  resp.setContentType("text/html; charset=UTF-8");  PrintWriter out=resp.getWriter();  HttpServletRequest httpReq = (HttpServletRequest) req;  User user = (User) httpReq.getSession().getAttribute("CURRENT\_USER");  if(user != null){  chain.doFilter(req, resp);//sends request to next resource  }  else{  HttpServletResponse httpResp = (HttpServletResponse) resp;  httpResp.sendRedirect("Login");  }  }  public void init(FilterConfig fConfig) throws ServletException {  }  } |

Сторінки з входом та реєстрацією – рисунок 1-2.

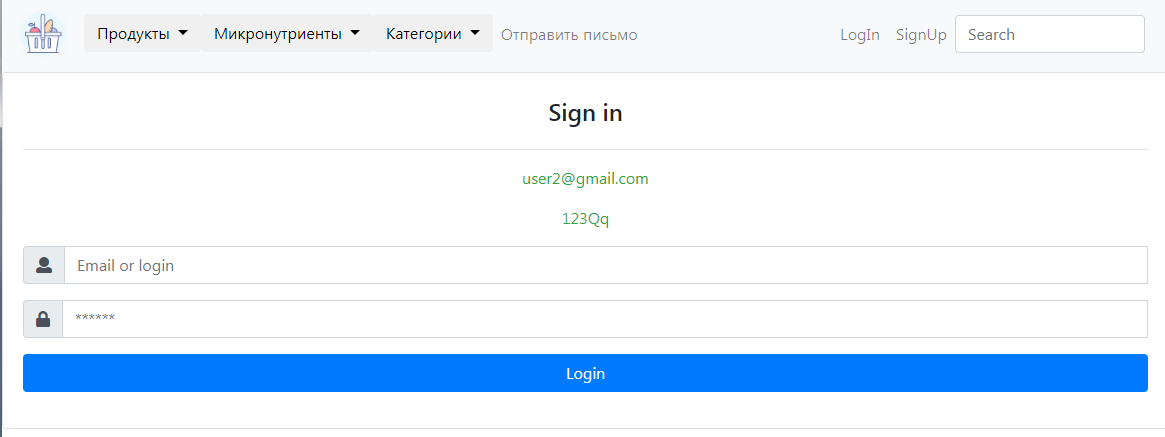


Рисунок 1 – Сторінка входу

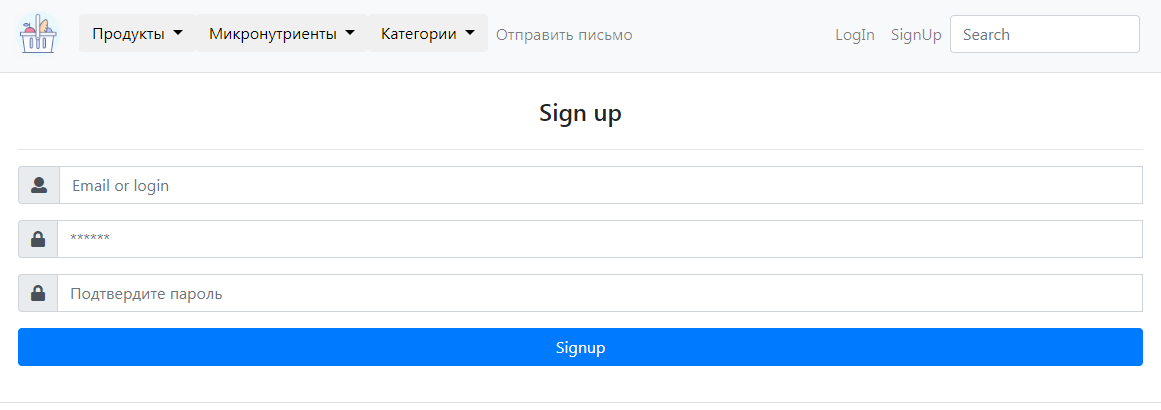


Рисунок 2 – Сторінка реєстрації

На стороні клієнта здійснюється валідація за допомогою bootstrap класів та jquery скрипту. Приклад проходження валідації – рисунок 3.

