Sporty Shoes

- Pedro Leal | PG FSD Implement Frameworks the DevOps way | 22/04/20223

Index

Content

Overview

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their ecommerce portal sportyshoes.com.

The admin is able:

* Manage the products in the store including categorizing them

* Browse the list of users who have signed up and be able to search users

* See purchase reports filtered by date and category

* Change password

Producty Backlog

Sprint no	Task planed	Start date	End date	status	Details
1	Create Product Page	11-4	11-4	completed	
	Create User Login	11-4	11-4	completed	
	Differentiate access levels between administrator and customers	4-4	4-4	completed	I create DAO for all objects using hibernate, the DAO include create, update, retrieve, list methods.
	New customers can register in the web site	11-4	11-4	completed	
	Admin can view, edit and create products	11-4	11-4	completed	
	Logged in users can manage brands	11-4	11-4	completed	
	Customers can buy products	16-4	16-4	completed	
	Browse List of signed up users	17-4	17-4	completed	
2	Administrators can edit products	17-4	17-4	completed	
	Administrator can manage brands	19-4	19-4	completed	
	Administrator can change password	22-4	22-4	completed	

Database Diagram

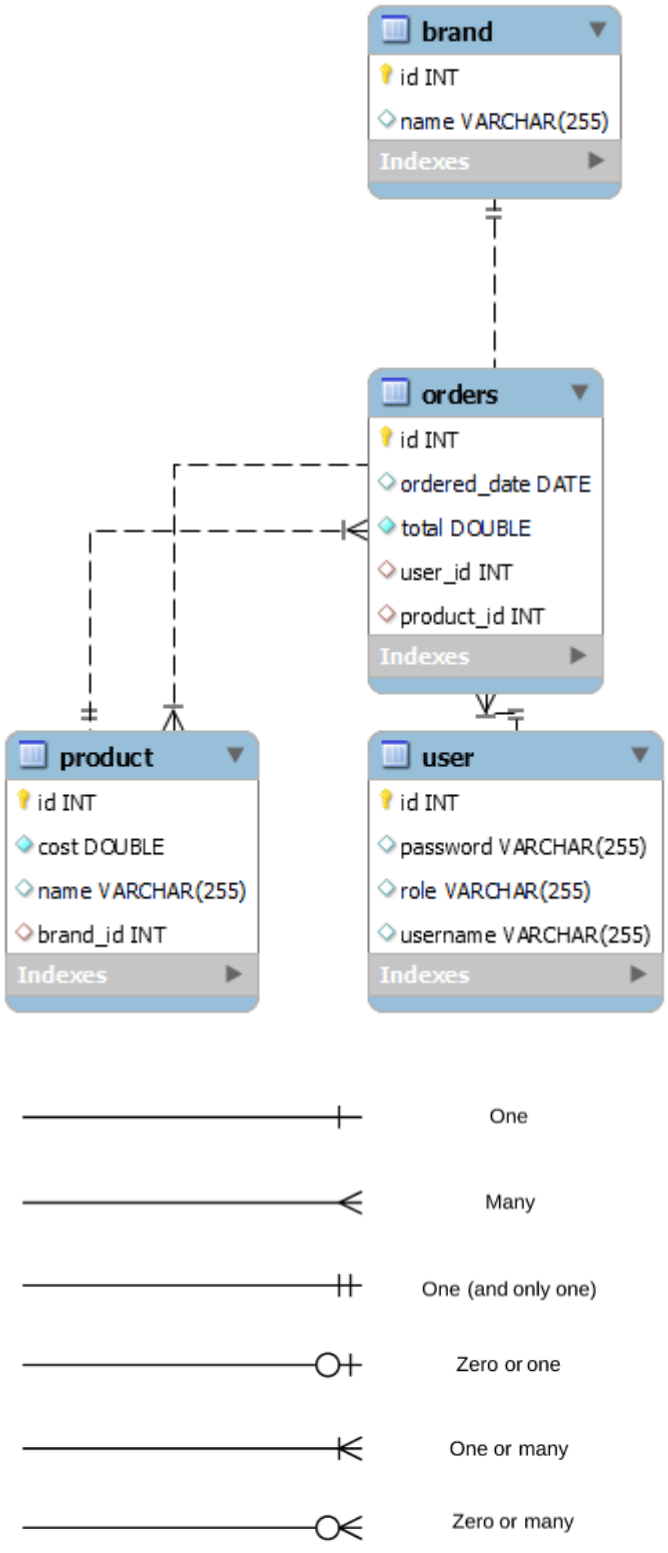


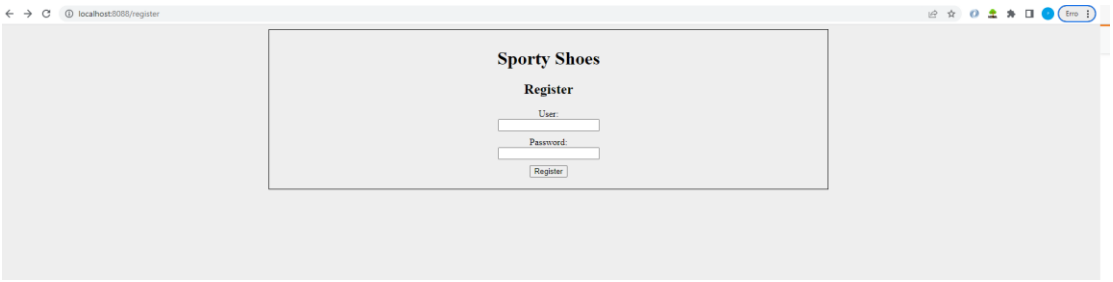
Figure 1 Database Diagram Legend

Key Features

USER AUTHENTICATION

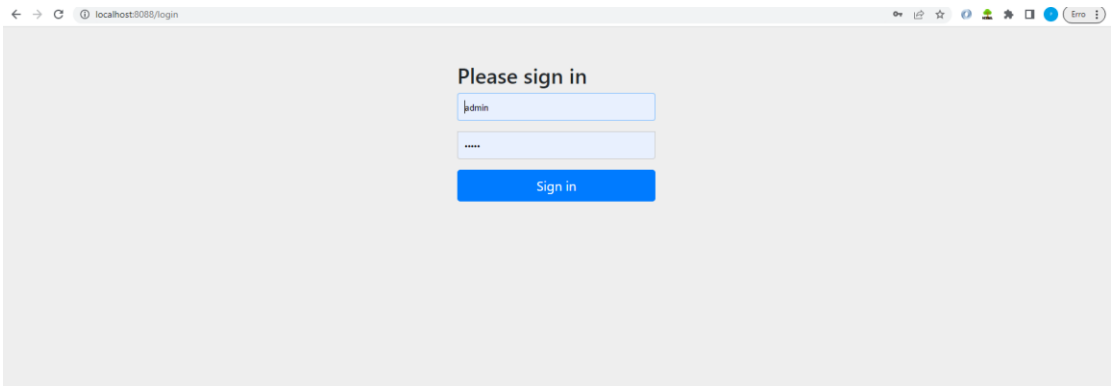
- This is the first page of web site which can anyone access.

- New customers can register

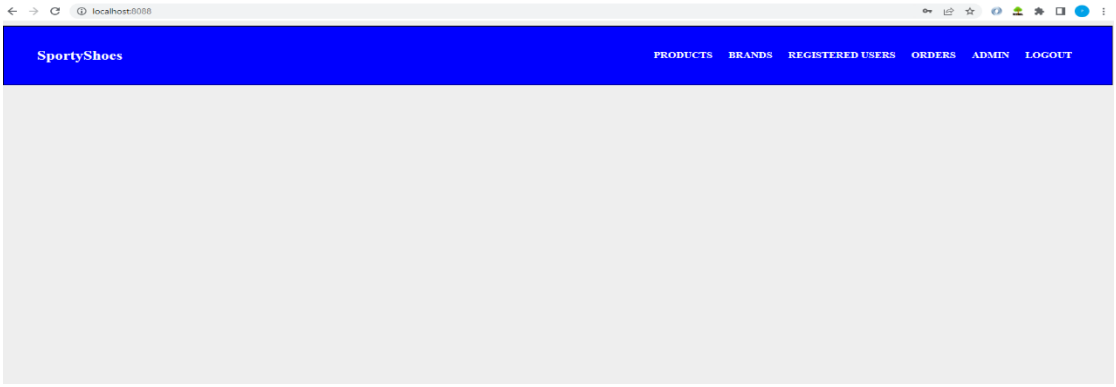


```java
@PostMapping("/register")
public ModelAndView registerUser(HttpServletRequest request, HttpServletResponse response) {
    ModelAndView mv = new ModelAndView("redirect:/login");

    User user = new User();
    user.setUsername(request.getParameter("user"));
    bCryptPasswordEncoder.encode(request.getParameter("pwd"));
    user.setPassword(    bCryptPasswordEncoder.encode
            (request.getParameter("pwd")));


    String password = user.getPassword();
    user.setPassword(password);
    user.setRole("CUSTOMER");



    userRepository.save(user);

    return mv;

}
```

