|  |  |  |  |
| --- | --- | --- | --- |
| Модули | Описание | Количество строк кода | Размер (в Кбайтах) |
| 1 | 2 | 3 | 4 |
| main\java\com\example\demo\DemoApplication.java |  | 12 | 1 |
| main\java\com\example\demo\Service\ConsoleService.java |  | 22 | 1 |
| main\java\com\example\demo\Service\ExportExel.java |  | 139 | 6 |
| main\java\com\example\demo\Service\ProductImageService.java |  | 41 | 2 |
| main\java\com\example\demo\Service\ReportService.java |  | 21 | 1 |
| main\java\com\example\demo\Service\UserService.java |  | 19 | 1 |
| main\java\com\example\demo\config\MvcConfig.java |  | 13 | 1 |
| main\java\com\example\demo\config\WebSecurityConfig.java |  | 48 | 3 |
| main\java\com\example\demo\controllers\BookingController.java |  | 477 | 23 |
| main\java\com\example\demo\controllers\CartController.java |  | 92 | 4 |
| main\java\com\example\demo\controllers\CategoryController.java |  | 71 | 3 |
| main\java\com\example\demo\controllers\ImageController.java |  | 25 | 1 |
| main\java\com\example\demo\controllers\PhoneNumberVerificationController.java |  | 61 | 3 |
| main\java\com\example\demo\controllers\ProductController.java |  | 147 | 7 |
| main\java\com\example\demo\controllers\RegistrationController.java |  | 57 | 3 |
| main\java\com\example\demo\controllers\ReportController.java |  | 3 | 1 |
| main\java\com\example\demo\controllers\RoleController.java |  | 36 | 2 |
| main\java\com\example\demo\controllers\StatusController.java |  | 70 | 3 |
| main\java\com\example\demo\controllers\UserController.java |  | 47 | 2 |
| main\java\com\example\demo\models\Booking.java |  | 109 | 3 |
| main\java\com\example\demo\models\Cart.java |  | 66 | 2 |
| main\java\com\example\demo\models\Category.java |  | 42 | 2 |
| main\java\com\example\demo\models\Image.java |  | 33 | 1 |
| main\java\com\example\demo\models\Product.java |  | 102 | 3 |
| main\java\com\example\demo\models\ProductSort.java |  | 15 | 1 |
| main\java\com\example\demo\models\Role.java |  | 24 | 1 |
| main\java\com\example\demo\models\Status.java |  | 41 | 2 |
| main\java\com\example\demo\models\User.java |  | 73 | 2 |
| main\java\com\example\demo\repo\BookingRepository.java |  | 14 | 1 |
| main\java\com\example\demo\repo\CartRepository.java |  | 11 | 1 |
| main\java\com\example\demo\repo\CategoryRepository.java |  | 11 | 1 |
| main\java\com\example\demo\repo\ImageRepository.java |  | 10 | 1 |
| main\java\com\example\demo\repo\ProductRepository.java |  | 17 | 1 |
| main\java\com\example\demo\repo\StatusRepository.java |  | 7 | 1 |
| main\java\com\example\demo\repo\UserRepos.java |  | 15 | 1 |
| main\resources\application.properties |  | 8 | 1 |
| main\resources\static\jees.js |  | 7 | 1 |
| main\resources\static\toggleCheckbox.scss |  | 105 | 3 |
| main\resources\static\wb.css |  | 101 | 2 |
| main\resources\templates\accept-user\policy-terms.html |  | 186 | 15 |
| main\resources\templates\accept-user\userAccept.html |  | 223 | 19 |
| main\resources\templates\admin\userEdit.html |  | 165 | 5 |
| main\resources\templates\admin\userList.html |  | 268 | 7 |
| main\resources\templates\category\category-add.html |  | 193 | 5 |
| main\resources\templates\category\category-edit.html |  | 192 | 5 |
| main\resources\templates\category\category.html |  | 401 | 10 |
| main\resources\templates\login.html |  | 223 | 6 |
| main\resources\templates\orders\details.html |  | 303 | 8 |
| main\resources\templates\orders\orders-add.html |  | 296 | 9 |
| main\resources\templates\orders\orders-edit.html |  | 291 | 9 |
| main\resources\templates\orders\orders.html |  | 338 | 10 |
| main\resources\templates\orders\withdrawals.html |  | 407 | 13 |
| main\resources\templates\products\products-add.html |  | 266 | 8 |
| main\resources\templates\products\products-edit.html |  | 265 | 8 |
| main\resources\templates\products\products.html |  | 335 | 10 |
| main\resources\templates\registration.html |  | 257 | 8 |
| main\resources\templates\status\status-add.html |  | 197 | 5 |
| main\resources\templates\status\status-edit.html |  | 197 | 5 |
| main\resources\templates\status\status.html |  | 404 | 10 |
| test\java\com\example\demo\DemoApplicationTests.java |  | 13 | 1 |

1. main\java\com\example\demo\DemoApplication.java

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

2. main\java\com\example\demo\Service\ConsoleService.java

package com.example.demo.Service;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

public class ConsoleService {

@Deprecated

public static void exec(String command) throws IOException, InterruptedException {

Process process = Runtime.getRuntime().exec("cmd");

OutputStream stdin = process.getOutputStream();

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stdin));

writer.write(command);

writer.newLine();

writer.flush();

writer.close();

process.waitFor();

}

}

3. main\java\com\example\demo\Service\ExportExel.java

package com.example.demo.Service;

import com.example.demo.models.Booking;

import com.example.demo.models.Cart;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;

import org.apache.poi.hssf.usermodel.HSSFFont;

import org.apache.poi.hssf.util.HSSFColor;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.CellStyle;

import org.apache.poi.ss.usermodel.IndexedColors;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.xssf.usermodel.XSSFCellStyle;

import org.apache.poi.xssf.usermodel.XSSFFont;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServletResponse;

import javax.swing.\*;

import java.io.IOException;

import java.util.List;

public class ExportExel {

private XSSFWorkbook workbook;

private XSSFSheet sheet;

private List<Booking> listReport;

public ExportExel(List<Booking> listReport) {

this.listReport = listReport;

workbook = new XSSFWorkbook();

}

private void writeHeaderLine() {

sheet = workbook.createSheet("Отчеты");

Row row = sheet.createRow(0);

CellStyle style = workbook.createCellStyle();

XSSFCellStyle cellStyle = workbook.createCellStyle();

XSSFFont font = workbook.createFont();

font.setFontName(XSSFFont.DEFAULT\_FONT\_NAME);

font.setFontHeightInPoints((short)10);

font.setColor(IndexedColors.BLUE.getIndex());

cellStyle.setFont(font);

style.setFont(font);

createCell(row, 0, "Номер заказа", style);

createCell(row, 1, "Пользователь", style);

createCell(row, 2, "Номер столика", style);

createCell(row, 3, "Статус заказа", style);