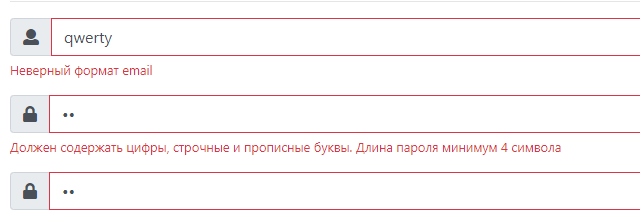


Рисунок 3 – Процес валідації

Для зберігання даних про користувачів до проекту була додана інша база даних, а саме SQLite. Для цього було визначено залежність sqlite-jdbc у файлі з налаштуваннями pom.xml. До BuildPath був доданий пакет з діалектом hibernate призначеним для роботи з SQLite. Цей діалект може бути знайдений за посиланням <https://github.com/gwenn/sqlite-dialect>.

До класу ресурсів був доданий новий конфігураційний файл hibernate – sqlite.cfg.xml. У класі HibernateHelper був налаштований конструктор, у якому створюються два об’єкта типу SessionFactory для MySql та SQLite відповідно. Надалі методи що працюють з даними користувачів використовуватимуть SessionFactory що повязаний з SQLite і дані користувачів зберігатимуться саме у базі SQLite.

Файл sqlite.cfg.xml наведено у лістингу 6.

Файл HibernateHelper.java наведено у лістингу 7.

Лістинг 6 - sqlite.cfg.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 5.3//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-5.3.dtd">  <hibernate-configuration>  <session-factory>  <property name=*"show\_sql"*>true</property>  <property name=*"format\_sql"*>true</property>  <property name=*"dialect"*>org.sqlite.hibernate.dialect.SQLiteDialect</property>  <property name=*"connection.driver\_class"*>org.sqlite.JDBC</property>  <property name=*"connection.url"*>jdbc:sqlite:javaee.db</property>  <property name=*"connection.username"*>""</property>  <property name=*"connection.password"*>""</property>  <property name=*"hibernate.hbm2ddl.auto"*>update</property>  <mapping class=*"models.User"* />  </session-factory>  </hibernate-configuration> |

Лістинг 7 - HibernateHelper.java

|  |
| --- |
| **imports…**  **public** **class** HibernateHelper **implements** DbOperations{  **private** SessionFactory sf1 = **null**;  **private** SessionFactory sf2 = **null**;    **public** HibernateHelper() {  Configuration cfg1 = **new** Configuration();  cfg1.configure("mysql.cfg.xml");  **this**.sf1 = cfg1.buildSessionFactory();  Configuration cfg2 = **new** Configuration();  cfg2.configure("sqllite.cfg.xml");  **this**.sf2 = cfg2.buildSessionFactory();  }  persistent methods…  } |

2 Додати до функціоналу веб-застосування функцію відправлення даних одному або декільком адресатам на e-mail.

Для реалізації можливості відсилати дані таблиць поштою було зроблено декілька кроків. У файлі налаштувань pom.xml булла додана нова залежність – javax.mail. Був створений сервлет SendMailServlet.java, та SendMail.jsp сторінка з формою вибору отримувачів та посилаємих даних. У сервлеті SendMailServlet був використаний PRG паттерн для усунення можливості повторної відправки даних. Також був створений службовий клас Mailer.java із статичним методом відправки повідомлень.

Код SendMailServlet.java наведено у лістингу 8.

Код SendMail.jsp наведено у лістингу 9.

Код Mailer.java наведено у лістингу 10.

Вигляд форми з вибором отримувачів та даних – рисунок 4.