- Administrator and already registered customers can login in the portal



- After the Login, if it's Administrator he can access products, signed-up users, and view orders

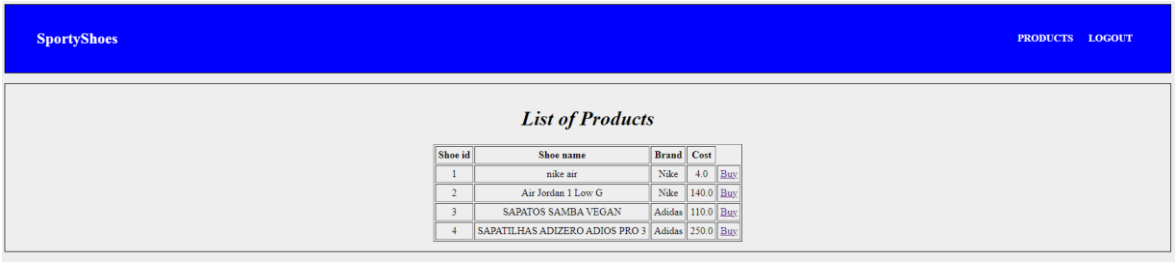


- The customers can access the products page, and buy products


```
@Override
     protected void configure(HttpSecurity http) throws Ex-
ception {
          // TODO Auto-generated method stub
          http.csrf().disable().authorizeRequests()
          .antMatchers("/product/index").hasAnyAutho-
rity("CUSTOMER","ADMIN")
          .antMatchers("/product/addProduct").hasAutho-
rity("ADMIN")
          .antMatchers("/product/delete").hasAuthority("AD-
MIN")
          .antMatchers("/product/edit").hasAuthority("AD-
MIN")

          .antMatchers("/order/index").hasAnyAuthority("AD-
MIN")
          .antMatchers("/order/search").hasAnyAutho-
rity("ADMIN")
          .antMatchers("/order/buy").hasAnyAuthority("CUS-
TOMER")
          .antMatchers("/user/index").hasAnyAuthority("AD-
MIN")
          .antMatchers("/user/profile").hasAnyAutho-
rity("CUSTOMER","ADMIN")
          .antMatchers("/user/changePassword").hasAnyAutho-
rity("CUSTOMER","ADMIN")
          .antMatchers("/brand/**").hasAnyAuthority("AD-
MIN")
          .and().formLogin();

     }
```

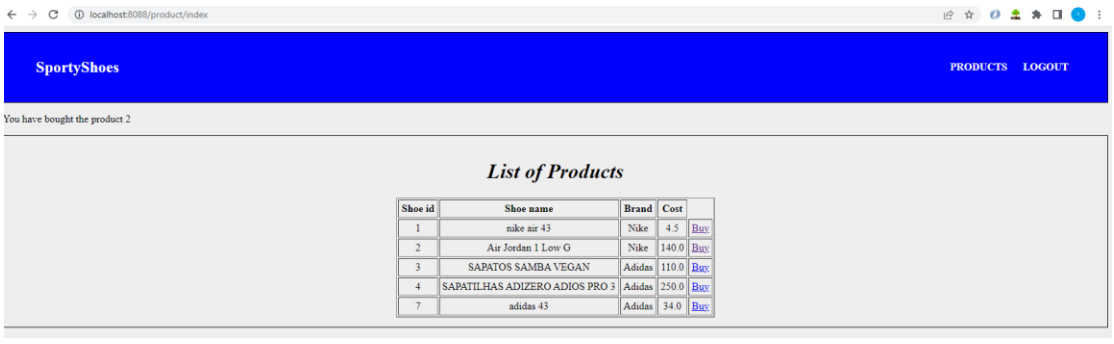
PRODUCTS LISTINGS

- After authentication customers and admin can view listing of products
- If the authenticated user is customer, he can buy product