}

private void createCell(Row row, int columnCount, Object value, CellStyle style) {

sheet.autoSizeColumn(columnCount);

Cell cell = row.createCell(columnCount);

if (value instanceof Integer) {

cell.setCellValue((Integer) value);

} else if (value instanceof Boolean) {

cell.setCellValue((Boolean) value);

}else {

cell.setCellValue((String) value);

}

cell.setCellStyle(style);

}

private void writeDataLines() {

int rowCount = 1;

int fullPrice = 0;

CellStyle style = workbook.createCellStyle();

XSSFFont font = workbook.createFont();

font.setFontHeight(14);

style.setFont(font);

for (Booking booking : listReport) {

int fullPriceOneOrder = 0;

Row row = sheet.createRow(rowCount++);

int columnCount = 0;

int columnCount1 = 0;

Row row1 = sheet.createRow(rowCount++);

createCell(row, columnCount++, booking.getId().toString(), style);

createCell(row, columnCount++, booking.getUser().getUsername().toString(), style);

createCell(row, columnCount++, booking.getTableNumber().toString(), style);

createCell(row, columnCount++, booking.getStatus().getNameStatus().toString(), style);

createCell(row1,columnCount1++,"Наименование блюда",style);

createCell(row1,columnCount1++,"Категория",style);

createCell(row1,columnCount1++,"Вес",style);

createCell(row1,columnCount1++,"Цена",style);

createCell(row1,columnCount1++,"Количество",style);

createCell(row1,columnCount1++,"Итоговая цена",style);

for (Cart cart : booking.getCarts()) {

Row row2 = sheet.createRow(rowCount++);

int columnCount2 = 0;

fullPriceOneOrder+=cart.getCount()\*cart.getProduct().getPrice();

fullPrice+=cart.getCount()\*cart.getProduct().getPrice();

createCell(row2, columnCount2++, cart.getProduct().getNameProduct().toString(), style);

createCell(row2, columnCount2++, cart.getProduct().getCategory().getNameCategory().toString(), style);

createCell(row2, columnCount2++, cart.getProduct().getWeight().toString(), style);

createCell(row2, columnCount2++, cart.getProduct().getPrice().toString(), style);

createCell(row2, columnCount2++, cart.getCount(), style);

createCell(row2, columnCount2++, cart.getCount() \* cart.getProduct().getPrice(), style);

}

row1 = sheet.createRow(rowCount++);

columnCount1 = 0;

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "Итого", style);

createCell(row1, columnCount1++, fullPriceOneOrder, style);

}

Row row1 = sheet.createRow(rowCount++);

int columnCount1 = 0;

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "", style);

createCell(row1, columnCount1++, "ИТОГ ПО ВСЕМ ЗАКАЗАМ", style);

createCell(row1, columnCount1++, fullPrice, style);

}

public void export(HttpServletResponse response) throws IOException {

writeHeaderLine();

writeDataLines();

ServletOutputStream outputStream = response.getOutputStream();

workbook.write(outputStream);

workbook.close();

outputStream.close();

}

}

4. main\java\com\example\demo\Service\ProductImageService.java

package com.example.demo.Service;

import com.example.demo.models.Image;

import com.example.demo.models.Product;

import com.example.demo.repo.ImageRepository;

import com.example.demo.repo.ProductRepository;

import lombok.RequiredArgsConstructor;

import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

@Service

@Slf4j

@RequiredArgsConstructor

public class ProductImageService {

private final ProductRepository productRepository;

private final ImageRepository imageRepository;

public void saveImageAndProduct(Product product, MultipartFile file) throws IOException {

if (file != null && !file.isEmpty()) {

Image image = saveImageEntity(file);

imageRepository.save(image);

product.setImage(image);

}

productRepository.save(product);

}

private Image saveImageEntity(MultipartFile file) throws IOException {

Image image = new Image();

image.setName(file.getName());

image.setOriginalFileName(file.getOriginalFilename());

image.setContentType(file.getContentType());

image.setSize(file.getSize());

image.setBytes(file.getBytes());

return image;

}

}

5. main\java\com\example\demo\Service\ReportService.java

package com.example.demo.Service;

import com.example.demo.models.Booking;

import com.example.demo.repo.BookingRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Sort;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

@Service

@Transactional

public class ReportService {

@Autowired

private BookingRepository bookingRepository;

public List<Booking> listAll() {

return bookingRepository.findAll(Sort.by("tableNumber").ascending());

}

}

6. main\java\com\example\demo\Service\UserService.java

package com.example.demo.Service;

import com.example.demo.repo.UserRepos;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

@Service

public class UserService implements UserDetailsService {

@Autowired

private UserRepos userRepository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

return (UserDetails) userRepository.findByUsername(username);

}

}

7. main\java\com\example\demo\config\MvcConfig.java

package com.example.demo.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class MvcConfig implements WebMvcConfigurer {

public void addViewControllers(ViewControllerRegistry registry){

registry.addViewController("/login").setViewName("login");

registry.addViewController("/").setViewName("orders/orders");

}

}

8. main\java\com\example\demo\config\WebSecurityConfig.java

package com.example.demo.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import javax.sql.DataSource;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private DataSource dataSource;

@Autowired

private PasswordEncoder passwordEncoder;

@Bean

public PasswordEncoder getPasswordEncoder() {

return new BCryptPasswordEncoder(8);

}

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/registration", "/login", "/policy-terms/", "/userAccept", "/orders", "/orders/add", "/orders/edit", "/withdrawals").permitAll()

.and()

.formLogin().successForwardUrl("/main").loginPage("/login").permitAll()

.and()

.logout();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication()

.dataSource(dataSource)

.passwordEncoder(passwordEncoder)

.usersByUsernameQuery("select username, password, active from user where username=?")

.authoritiesByUsernameQuery("select u.username, ur.roles from user u inner join user\_role ur on u.id = ur.user\_id" +

" where u.username=?");

}

}

9. main\java\com\example\demo\controllers\BookingController.java

package com.example.demo.controllers;

import com.example.demo.Service.ExportExel;

import com.example.demo.Service.ReportService;

import com.example.demo.models.\*;

import com.example.demo.repo.\*;

import com.itextpdf.text.\*;

import com.itextpdf.text.pdf.BaseFont;

import com.itextpdf.text.pdf.PdfPTable;

import com.itextpdf.text.pdf.PdfWriter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.Resource;

import org.springframework.core.io.ResourceLoader;

import org.springframework.http.HttpHeaders;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.method.P;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.ObjectError;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletResponse;

import java.io.FileOutputStream;

import java.io.IOException;

import java.nio.file.Paths;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.List;

import java.util.stream.IntStream;

@Controller

public class BookingController {

private final BookingRepository bookingRepository;

private final ProductRepository productRepository;

private final ResourceLoader resourceLoader;

private static final String FONT = "fonts\\calibri.ttf";

private final CategoryRepository categoryRepository;

private final StatusRepository statusRepository;

private final UserRepos userRepos;