Лістинг 8 - SendMailServlet.java

|  |
| --- |
| package Servlets;  imports…  @WebServlet("/SendMail")  public class SendMailServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  private static DbOperations dbHelper = null;  private Product product = null;  private Element element = null;  private Group group = null;  private List<Product> products = null;  private List<Element> elements = null;  private List<Group> groups = null;  private List<byte[]> xmls = new ArrayList<>();    public static void setDbHelper(DbOperations DbHelper) {  dbHelper = DbHelper;  }    public SendMailServlet() {  super();  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html; charset=UTF-8");  response.setCharacterEncoding("UTF-8");  List<Object> objects = new ArrayList<>(Arrays.asList(  dbHelper.getElements().stream().map(el -> el.toString() + " - Микронутриент").collect(Collectors.toList()),  dbHelper.getProducts().stream().map(el -> el.toString() + " - Продукт").collect(Collectors.toList()),  dbHelper.getGroups().stream().map(el -> el.toString() + " - Категория").collect(Collectors.toList())));  request.setAttribute("objects", objects);  request.getRequestDispatcher("WEB-INF/Views/Mail/SendMail.jsp").forward(request, response);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  xmls.clear();  String subject = request.getParameter("subject");  String[] objects = request.getParameterValues("objects");  String text = request.getParameter("text");  String[] usernames = request.getParameterValues("username");  if (usernames != null && objects != null) {  for (int j = 0; j < objects.length; j++) { // Loop through each object selected by user  String object = objects[j].split(" - ")[0];  String type = objects[j].split(" - ")[1];  try {  switch(type) {  case "Продукт":  product = dbHelper.getProducts().stream().filter(el -> el.getName().equals(object)).findFirst().get();  xmls.add(XMLHelper.writeObject(product));  break;  case "Микронутриент":  element = dbHelper.getElements().stream().filter(el -> el.getName().equals(object)).findFirst().get();  xmls.add(XMLHelper.writeObject(element));  break;  case "Категория":  group = dbHelper.getGroups().stream().filter(el -> el.getName().equals(object)).findFirst().get();  xmls.add(XMLHelper.writeObject(group));  break;  case "Продукты":  products = dbHelper.getProducts();  xmls.add(XMLHelper.writeObject(products));  break;  case "Микронутриенты":  elements = dbHelper.getElements();  xmls.add(XMLHelper.writeObject(elements));  break;  case "Категории":  groups = dbHelper.getGroups();  xmls.add(XMLHelper.writeObject(groups));  break;  default:  break;  }  }  catch(EntityNotFoundException ex) {  ex.printStackTrace();  };  }  try {  String username = "";  Address[] addresses = new Address[usernames.length];  for (int i = 0; i < usernames.length; i++) {  addresses[i] = new InternetAddress(usernames[i]);  }  if (request.getSession().getAttribute("CURRENT\_USER") != null) {  username = ((User) request.getSession().getAttribute("CURRENT\_USER")).getEmail();  }  Mailer.sendMail(addresses, text, subject, xmls, objects, username);  } catch (MessagingException e) {  e.printStackTrace();  }  }  response.sendRedirect(request.getContextPath() + "/SendMail");  }  } |