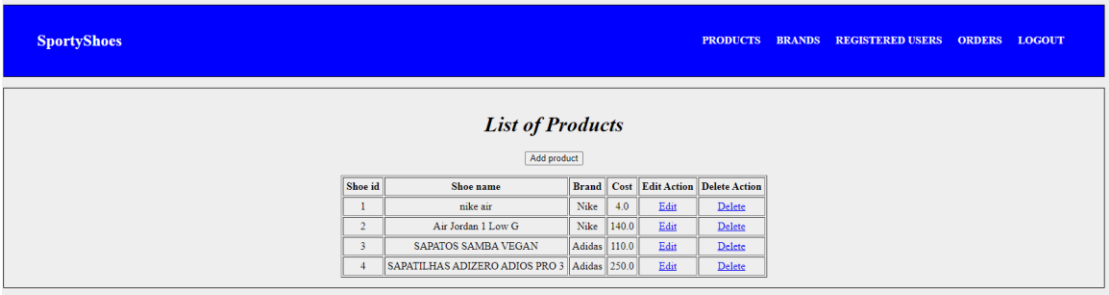
```
<jsp:include page="header.jsp" />

<%@page import="com.caltech.sportyshoes.pojo.*"%>
<%@ taglib
uri="http://www.springframework.org/security/tags"
     prefix="sec"%>
<%@page import="java.util.*"%>

<div class="container">
     <h1>
          <i>List of Products</i>
     </h1>
     <sec:authorize access="hasAuthority('ADMIN')">
          <form action="addProduct">
               <input type="submit" value="Add product">
          </form>
     </sec:authorize>

     <table border=1 cellspacing=2 cellpadding=4>
          <tr>
               <th>Shoe id</th>
               <th>Shoe name</th>
               <th>Brand</th>
               <th>Cost</th>
               <sec:authorize
access="hasAuthority('ADMIN')">
                    <th>Edit Action</th>
                    <th>Delete Action</th>
               </sec:authorize>
          </tr>
          <%
          List<Product> list = (List<Product>)
request.getAttribute("list");
          for (Product product : list) {
          %>
          <tr>
               <td><%=product.getId()%></td>
               <td><%=product.getName()%></td>
               <td><%=product.getBrand().getName()%></td>
               <td><%=product.getCost()%></td>
               <sec:authorize
access="hasAuthority('ADMIN')">
                    <td><a
href="/product/edit?id=<%=product.getId()%>">Edit</a></td>
                    <td><a
href="/product/delete?id=<%=product.getId()%>">Delete</a></
td>
               </sec:authorize>
               <sec:authorize
access="hasAuthority('CUSTOMER')">
```

- Here a customer bought product with id 2



```
@Controller
@RequestMapping("/order")
public class OrderController {

	…

	@RequestMapping("/buy")

	public ModelAndView buy(HttpServletRequest request,
HttpServletResponse response, Authentication
authentication,
			RedirectAttributes attributes) {
		ModelAndView mv = new
ModelAndView("redirect:/product/index");

		int productId =
Integer.parseInt(request.getParameter("id"));
		MyUsersDetails details = (MyUsersDetails)
authentication.getPrincipal();

		Product product =
productDao.findById(productId).orElseThrow();
		Order order = new Order();
		order.setProduct(product);
		order.setTotal(product.getCost());


	order.setCustomer(userDao.findById(details.getId()).or
ElseThrow());
		order.setOrderedDate(new Date());
		orderDao.insert(order);
		attributes.addFlashAttribute("message", "You have
bought the product " + productId);

		return mv;

	}
	…

}
```

- If it is admin he can add, edit and delete product



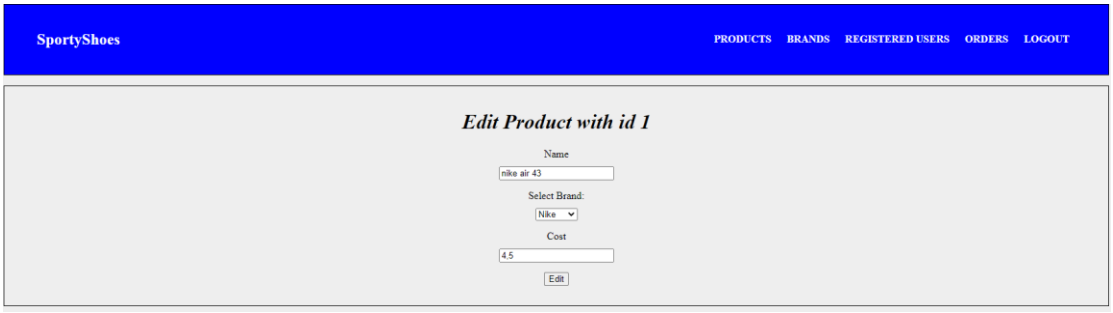
```
<jsp:include page="header.jsp" />

<%@page import="com.caltech.sportyshoes.pojo.*"%>
<%@ taglib
uri="http://www.springframework.org/security/tags"
     prefix="sec"%>
<%@page import="java.util.*"%>

<div class="container">
     <h1>
          <i>List of Products</i>
     </h1>
     <sec:authorize access="hasAuthority('ADMIN')">
          <form action="addProduct">
               <input type="submit" value="Add product">
          </form>
     </sec:authorize>

     <table border=1 cellspacing=2 cellpadding=4>
          <tr>
               <th>Shoe id</th>
               <th>Shoe name</th>
               <th>Brand</th>
               <th>Cost</th>
               <sec:authorize
access="hasAuthority('ADMIN')">
                    <th>Edit Action</th>
                    <th>Delete Action</th>
               </sec:authorize>
          </tr>
          <%
          List<Product> list = (List<Product>)
request.getAttribute("list");
          for (Product product : list) {
          %>
          <tr>
               <td><%=product.getId()%></td>
               <td><%=product.getName()%></td>
               <td><%=product.getBrand().getName()%></td>
               <td><%=product.getCost()%></td>
               <sec:authorize
access="hasAuthority('ADMIN')">
                    <td><a
href="edit.jsp?id=<%=product.getId()%>">Edit</a></td>
                    <td><a
href="delete.jsp?id=<%=product.getId()%>">Delete</a></td>
               </sec:authorize>
               <sec:authorize
access="hasAuthority('CUSTOMER')">
                    <td><a
href="/order/buy?id=<%=product.getId()%>">Buy</a></td>
```