@Autowired

private ReportService reportService;

private CartRepository cartRepository;

public BookingController(BookingRepository bookingRepository, ResourceLoader resourceLoader, CategoryRepository categoryRepository, ProductRepository productRepository, StatusRepository statusRepository, UserRepos userRepos, CartRepository cartRepository) {

this.bookingRepository = bookingRepository;

this.resourceLoader = resourceLoader;

this.productRepository = productRepository;

this.userRepos = userRepos;

this.categoryRepository = categoryRepository;

this.statusRepository = statusRepository;

this.cartRepository = cartRepository;

}

@RequestMapping("/")

public String blogMain(@ModelAttribute("booking") Booking booking, Model model) {

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

Iterable<Booking> bookings = bookingRepository.findAll();

Iterable<Product> products = productRepository.findAll();

Iterable<User> users = userRepos.findAll();

List<String> tablesArray = new ArrayList<>();

for (int i =0;i< 100;i++){

tablesArray.add(String.valueOf(i+1));

}

for (Booking booking1 : bookings){

tablesArray.removeIf(tableNumber -> (booking1.getTimeDeparture() == null || booking1.getTimeDeparture().after(new Date())) && Integer.parseInt(tableNumber) == booking1.getTableNumber());

}

model.addAttribute("products", products);

model.addAttribute("User", users);

model.addAttribute("bookings", bookings);

model.addAttribute("freeNumberTables", tablesArray);

model.addAttribute("isAdmin", authentication.getAuthorities().toString().contains("ADMIN"));

model.addAttribute("isSaler", authentication.getAuthorities().toString().contains("SALER"));

model.addAttribute("isUser", authentication.getAuthorities().toString().contains("USER"));

return "orders/orders";

}

@GetMapping("/orders/add")

public String ordersAdd(@ModelAttribute("booking") @Validated Booking booking, Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

Iterable<Product>

products = productRepository.findAll();

model.addAttribute("products", products);

return "orders/orders-add";

}

@GetMapping("/orders/search")

public String getSearch(@RequestParam(required = false) String search, @RequestParam(required = false) Booking booking, Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

Iterable<Product> products;

if (search != null && !search.equals("")) {

products = productRepository.findByNameProductContains(search);

int count = 0;

for (Product product : products) {

count++;

}

if (count == 0) {

products = productRepository.findAllByCategoryNameCategoryContains(search);

}

} else {

products = productRepository.findAll();

}

int productCount = 0;

for (Product product : products) {

productCount++;

}

model.addAttribute("productCount", productCount);

model.addAttribute("products", products);

model.addAttribute("booking", booking);

int cartCount = 0;

for (Cart cart : booking.getCarts()) {

cartCount += cart.getCount();

}

model.addAttribute("cartCount", cartCount);

return "orders/orders-add";

}

@PostMapping("/orders/add")

public String ordersAdd(@ModelAttribute("booking") Booking booking, BindingResult result, Model model) {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

Booking booking1 = bookingRepository.findTopByTableNumberOrderByIdDesc(booking.getTableNumber());

if (booking1 != null && (booking1.getTimeDeparture() == null || booking1.getTimeDeparture().after(new Date()))) {

result.addError(new ObjectError("tableNumber", "Данный стол уже занят!"));

String[] errorsMessage = {"\*Данный стол уже занят!", "Попробуйте ввести другой"};

model.addAttribute("ErrorMassage",errorsMessage );

}

if (result.hasErrors()) {

Iterable<Booking> bookings = bookingRepository.findAll();

Iterable<Product> products = productRepository.findAll();

Iterable<User> users = userRepos.findAll();

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

model.addAttribute("products", products);

model.addAttribute("User", users);

model.addAttribute("bookings", bookings);

model.addAttribute("freeNumberTables", IntStream.range(1,100).toArray());

model.addAttribute("isAdmin", auth.getAuthorities().toString().contains("ADMIN"));

model.addAttribute("isSaler", auth.getAuthorities().toString().contains("SALER"));

model.addAttribute("isUser", auth.getAuthorities().toString().contains("USER"));

return "orders/orders";

}

Iterable<Product> products = productRepository.findAll();

Iterable<User> users = userRepos.findAll();

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

model.addAttribute("users", users);

model.addAttribute("products", products);

booking.setStatus(statusRepository.findFirstByOrderByIdAsc());

booking.setTimeArrival(new Date());

bookingRepository.save(booking);

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

int cartCount = 0;

for (Cart cart : carts) {

cartCount += cart.getCount();

}

model.addAttribute("cartCount", cartCount);

return "orders/orders-add";

}

@GetMapping("/orders/{id}/edit")

public String ordersEdit(@PathVariable("id") long id, Model model) {

Iterable<Product> products = productRepository.findAll();

Booking booking = bookingRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Неверный id: " + id));

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

model.addAttribute("booking", booking);

model.addAttribute("products", products);

Iterable<Status> status = statusRepository.findAll();

model.addAttribute("statuses", status);

model.addAttribute("carts", cartRepository.findAllByBooking(booking));

int cartCount = 0;

int fullPrice = 0;

for (Cart cart : carts) {

cartCount += cart.getCount();

fullPrice += cart.getCount() \* cart.getProduct().getPrice();

}

model.addAttribute("cartCount", cartCount);

model.addAttribute("fullPrice", fullPrice);

return "orders/orders-edit";

}

@PostMapping("/orders/{id}/edit")

public String ordersUpdate(@PathVariable("id") long id,

@ModelAttribute("booking")

Booking booking, Model model) {

booking.setId(id);

Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

User user = userRepos.findByUsername(authentication.getName());

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

Iterable<Status> status = statusRepository.findAll();

model.addAttribute("statuses", status);

booking.setUser(user);

booking.setTimeArrival(new Date());

bookingRepository.save(booking);

return "redirect:/login";

}

@PostMapping("/orders/details")

public String detailedDish(@RequestParam Booking booking,

@RequestParam Product product,

Model model) {

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

int cartCount = 0;

for (Cart cart : carts) {

cartCount += cart.getCount();

}

model.addAttribute("cartCount", cartCount);

model.addAttribute("booking", booking);

model.addAttribute("product", product);

return "orders/details";

}

@PostMapping("/orders/return")

public String ordersReturn(@RequestParam Booking booking, Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

Iterable<Product> products =

productRepository.findAll();

model.addAttribute("products", products);

model.addAttribute("booking", booking);

int cartCount = 0;

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

for (Cart cart : carts) {

cartCount += cart.getCount();

}

model.addAttribute("booking", booking);

model.addAttribute("cartCount", cartCount);

return "orders/orders-add";

}

@PostMapping("/hiddenBoughtOrders")

public String hiddenBoughtOrders(@RequestParam (required = false) boolean payment, Model model){