Лістинг 9 - SendMail.jsp

|  |
| --- |
| <%@taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>  <%@ include file = *"/WEB-INF/Views/Shared/Header.jsp"* %>  <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <c:if test=*"*${ ErrorMessage != null }*"*>  <div class = *"alert alert-danger"*>  ${ ErrorMessage }  </div>  </c:if>  <div class=*"card"*>  <article class=*"card-body"*>  <h4 class=*"card-title text-center mb-4 mt-1"*>Отправить письмо</h4>  <hr>  <p class=*"text-success text-center"*>Почта будет отправлена с адреса microsmacrosjavaee@gmail.com</p>  <button class = *"btn btn-info"* onclick = "addrecipient()">Добавить получателя</button>  <form id = *"form"* class= *"needs-validation"* novalidate method = *"post"* action = *"SendMail"*>  <div class=*"form-group"* id = *"recipients"*>  <div class=*"input-group"*>  <div class=*"input-group-prepend"*>  <span class=*"input-group-text"*> <i class=*"fa fa-user"*></i> </span>  </div>  <input name=*"username"* class=*"form-control"* placeholder=*"Почта получателя"* type=*"email"* required>  </div>  </div>  <div class=*"invalid-feedback"*>  Неверный формат email  </div>    <div class=*"form-group"*>  <div class=*"input-group"*>  <div class=*"input-group-prepend"*>  <span class=*"input-group-text"*> <i class=*"fa fa-envelope-open"*></i> </span>  </div>  <input  name = *"subject"*  class=*"form-control"* placeholder=*"Тема сообщения"* type=*"text"* required>  <div class=*"invalid-feedback"*>  Укажите тему сообщения  </div>  </div>  </div>  <div class=*"form-group"*>  <label for=*"multipleSelect"*>Выберите иформацию о каком объекте нужно отправить</label>  <select name = *"objects"* multiple class=*"form-control"* id=*"multipleSelect"* required>  <c:forEach items = *"*${ objects }*"* var = *"list"*>  <c:forEach items = *"*${ list }*"* var = *"object"*>  <option>${ object }</option>  </c:forEach>  </c:forEach>  <option>Все обьекты - Продукты</option>  <option>Все обьекты - Микронутриенты</option>  <option>Все обьекты - Категории</option>  </select>  <div class=*"invalid-feedback"*>  Выберите по крайней мере один объект  </div>  </div>  <div class=*"form-group"*>  <label for=*"textArea"*>Добавить текст (Опционально)</label>  <textarea name = *"text"* class=*"form-control"* id=*"textArea"* rows=*"3"*></textarea>  </div>  <div class=*"form-group"*>  <button type=*"submit"* class=*"btn btn-primary btn-block"*>Отправить</button>  </div>  </form>  </article>  </div>  <script>  **function** addrecipient(){  $('<div>\  <div **class**="input-group-prepend">\  <span **class**="input-group-text"> <i **class**="fa fa-user"></i> </span>\  </div>\  <input name="username" **class**="form-control" placeholder="Почта получателя" type="email" required>\  </div>').addClass('input-group').appendTo('#recipients')  }  (**function**() {  'use strict';  window.addEventListener('load', **function**() {  **var** forms = document.getElementsByClassName('needs-validation');  **var** validation = Array.prototype.filter.call(forms, **function**(form) {  form.addEventListener('submit', **function**(event) {  **if** (form.checkValidity() === **false**) {  event.preventDefault();  event.stopPropagation();  }  form.classList.add('was-validated');  }, **false**);  });  }, **false**)})()    </script>  <%@ include file = *"/WEB-INF/Views/Shared/Footer.jsp"* %> |

Лістинг 10 – Mailer.java

|  |
| --- |
| package models;  imports…  public class Mailer {  private static String host = "smtp.gmail.com";  private static String username = "macrosmicrosjava";  private static String password = "\*\*\*\*\*";  private static Properties props = new Properties();  private static MimeBodyPart attachment = null;  private static Multipart multipart = null;  private static ByteArrayDataSource ds = null;  static {  props.setProperty("mail.smtp.ssl.enable", "true");  props.put("mail.smtp.ssl.enable", true);  props.put("mail.smtp.host", host);  }  private static Session session = Session.getInstance(props);  private static Message msg = new MimeMessage(session);  private static MimeBodyPart messageBodyPart = null;    public static void sendMail(Address[] recipients, String message, String subject, List<byte[]> xmls, String[] objects, String email) throws AddressException, MessagingException {  multipart = new MimeMultipart();  msg.setRecipients(Message.RecipientType.TO, recipients);  msg.setText(message);  msg.setSubject(subject);  msg.setSentDate(new Date());  messageBodyPart = new MimeBodyPart();  messageBodyPart.setContent(message + " | Отправлено пользователем " + email, "text/plain; charset=UTF-8");  multipart.addBodyPart(messageBodyPart);  for (int i = 0; i < xmls.size(); i++) {  attachment = new MimeBodyPart();  ds = new ByteArrayDataSource(xmls.get(i), "application/xml");  attachment.setDataHandler(new DataHandler(ds));  attachment.setFileName(objects[i] + ".xml");  multipart.addBodyPart(attachment);  }  msg.setContent(multipart);  Transport.send(msg, username, password);  }  } |