PRODUCT INSERTION


```
@RequestMapping("/addProduct")
     public ModelAndView showAddProduct(HttpServletRequest
request, HttpServletResponse response) {
          ModelAndView mv = new ModelAndView();

          Product product = new Product();
          List<Brand> brands= brandDao.getAll();
          mv.addObject("brands", brands);
          mv.setViewName("/addProduct.jsp");
          return mv;
     }

     @PostMapping("/addProduct")
     public ModelAndView addProduct(HttpServletRequest
request, HttpServletResponse response) {
          ModelAndView mv = new
ModelAndView("redirect:/product/index");

          Product product = new Product();
          product.setName(request.getParameter("name"));
          product.setCost(Double.parseDouble(
request.getParameter("cost")));
          int brandId = Integer.parseInt(
request.getParameter("brand"));
          Brand brand =
brandDao.getByid(brandId).orElse(null);

          product.setBrand(brand);

          dao.insert(product);



          return mv;

     }
```

PRODUCT EDIT

- Edit products page



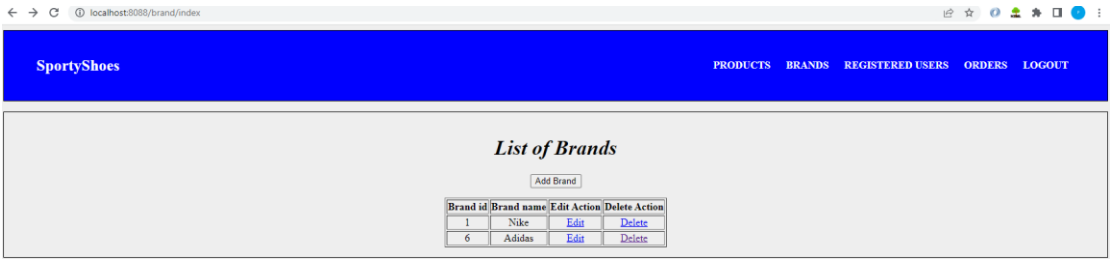

```
     @RequestMapping("/edit")
     public ModelAndView update(HttpServletRequest request,
HttpServletResponse response,
                @RequestParam("id") String id) {
          ModelAndView mv = new ModelAndView();
          Optional<Product> product =
dao.findById(Integer.parseInt(id));
          List<Brand> brands = brandDao.getAll();
          mv.addObject("brands", brands);
          mv.addObject("product", product.orElseThrow());
          mv.setViewName("/editProduct.jsp");
          return mv;
     }
     @PostMapping("/edit")
     public ModelAndView edit(HttpServletRequest request,
HttpServletResponse response,
                RedirectAttributes attributes,
               @RequestParam(value="id", required=true)
String id,
               @RequestParam(value="name", required=true)
String name,
               @RequestParam(value="brand", required=true)
String brand,
               @RequestParam(value="cost", required=true)
String cost) {

          ModelAndView mv = new
ModelAndView("redirect:/product/index");

          Product product = new Product();
          product.setId(NumberUtils.toInt(id));
          product.setName(name);
          product.setCost(NumberUtils.toDouble(cost));


     product.setBrand(brandDao.getByid(NumberUtils.toInt(br
and)).orElseThrow());
          dao.update(product);

          attributes.addFlashAttribute("message","You
succefully updated product with id " + product.getId());


          return mv;

     }
```

```
package com.caltech.sportyshoes.controllers;

import java.util.List;
import java.util.Optional;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang3.math.NumberUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.caltech.sportyshoes.dao.BrandDAO;
import com.caltech.sportyshoes.dao.ProductDAO;
import com.caltech.sportyshoes.pojo.Brand;
import com.caltech.sportyshoes.pojo.Product;



@Controller
@RequestMapping("/product")
public class ProductController {
	@Autowired
	ProductDAO dao;
	@Autowired
```

```
package com.caltech.sportyshoes.controllers;

import java.util.List;
import java.util.Optional;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang3.math.NumberUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.caltech.sportyshoes.dao.BrandDAO;
import com.caltech.sportyshoes.dao.ProductDAO;
import com.caltech.sportyshoes.pojo.Brand;
import com.caltech.sportyshoes.pojo.Product;



@Controller
@RequestMapping("/product")
public class ProductController {
	@Autowired
	ProductDAO dao;
	@Autowired
```