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

Iterable<Booking> bookings = bookingRepository.findAllByUser(userRepos.findByUsername(auth.getName()));

if (auth.getAuthorities().toString().contains("SALER"))

bookings = bookingRepository.findAll();

UserDetails userDetails = (UserDetails) auth.getPrincipal();

model.addAttribute("isAuth", userDetails.getUsername());

List<User> users = (List<User>) userRepos.findAll();

model.addAttribute("users", users);

model.addAttribute("isSaler", auth.getAuthorities().toString().contains("SALER"));

model.addAttribute("isUser", auth.getAuthorities().toString().contains("USER"));

model.addAttribute("bookings", bookings);

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

Iterable<Status> statuses = statusRepository.findAll();

model.addAttribute("statuses", statuses);

List<Cart> carts = new ArrayList<>();

List<String> fullPrices = new ArrayList<>();

int bookingCount = 0;

for (Booking booking1 : bookings) {

Iterable<Cart> carts1 = cartRepository.findAllByBooking(booking1);

carts.addAll((List) carts1);

int fullPrice = 0;

for (Cart cart : carts1) {

fullPrice += cart.getCount() \* cart.getProduct().getPrice();

}

fullPrices.add("" + fullPrice + "");

if (!booking1.isPayment()) bookingCount++;

}

model.addAttribute("bookingCount", bookingCount);

model.addAttribute("fullPrice", fullPrices);

model.addAttribute("carts", carts);

model.addAttribute("payment", payment);

return "orders/withdrawals";

}

@GetMapping("/withdrawals")

public String withdrawals(Model model) {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

Iterable<Booking> bookings = bookingRepository.findAllByUser(userRepos.findByUsername(auth.getName()));

if (auth.getAuthorities().toString().contains("SALER"))

bookings = bookingRepository.findAll();

UserDetails userDetails = (UserDetails) auth.getPrincipal();

model.addAttribute("isAuth", userDetails.getUsername());

List<User> users = (List<User>) userRepos.findAll();

model.addAttribute("users", users);

model.addAttribute("isSaler", auth.getAuthorities().toString().contains("SALER"));

model.addAttribute("isUser", auth.getAuthorities().toString().contains("USER"));

model.addAttribute("bookings", bookings);

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

Iterable<Status> statuses = statusRepository.findAll();

model.addAttribute("statuses", statuses);

List<Cart> carts = new ArrayList<>();

List<String> fullPrices = new ArrayList<>();

int bookingCount = 0;

for (Booking booking1 : bookings) {

Iterable<Cart> carts1 = cartRepository.findAllByBooking(booking1);

carts.addAll((List) carts1);

int fullPrice = 0;

for (Cart cart : carts1) {

fullPrice += cart.getCount() \* cart.getProduct().getPrice();

}

fullPrices.add("" + fullPrice + "");

if (!booking1.isPayment()) bookingCount++;

}

model.addAttribute("bookingCount", bookingCount);

model.addAttribute("fullPrice", fullPrices);

model.addAttribute("carts", carts);

return "orders/withdrawals";

}

@PostMapping("/withdrawals")

public String wildMain(@ModelAttribute("booking")

Booking booking, Model model) {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

Iterable<Booking> bookings = bookingRepository.findAllByUser(userRepos.findByUsername(auth.getName()));

if (auth.getAuthorities().toString().contains("SALER"))

bookings = bookingRepository.findAll();

UserDetails userDetails = (UserDetails) auth.getPrincipal();

model.addAttribute("isAuth", userDetails.getUsername());

List<User> users = (List<User>) userRepos.findAll();

model.addAttribute("users", users);

model.addAttribute("isSaler", auth.getAuthorities().toString().contains("SALER"));

model.addAttribute("isUser", auth.getAuthorities().toString().contains("USER"));

model.addAttribute("bookings", bookings);

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

Iterable<Status> status = statusRepository.findAll();

model.addAttribute("statuses", status);

booking.setUser((User) users);

bookingRepository.save(booking);

return "orders/withdrawals";

}

@GetMapping("/report/excel")

public void exportToExcel(HttpServletResponse response) throws IOException {

response.setContentType("application/octet-stream");

DateFormat dateFormatter = new SimpleDateFormat("yyyy-MM-dd\_HH:mm:ss");

String currentDateTime = dateFormatter.format(new Date());

String headerKey = "Content-Disposition";

String headerValue = "attachment; filename=reports\_" + currentDateTime + ".xlsx";

response.setHeader(headerKey, headerValue);

List<Booking> listReport = reportService.listAll();

ExportExel excelExporter = new ExportExel(listReport);

excelExporter.export(response);

}

@PostMapping("/orders/{id}/remove")

public String deleteOrder(@PathVariable("id") long id, Model model) {

Booking booking = bookingRepository.findById(id).orElseThrow();

bookingRepository.delete(booking);

return "redirect:/withdrawals";

}

@PostMapping("/orders/save")

public String saveStatus(@RequestParam Status status, @RequestParam Booking booking) {

booking.setStatus(status);

bookingRepository.save(booking);

return "redirect:/withdrawals";

}

@PostMapping("/pay")

public ResponseEntity<Resource> exportPDF(@RequestParam Booking booking) throws IOException, DocumentException {

booking.setPayment(true);

Calendar calendar = Calendar.getInstance();

calendar.setTime(booking.getTimeArrival());

calendar.add(Calendar.MINUTE, 1);

booking.setTimeDeparture(calendar.getTime());

bookingRepository.save(booking);

Document document = new Document();

Resource resource = resourceLoader.getResource(Paths.get("fonts\\cheque.pdf").toUri().toString());

PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream(resource.getFile()));

document.open();

BaseFont baseFont = BaseFont.createFont(FONT, BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);

Font font = new Font(baseFont, 11, Font.NORMAL);

addTable(document, font, booking);

document.close();

return ResponseEntity.ok()

.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=" + resource.getFilename())

.contentLength(resource.contentLength())

.body(resource);

}

public void addTable(Document document, Font font, Booking booking) throws DocumentException {

Paragraph paragraph = new Paragraph("Кассовый чек", font);

paragraph.setAlignment(Element.ALIGN\_CENTER);

document.add(paragraph);

paragraph = new Paragraph("Whitespace", font);

font.setColor(BaseColor.WHITE);

paragraph.setAlignment(Element.ALIGN\_CENTER);

document.add(paragraph);

font.setColor(BaseColor.BLACK);

PdfPTable table = new PdfPTable(6);

font.setSize(11);

String[] items = {"Логин пользователя", "Номер заказа", "Номер столика", "Время выдачи чека"};

String[] items1 = {"Наименование", "Категория", "Вес", "Цена", "Количество", "Итоговая цена"};

for (int i = 0; i < 6; i++) {

if (i == 4 || i == 5) {

table.addCell(new Paragraph("", font));

continue;