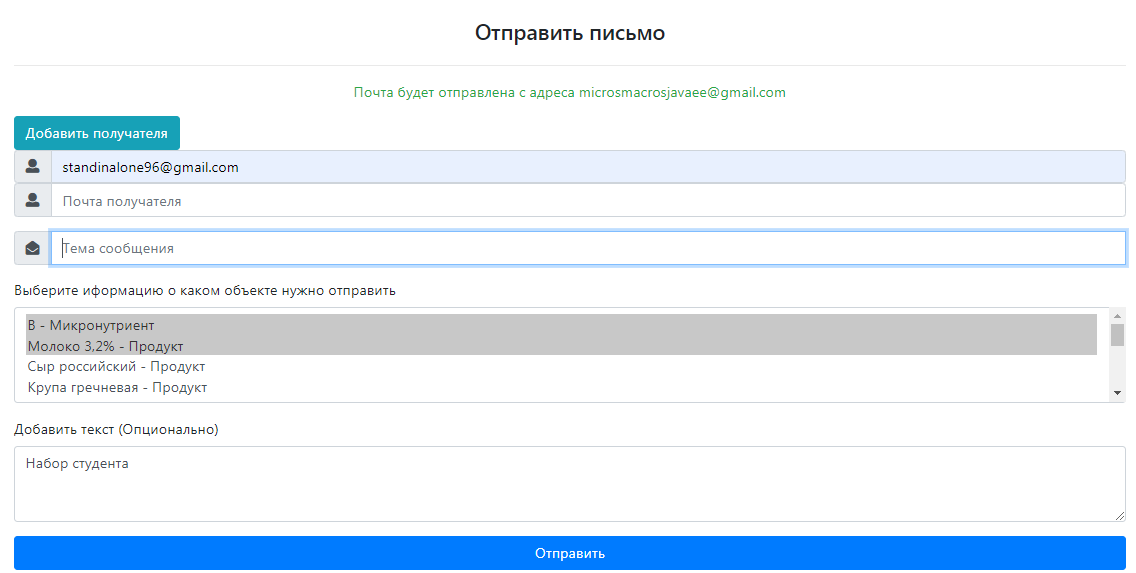


Рисунок 4 – Вибір одержувачів та необхідних даних

Коли сервлет отримує список обраних даних – він викликає функцію writeObject() нового службового класу XMLHelper, який створює масив байтів для кожного об’єкту та відправляє ці масиви як вкладення до листа – рисунок 5.

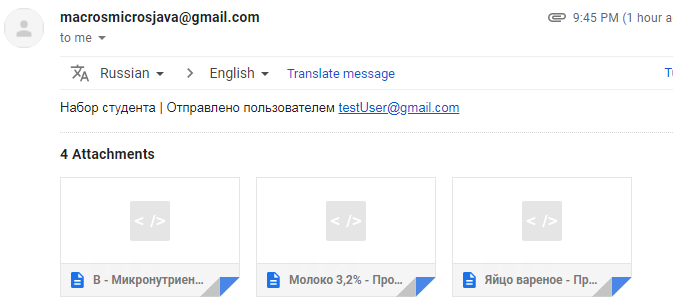


Рисунок 5 – Лист з вкладеннями для кожного обраного обєкту

Вміст вкладень – звичаний xml файл з інформацією про об’єкт. Зразок такого файлу наведено у лістингу 11.

Лістинг 11 - B - Микронутриент.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <java version="1.8.0\_221" class="java.beans.XMLDecoder">  <object class="models.Element" id="Element0">  <void class="models.Element" method="getField">  <string>Name</string>  <void method="set">  <object idref="Element0"/>  <string>B</string>  </void>  </void>  <void class="models.Element" method="getField">  <string>id</string>  <void method="set">  <object idref="Element0"/>  <int>45</int>  </void>  </void>  </object>  <void class="org.hibernate.collection.internal.PersistentBag">  <void property="owner">  <object idref="Element0"/>  </void>  </void>  </java> |

Висновки

При виконанні цієї лабораторної роботи була здійснена робота з smtp протоколом з застосуванням JavaMail API що дозволив вирористовувати акаунт Google для відправки листів. Був доданий фільтр авторизованих користувачів – ще один крок до більш безпечного користування застосунком.