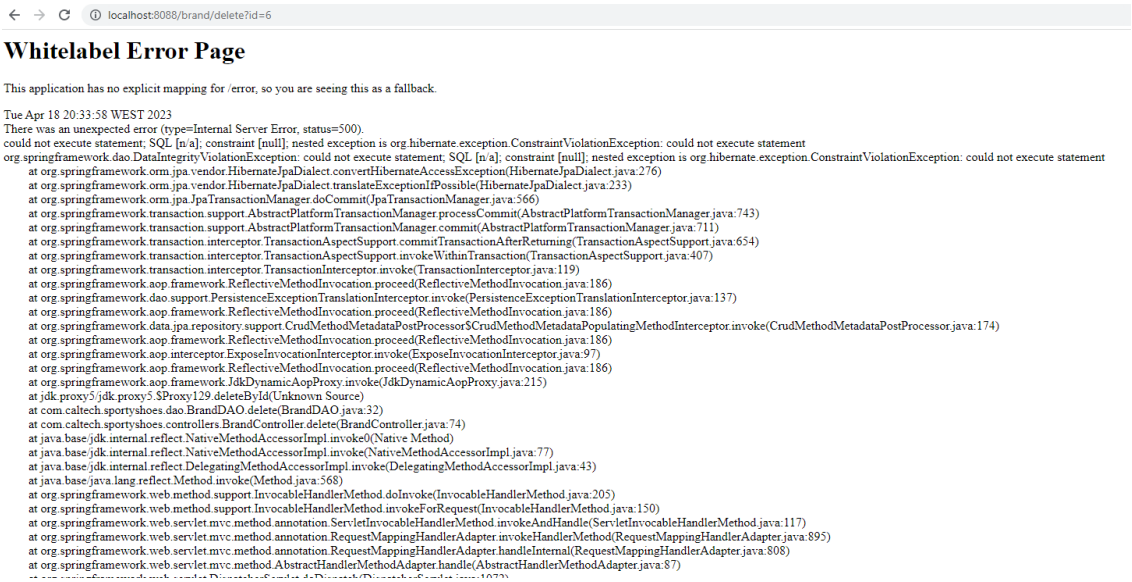
BRANDS

- The admin can view, edit, and delete brands




```
@RequestMapping("/index")
	public ModelAndView display(HttpServletRequest
request, HttpServletResponse response,
			@ModelAttribute("message") String message
			) {
		ModelAndView mv = new ModelAndView();
//
		Product product = new Product();
		List<Brand> brand = brandDao.getAll();
		mv.addObject("list", brand);
		//if(StringUtils.isNotBlank(message))
		mv.addObject(message);
		mv.setViewName("/brands.jsp");
		return mv;
	}
```

- If the admin tries to delete a brand with products associated, he will get an error