}

table.addCell(new Paragraph(items[i], font));

}

int fullPrice = 0;

for (int i = 0; i < 6; i++) {

switch (i) {

case 0 -> table.addCell(new Paragraph(booking.getUser().getUsername(), font));

case 1 -> table.addCell(new Paragraph(booking.getId().toString(), font));

case 2 -> table.addCell(new Paragraph(booking.getTableNumber().toString(), font));

case 3 -> table.addCell(new Paragraph(booking.getTimeArrival().toString().split("\\.")[0], font));

default -> table.addCell(new Paragraph("", font));

}

if (i == 5) {

for (int j = 0; j < 6; j++) {

table.addCell(new Paragraph(items1[j], font));

}

for (Cart cart : booking.getCarts()) {

fullPrice+=cart.getCount()\*cart.getProduct().getPrice();

table.addCell(new Paragraph(cart.getProduct().getNameProduct(), font));

table.addCell(new Paragraph(cart.getProduct().getCategory().getNameCategory(), font));

table.addCell(new Paragraph(cart.getProduct().getWeight().toString(), font));

table.addCell(new Paragraph(cart.getProduct().getPrice().toString(), font));

table.addCell(new Paragraph(String.valueOf(cart.getCount()), font));

table.addCell(new Paragraph(String.valueOf(cart.getCount() \* cart.getProduct().getPrice()), font));

}

}

}

font.setStyle(Font.NORMAL);

table.addCell(new Paragraph(""));

table.addCell(new Paragraph(""));

table.addCell(new Paragraph(""));

table.addCell(new Paragraph(""));

table.addCell(new Paragraph("ИТОГО",font));

table.addCell(new Paragraph(String.valueOf(fullPrice),font));

document.add(table);

}

}

10. main\java\com\example\demo\controllers\CartController.java

package com.example.demo.controllers;

import com.example.demo.models.\*;

import com.example.demo.repo.\*;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

@Controller

public class CartController {

public final BookingRepository bookingRepository;

public final ProductRepository productRepository;

private final StatusRepository statusRepository;

private final CategoryRepository categoryRepository;

private final CartRepository cartRepository;

public CartController(BookingRepository bookingRepository, CategoryRepository categoryRepository, ProductRepository productRepository, StatusRepository statusRepository, CartRepository cartRepository) {

this.bookingRepository = bookingRepository;

this.productRepository = productRepository;

this.statusRepository = statusRepository;

this.categoryRepository = categoryRepository;

this.cartRepository = cartRepository;

}

@PostMapping("/cart/add")

public String addToCart(@ModelAttribute("booking") Booking booking, @RequestParam(required = false) Product product, Model model) {

Iterable<Status> status = statusRepository.findAll();

model.addAttribute("statuses", status);

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

Cart cart;

if (cartRepository.findCartByBookingAndProduct(booking, product) != null) {

cart = cartRepository.findCartByBookingAndProduct(booking, product);

cart.setCount(cart.getCount() + 1);

} else {

cart = new Cart(1, booking, product);

}

cartRepository.save(cart);

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

int cartCount = 0;

for(Cart cartt : carts){

cartCount += cartt.getCount();

}

model.addAttribute("cartCount", cartCount);

return "orders/orders-add";

}

@PostMapping("/cart/delete")

public String deleteProductCart(@RequestParam Booking booking, @RequestParam Cart cart) {

cartRepository.delete(cart);

return "redirect:/orders/" + booking.getId() + "/edit";

}

@PostMapping("/cart/remove1")

public String delete1ProductCart(@RequestParam Booking booking, @RequestParam Cart cart) {

if (cart.getCount() -1 == 0 ){

cartRepository.delete(cart);

}

else {

cart.setCount(cart.getCount()-1);

cartRepository.save(cart);

}

return "redirect:/orders/" + booking.getId() + "/edit";

}

@PostMapping("/cart/plus1")

public String plus1ProductCart(@RequestParam Booking booking, @RequestParam Cart cart) {

cart.setCount(cart.getCount()+1);

cartRepository.save(cart);

return "redirect:/orders/" + booking.getId() + "/edit";

}

@GetMapping("/cart/returnToOrder")

public String returnToOrder(@RequestParam Booking booking, Model model) {

Iterable<Cart> carts = cartRepository.findAllByBooking(booking);

int cartCount = 0;

for(Cart cart : carts){

cartCount += cart.getCount();

}

model.addAttribute("cartCount", cartCount);

model.addAttribute("booking", booking);

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

Iterable<Product> products = productRepository.findAll();

model.addAttribute("products", products);

return "orders/orders-add";

}

}

11. main\java\com\example\demo\controllers\CategoryController.java

package com.example.demo.controllers;

import com.example.demo.models.Category;

import com.example.demo.repo.CategoryRepository;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

@PreAuthorize("hasAnyAuthority('SALER')")

public class CategoryController {

private final CategoryRepository categoryRepository;

public CategoryController(CategoryRepository categoryRepository) {

this.categoryRepository = categoryRepository;

}

@GetMapping("/category")

public String categoryMain(Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

return "category/category";

}

@GetMapping("/category/add")

public String categoryAdd(@ModelAttribute("category") Category category, Model model) {

return "category/category-add";

}

@PostMapping("/category/add")

public Object categoryAdd(@ModelAttribute("category") @Validated Category category, BindingResult bindingResult, Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

if (bindingResult.hasErrors()) return "category/category";

categoryRepository.save(category);

return "redirect:/category";

}

@GetMapping("/category/{id}/edit")

public String categoryEdit(@ModelAttribute("category") @PathVariable("id") long id, Model model) {

Category res = categoryRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Неверный id: " + id));

model.addAttribute("category", res);

return "category/category-edit";

}

@PostMapping("/category/{id}/edit")

public String categoryUpdate(@PathVariable("id") long id, @ModelAttribute("category")

Category category, BindingResult bindingResult) {

category.setId(id);

if (bindingResult.hasErrors()) {

return "category/category-edit";

}

categoryRepository.save(category);

return "redirect:/category";

}

@PostMapping("/category/{id}/remove")

public String categoryRemove(@PathVariable("id") long id, Model model) {

Category category = categoryRepository.findById(id).orElseThrow();

categoryRepository.delete(category);

return "redirect:/category";

}

}

12. main\java\com\example\demo\controllers\ImageController.java

package com.example.demo.controllers;

import com.example.demo.models.Image;

import org.springframework.core.io.InputStreamResource;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.io.ByteArrayInputStream;

@RestController

@RequestMapping("/image")

public class ImageController {

@GetMapping("{image}")

private ResponseEntity<?> imageIndex(@PathVariable Image image){

return ResponseEntity.ok()

.contentType(MediaType.valueOf(image.getContentType()))

.contentLength(image.getSize())

.body(new InputStreamResource(new ByteArrayInputStream(image.getBytes())));

}

}

13. main\java\com\example\demo\controllers\PhoneNumberVerificationController.java

package com.example.demo.controllers;

import com.twilio.Twilio;

import com.twilio.rest.verify.v2.service.Verification;

import com.twilio.rest.verify.v2.service.VerificationCheck;

import lombok.extern.slf4j.Slf4j;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.time.LocalDateTime;

import java.util.Random;

import static com.twilio.example.ValidationExample.ACCOUNT\_SID;

import static com.twilio.example.ValidationExample.AUTH\_TOKEN;

@Controller

@Slf4j

public class PhoneNumberVerificationController {

@GetMapping("/generateOTP")

public String generateOTP(){

Twilio.init("ACd20cff29460f8a65997087a81b68141e","e7dbc249e080bbb7ba00d905b5f84a7f");

Verification verification = Verification.creator(

"VA954765dc76826bc3895c459da6744b6f", // this is your verification sid

"+79530994567", //this is your Twilio verified recipient phone number

"sms") // this is your channel type

.create();

System.out.println(verification.getStatus());

log.info("OTP has been successfully generated, and awaits your verification {}", LocalDateTime.now());

return "redirect:/login";

}

@GetMapping("/verifyOTP")

public ResponseEntity<?> verifyUserOTP() throws Exception {

Twilio.init(ACCOUNT\_SID, AUTH\_TOKEN);

try {

Random random = new Random();

random.nextInt(1000,9999);

VerificationCheck verificationCheck = VerificationCheck.creator(

"VA954765dc76826bc3895c459da6744b6f")

.setTo("+79530994567")

.setCode(random.toString())

.create();

System.out.println(verificationCheck.getStatus());

} catch (Exception e) {

return new ResponseEntity<>("Verification failed.", HttpStatus.BAD\_REQUEST);

}

return new ResponseEntity<>("This user's verification has been completed successfully", HttpStatus.OK);

}

}

14. main\java\com\example\demo\controllers\ProductController.java

package com.example.demo.controllers;

import com.example.demo.Service.ProductImageService;

import com.example.demo.models.Category;

import com.example.demo.models.Product;

import com.example.demo.models.ProductSort;

import com.example.demo.repo.CategoryRepository;

import com.example.demo.repo.ProductRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.security.access.prepost.PostAuthorize;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.ObjectError;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import javax.validation.Valid;

import java.io.IOException;

@Controller

@PreAuthorize("hasAnyAuthority('SALER')")

public class ProductController {

private final ProductRepository productRepository;

private final ProductImageService productImageService;

private final CategoryRepository categoryRepository;

public ProductController(ProductRepository productRepository, ProductImageService productImageService, CategoryRepository categoryRepository) {

this.productRepository = productRepository;

this.productImageService = productImageService;

this.categoryRepository = categoryRepository;

}

@GetMapping("/products")

public String productsMain(Model model) {

Iterable<Product> products = productRepository.findAll(PageRequest.of(0, 5));

int size = 0;

for (Product product : productRepository.findAll()){

size+=1;

}

size = size/5;

model.addAttribute("size",size);

model.addAttribute("products", products);

model.addAttribute("offset",0);

return "products/products";

}

@PostMapping("/hiddenDish")

public String hiddenDish(@RequestParam Product product) {

product.setHiddenDish(!product.isHiddenDish());

productRepository.save(product);

return "redirect:/products";

}

@GetMapping("/products/add")

public String productsAdd(@ModelAttribute("products") Product product, Model model) {

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

return "products/products-add";

}

@GetMapping("productsSort")

public String getAllAndEnumSort(

@RequestParam("offset") Integer offset, Model model){

offset-=1;

Iterable<Product> products;

if(offset < 0){

offset = 0;

products = productRepository.findAll(PageRequest.of(offset, 5));

}

else {

products = productRepository.findAll(PageRequest.of(offset, 5));

}

int size = 0;

for (Product product : productRepository.findAll()){

size+=1;

}

size = (int) Math.ceil(size/5);

if(size < offset){

offset = size;

products = productRepository.findAll(PageRequest.of(offset, 5));

}

model.addAttribute("size",size);

model.addAttribute("offset",offset);

model.addAttribute("products",products);

return "products/products";

}

@PostMapping("/products/add")

public Object productsAdd(@ModelAttribute("products") @Valid Product product, BindingResult bindingResult,

@RequestParam("file") MultipartFile file, Model model) throws IOException {

if (file.isEmpty()) {

bindingResult.addError(new ObjectError("image", "Изображение товара не должно быть пустым!"));

model.addAttribute("errorMessageImage", "Изображение товара не должно быть пустым!");

}

if (bindingResult.hasErrors()) return "products/products-add";

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

productImageService.saveImageAndProduct(product, file);

return "redirect:/products";

}

@GetMapping("/products/{id}/edit")

public String productsEdit(@PathVariable("id") long id, Model model) {

Product product = productRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Неверный id: " + id));

model.addAttribute("products", product);

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

return "products/products-edit";

}

@PostMapping("/products/{id}/edit")

public String productsUpdate(@PathVariable("id") long id,

@ModelAttribute("products")

@Validated Product product,

@RequestParam("file") MultipartFile file,

BindingResult bindingResult, Model model) throws IOException {

if (file.isEmpty()) {

bindingResult.addError(new ObjectError("image", "Изображение товара не должно быть пустым!"));

model.addAttribute("errorMessageImage", "Изображение товара не должно быть пустым!");

}

product.setId(id);

if (bindingResult.hasErrors()) {

return "products/products-edit";

}

Iterable<Category> categories = categoryRepository.findAll();

model.addAttribute("categories", categories);

productRepository.save(product);

productImageService.saveImageAndProduct(product, file);

return "redirect:/products";

}

@PostMapping("/products/{id}/remove")

public String productsRemove(@PathVariable("id") long id, Model model) {

Product product = productRepository.findById(id).orElseThrow();

productRepository.delete(product);

return "redirect:/products";

}

}

15. main\java\com\example\demo\controllers\RegistrationController.java

package com.example.demo.controllers;

import com.example.demo.models.Role;

import com.example.demo.models.User;

import com.example.demo.repo.UserRepos;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.ObjectError;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;

import java.util.Collections;

@Controller

public class RegistrationController {

@Autowired

private UserRepos userRepository;

@Autowired

private PasswordEncoder passwordEncoder;

@GetMapping("/registration")

public String registration(@ModelAttribute("user") User user){

return "registration";

}

@PostMapping("/registration")

public String registration(@ModelAttribute("user") @Valid User user, BindingResult bindingResultUser,

Model model) {

if (userRepository.findByUsername(user.getUsername()) != null) {

bindingResultUser.addError(new ObjectError("username", "Данный логин уже занят"));

model.addAttribute("errorMessageUsername", "Данный логин уже занят");