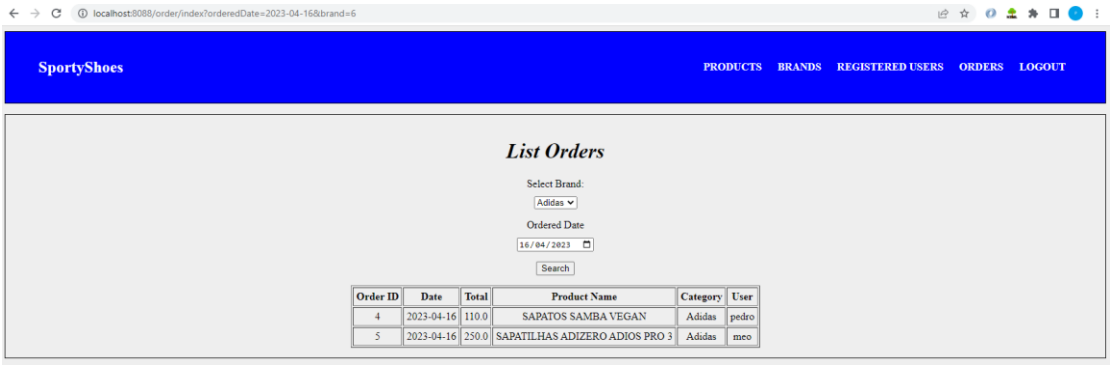


- Edit Brand page

```
@RequestMapping("/edit")
     public ModelAndView update(HttpServletRequest request,
HttpServletResponse response,
                @RequestParam("id") String id) {
          ModelAndView mv = new ModelAndView();
          Optional<Brand> brand =
brandDao.getByid(Integer.parseInt(id));

          mv.addObject("brand", brand.orElseThrow());
          mv.setViewName("/editBrand.jsp");
          return mv;
     }
     @PostMapping("/edit")
     public ModelAndView edit(HttpServletRequest request,
HttpServletResponse response,
                RedirectAttributes attributes,
                @RequestParam(value="id", required=true)
String id,
                @RequestParam(value="name", required=true)
String name
                ) {

          ModelAndView mv = new
ModelAndView("redirect:/brand/index");

          Brand brand = new Brand();
          brand.setId(NumberUtils.toInt(id));
          brand.setName(name);



          brandDao.update(brand);

          attributes.addFlashAttribute("message","You
succefully updated product with id " + brand.getId());


          return mv;

     }
```

ORDERS

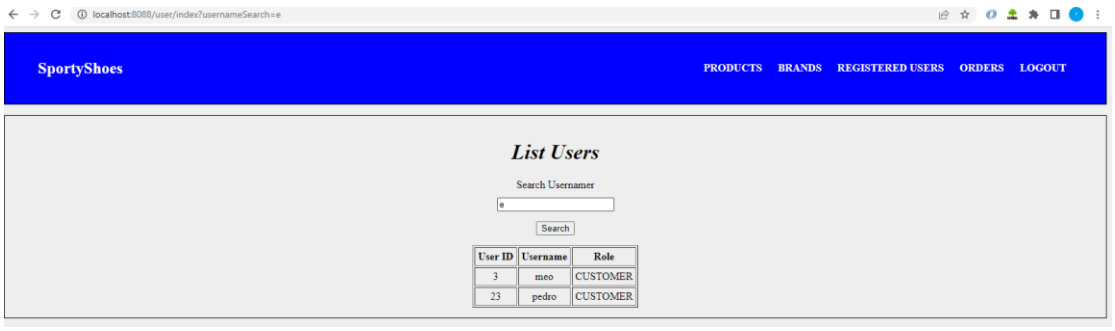
- Th admin can filter purchased orders by date and category of product,
- The fields shown are product name, total of order, customer name and ordered date



```
@PostMapping("/search")
     public ModelAndView search(HttpServletRequest request,
HttpServletResponse response,
                RedirectAttributes attributes) throws
ParseException {
          ModelAndView mv = new
ModelAndView("redirect:/order/index");

          Date orderedDate = new SimpleDateFormat("yyyy-MM-
dd").

     parse(request.getParameter("orderedDate"));
          String brand = request.getParameter("brand");


          attributes.addAttribute("orderedDate",
request.getParameter("orderedDate"));
          attributes.addAttribute("brand",brand);

          return mv;
     }

     @RequestMapping("/index")

     public ModelAndView index(HttpServletRequest request,
HttpServletResponse response, Authentication
authentication,
                @ModelAttribute("orderedDate") String
orderedDate,
                @ModelAttribute("brand") String brand
                ) throws ParseException {
          ModelAndView mv = new ModelAndView();
          //Object attribute = te("teste1");

          //String parameter =
request.getParameter("orderedDate");


          List<Brand> brands = brandDao.getAll();

           int brandID = NumberUtils.toInt(brand);
           brandID = brandID == 0 ? (brands.size() > 0 ?
brands.get(0).getId()
                     : 0) : brandID ;

          orderedDate =
StringUtils.defaultIfBlank(orderedDate,new
SimpleDateFormat("yyyy-MM-dd").format(new Date()));

          Date date = new SimpleDateFormat("yyyy-MM-
dd").parse(orderedDate);
```