}

if (bindingResultUser.hasErrors())

return "registration";

user.setPassword(passwordEncoder.encode(user.getPassword()));

user.setActive(true);

user.setRoles(Collections.singleton(Role.USER));

userRepository.save(user);

return "redirect:/login";

}

@GetMapping("/userAccept")

public String UserAccept(@ModelAttribute("user") User user){

return "accept-user/userAccept";

}

@GetMapping("/policy-terms")

public String PolicyTerms(@ModelAttribute("user") User user){

return "accept-user/policy-terms";

}

}

16. main\java\com\example\demo\controllers\ReportController.java

package com.example.demo.controllers;

public class ReportController { }

17. main\java\com\example\demo\controllers\RoleController.java

package com.example.demo.controllers;

import com.example.demo.models.Booking;

import com.example.demo.models.Role;

import com.example.demo.models.User;

import com.example.demo.repo.BookingRepository;

import com.example.demo.repo.UserRepos;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

public class RoleController {

private final UserRepos userRepos;

private final BookingRepository bookingRepository;

public RoleController(UserRepos userRepos, BookingRepository bookingRepository) {

this.userRepos = userRepos;

this.bookingRepository = bookingRepository;

}

@PostMapping("/main")

public String roleMain(){

if (getRole(SecurityContextHolder.getContext().getAuthentication()) == Role.USER) {

Booking booking = bookingRepository.findFirstByOrderByIdDesc();

User user = userRepos.findByUsername(SecurityContextHolder.getContext().getAuthentication().getName());

if (booking.getUser() == null) booking.setUser(user);

bookingRepository.save(booking);

}

return "redirect:/";

}

public Role getRole(Authentication authentication){

User user = userRepos.findByUsername(authentication.getName());

return (Role) user.getRoles().toArray()[0];

}

}

18. main\java\com\example\demo\controllers\StatusController.java

package com.example.demo.controllers;

import com.example.demo.models.Status;

import com.example.demo.repo.StatusRepository;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

@PreAuthorize("hasAnyAuthority('SALER')")

public class StatusController {

private final StatusRepository statusRepository;

public StatusController(StatusRepository statusRepository) {

this.statusRepository = statusRepository;

}

@GetMapping("/status")

public String statusMain(Model model) {

Iterable<Status> status = statusRepository.findAll();

model.addAttribute("status", status);

return "status/status";

}

@GetMapping("/status/add")

public String statusAdd(@ModelAttribute("status") Status status, Model model) {

return "status/status-add";

}

@PostMapping("/status/add")

public Object statusAdd(@ModelAttribute("status") @Validated Status status, BindingResult bindingResult, Model model) {

Iterable<Status> statuses = statusRepository.findAll();

model.addAttribute("status", statuses);

if (bindingResult.hasErrors()) return "status";

statusRepository.save(status);

return "redirect:/status";

}

@GetMapping("/status/{id}/edit")

public String statusEdit(@ModelAttribute("status") @PathVariable("id") long id, Model model) {

Status res = statusRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Неверный id: " + id));

model.addAttribute("status", res);

return "status/status-edit";

}

@PostMapping("/status/{id}/edit")

public String statusUpdate(@PathVariable("id") long id, @ModelAttribute("status")

Status status, BindingResult bindingResult) {

status.setId(id);

if (bindingResult.hasErrors()) {

return "status/status-edit";

}

statusRepository.save(status);

return "redirect:/status";

}

@PostMapping("/status/{id}/remove")

public String statusRemove(@PathVariable("id") long id, Model model) {

Status status = statusRepository.findById(id).orElseThrow();

statusRepository.delete(status);

return "redirect:/status";

}

}

19. main\java\com\example\demo\controllers\UserController.java

package com.example.demo.controllers;

import com.example.demo.models.Role;

import com.example.demo.models.User;

import com.example.demo.repo.UserRepos;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Optional;

@Controller

@RequestMapping("/admin")

@PreAuthorize("hasAnyAuthority('ADMIN')")

public class UserController {

@Autowired

private UserRepos userRepository;

@GetMapping

public String userList(Model model) {

model.addAttribute("users", userRepository.findAll());

return "admin/userList";

}

@GetMapping("/{id}/edit")

public String userEdit(@PathVariable(value = "id") long id, Model model) {

User user = userRepository.findById(id).get();

model.addAttribute("user", user);

model.addAttribute("rolesList", Role.values());

return "admin/userEdit";

}

@PostMapping

public String userSave(@ModelAttribute User user) {

userRepository.save(user);

return "redirect:/admin";

}

@PostMapping("/delete")

public String userRemove(@RequestParam User user) {

userRepository.delete(user);

return "redirect:/admin";

}

}

20. main\java\com\example\demo\models\Booking.java

package com.example.demo.models;

import org.hibernate.validator.constraints.Range;

import javax.persistence.\*;

import javax.validation.constraints.\*;

import java.util.Date;

import java.util.List;

//Модель заказа

@Entity

public class Booking {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@Range(min = 1, max = 100, message = "Диапазон от 1 до 100")

@NotNull

private Integer tableNumber;

private boolean payment;

private Date timeArrival;

private Date timeDeparture;

@ManyToOne(optional = true)

private User user;

@ManyToOne

@JoinColumn(name = "statusId", referencedColumnName = "id")

private Status status;

@OneToMany(mappedBy = "booking", cascade = CascadeType.REMOVE)

private List<Cart> carts;

public Booking(Integer tableNumber, Date timeArrival, Date timeDeparture, User user,Status status) {

this.tableNumber = tableNumber;

this.timeArrival = timeArrival;

this.timeDeparture = timeDeparture;

this.user = user;

this.status = status;

}

public Booking(){}

public List<Cart> getCarts() {

return carts;

}

public boolean isPayment() {

return payment;

}

public void setPayment(boolean payment) {

this.payment = payment;

}

public void setCarts(List<Cart> carts) {

this.carts = carts;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Integer getTableNumber() {

return tableNumber;

}

public void setTableNumber(Integer tableNumber) {

this.tableNumber = tableNumber;

}

public Date getTimeArrival() {

return timeArrival;

}

public void setTimeArrival(Date timeArrival) {

this.timeArrival = timeArrival;

}

public Date getTimeDeparture() {

return timeDeparture;

}

public void setTimeDeparture(Date timeDeparture) {

this.timeDeparture = timeDeparture;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Status getStatus() {

return status;

}

public void setStatus(Status status) {

this.status = status;

}

}

21. main\java\com\example\demo\models\Cart.java

package com.example.demo.models;

import org.hibernate.validator.constraints.Range;

import javax.persistence.\*;

import javax.validation.constraints.NotNull;

@Entity

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@NotNull(message = "Поле не может быть пустым")

@Range(min = 1, max = 100, message = "Диапазон от 1 до 100")

private int count;

@ManyToOne

@JoinColumn(name = "bookingId", referencedColumnName = "id")

private Booking booking;