BROWSE LIST OF SIGNED UP USERS

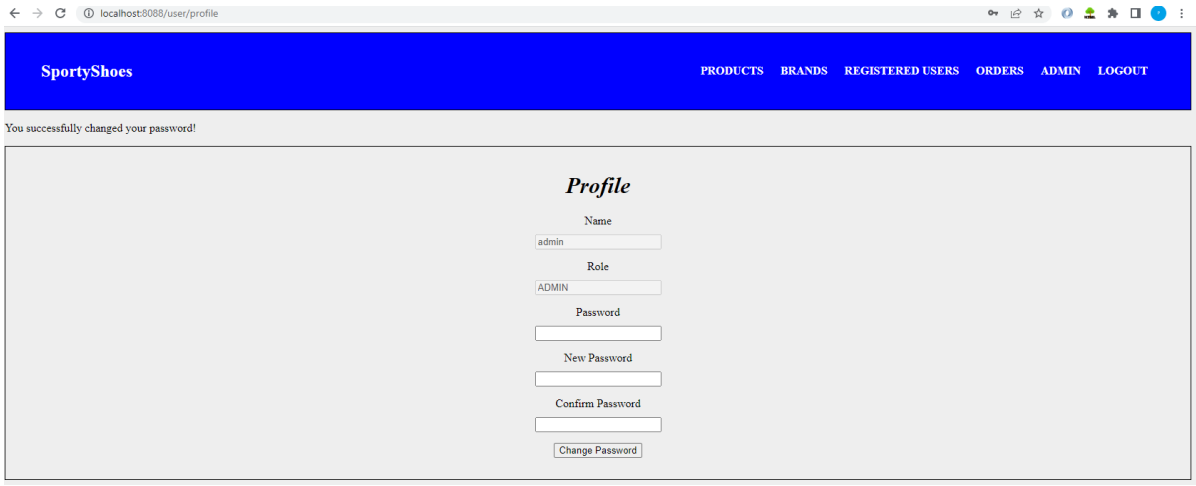
- The admin can view all registered users, and search users by username




```
    @GetMapping("/user/index")
    public ModelAndView listUser(HttpServletRequest
request, HttpServletResponse response
                ) {

          String username =
StringUtils.defaultString(request.getParameter("usernameSea
rch"));
          ModelAndView mv = new ModelAndView();
          List<User> users =
userDao.findByUsernameIsContaining(username);
          mv.addObject("users", users);
          mv.addObject("username", username);

          mv.setViewName("/users.jsp");
          return mv;

    }
```

ADMIN CAN CHANGE PASSWORD

- Logged in users can change their password, by clicking in his/her name in the navigation bar


localhost:8088/user/profile
SportyShoes
PRODUCTS
BRANDS
REGISTERED USERS
ORDERS
ADMIN
LOGOUT
You successfully changed your password!
Profile
Name
admin
Role
ADMIN
Password
New Password
Confirm Password
Change Password

```
@PostMapping("/user/changePassword")
     public ModelAndView changePassword(HttpServletRequest
request, HttpServletResponse response,
                @RequestParam(value = "password", required =
true) String password,
                @RequestParam(value = "newPassword",
required = true) String newPassword,
                @RequestParam(value = "confirmPassword",
required = true) String confirmPassword,
                Authentication authentication,
RedirectAttributes attributes) {
          ModelAndView mv = new
ModelAndView("redirect:/user/profile");
          MyUsersDetails details = (MyUsersDetails)
authentication.getPrincipal();
          if (!newPassword.equals(confirmPassword)) {
                attributes.addFlashAttribute("message", "The
passwords do not match!");
                return mv;
          }

          if (bCryptPasswordEncoder.matches(password,
details.getPassword())) {
                String encodedPassword =
bCryptPasswordEncoder.encode(newPassword);
                userDao.updatePassword(details.getId(),
encodedPassword);

                UserDetails userDetails =
userDetailsServiceImplementation.loadUserByUsername(authent
ication.getName());
                authentication = new
UsernamePasswordAuthenticationToken(userDetails,
userDetails.getPassword(),
                           userDetails.getAuthorities());


     SecurityContextHolder.getContext().setAuthentication(a
uthentication);

                attributes.addFlashAttribute("message", "You
successfully changed your password!");

          } else {
                attributes.addFlashAttribute("message", "You
entered a wrong password!");
          }

          return mv;

     }
```

Core Concepts

- Create Spring boot Apllication
- Used MVC architecture
- Object-Oriented: used to create and model objects for users and their credentials.
- Data Access Object: used to store and retrieve data.
- Data Sources: used to define a set of properties required to identify and access the database
- JpaRepository: used to provide CRUD functionality for Entity Class

Github Repository

https://github.com/pedrole/Learners-Academy.git

Conclusion

FUTURE WORK

- Implement robust Junit Test Suite
- Add Exception Handling
- Implement Cart Functionality
- Add image to products
- Add background image to index page
- Add Logo to navigation bar
- Add css framework (ex: bootstrap)