@ManyToOne

@JoinColumn(name = "productId", referencedColumnName = "id")

private Product product;

public Cart() {

}

public Cart(int count, Booking booking, Product product) {

this.count = count;

this.booking = booking;

this.product = product;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public Booking getBooking() {

return booking;

}

public void setBooking(Booking booking) {

this.booking = booking;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

22. main\java\com\example\demo\models\Category.java

package com.example.demo.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.Size;

@Entity

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@NotEmpty(message = "Поле не может быть пустым")

@Size(min = 1, max = 20, message = "От 1 до 20 символов")

private String nameCategory;

public Category(Long id, String nameCategory) {

this.id = id;

this.nameCategory = nameCategory;

}

public Category(){}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getNameCategory() {

return nameCategory;

}

public void setNameCategory(String nameCategory) {

this.nameCategory = nameCategory;

}

}

23. main\java\com\example\demo\models\Image.java

package com.example.demo.models;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.\*;

@Entity

@AllArgsConstructor

@NoArgsConstructor

@Data

public class Image {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long idImage;

private String name;

private String originalFileName;

private Long size;

private String contentType;

@Lob

private byte[] bytes;

@OneToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER, optional = true, mappedBy = "image")

private Product product;

}

24. main\java\com\example\demo\models\Product.java

package com.example.demo.models;

import org.hibernate.validator.constraints.Range;

import javax.persistence.\*;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@NotEmpty(message = "Поле не может быть пустым")

@Size(min = 1, max = 100, message = "От 1 до 100 символов")

private String nameProduct;

@NotNull(message = "Поле не может быть пустым")

@Range(min = 1, max = 1000, message = "Диапазон от 1 до 5")

private Integer price;

@NotNull(message = "Поле не может быть пустым")

@Range(min = 1, max = 1000, message = "От 1 до 100 символов")

private Integer weight;

private boolean hiddenDish;

private String details;

@OneToOne(optional = true, cascade = CascadeType.ALL)

@JoinColumn(name = "imageId")

private Image image;

@ManyToOne

@JoinColumn(name = "categoryId", referencedColumnName = "id")

private Category category;

public Product(String nameProduct, Integer price, Integer weight, boolean hiddenDish, String details, Image image, Category category) {

this.nameProduct = nameProduct;

this.price = price;

this.weight = weight;

this.hiddenDish = hiddenDish;

this.details = details;

this.image = image;

this.category = category;

}

public String getDetails() {

return details;

}

public void setDetails(String details) {

this.details = details;

}

public boolean isHiddenDish() {

return hiddenDish;

}

public void setHiddenDish(boolean hiddenDish) {

this.hiddenDish = hiddenDish;

}

public Category getCategory() {

return category;

}

public void setCategory(Category category) {

this.category = category;

}

public Product() {}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

public Integer getPrice() {return price;}

public void setPrice(Integer price) {this.price = price;}

public Integer getWeight() {return weight;}

public void setWeight(Integer weight) {this.weight = weight;}

public String getNameProduct() {return nameProduct;}

public void setNameProduct(String nameProduct) {this.nameProduct = nameProduct;}

}

25. main\java\com\example\demo\models\ProductSort.java

package com.example.demo.models;

import lombok.Getter;

import lombok.RequiredArgsConstructor;

import org.springframework.data.domain.Sort;

@Getter

@RequiredArgsConstructor

public enum ProductSort {

ID\_ASC(Sort.by(Sort.Direction.ASC, "id")),

ID\_DESC(Sort.by(Sort.Direction.DESC, "id"));

private final Sort sortValue;

}

26. main\java\com\example\demo\models\Role.java

package com.example.demo.models;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {

USER("Пользователь"),

ADMIN("Администратор"),

SALER("Сотрудник");

public String getRoleName() {

return roleName;

}

private final String roleName;

Role(String roleName) {

this.roleName = roleName;

}

@Override

public String getAuthority(){

return name();

}

}

27. main\java\com\example\demo\models\Status.java

package com.example.demo.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.Size;

@Entity

public class Status {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@NotEmpty(message = "Поле не может быть пустым")

@Size(min = 1, max = 20, message = "От 1 до 20 символов")

private String nameStatus;

public Status(Long id, String nameStatus) {

this.id = id;

this.nameStatus = nameStatus;

}

public Status(){}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getNameStatus() {

return nameStatus;

}

public void setNameStatus(String nameStatus) {

this.nameStatus = nameStatus;

}

}

28. main\java\com\example\demo\models\User.java

package com.example.demo.models;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.\*;

import java.util.Collection;

import java.util.Set;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String username;

private String password;

private boolean active;

@ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)

@CollectionTable(name = "user\_role",joinColumns = @JoinColumn(name="user\_id"))

@Enumerated(EnumType.STRING)

private Set<Role> roles;

public User(Long id, String username, String password, boolean active, Set<Role> roles) {

this.id = id;

this.username = username;

this.password = password;

this.active = active;

this.roles = roles;

}

public User(){}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

public Set<Role> getRoles() {

return roles;

}

public void setRoles(Set<Role> roles) {

this.roles = roles;

}

}

29. main\java\com\example\demo\repo\BookingRepository.java

package com.example.demo.repo;

import com.example.demo.models.Booking;

import com.example.demo.models.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookingRepository extends JpaRepository<Booking, Long> {

Iterable<Booking> findAllByUser(User user);

Booking findTopByTableNumberOrderByIdDesc(int tableNumber);

Booking findTopByUserOrderByIdDesc(User user);

Booking findFirstByOrderByIdDesc();

}

30. main\java\com\example\demo\repo\CartRepository.java

package com.example.demo.repo;

import com.example.demo.models.Booking;

import com.example.demo.models.Cart;

import com.example.demo.models.Product;

import org.springframework.data.jpa.repository.JpaRepository;

public interface CartRepository extends JpaRepository<Cart, Long> {

Iterable<Cart> findAllByBooking(Booking booking);

Cart findCartByBookingAndProduct(Booking booking, Product product);

}

31. main\java\com\example\demo\repo\CategoryRepository.java

package com.example.demo.repo;

import com.example.demo.models.Category;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface CategoryRepository extends JpaRepository<Category, Long> {

List<Category> findByNameCategoryContains(String nameCategory);

}

32. main\java\com\example\demo\repo\ImageRepository.java

package com.example.demo.repo;

import com.example.demo.models.Image;

import com.example.demo.models.Product;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ImageRepository extends JpaRepository<Image, Long> {

Image findImageByProduct(Product product);

}

33. main\java\com\example\demo\repo\ProductRepository.java

package com.example.demo.repo;

import com.example.demo.models.Product;

import org.springframework.data.domain.Page;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

import java.awt.print.Pageable;

import java.util.List;

import java.util.UUID;

public interface ProductRepository extends JpaRepository<Product, Long> {

List<Product> findByNameProductContains(String nameProduct);

Iterable<Product> findAllByCategoryNameCategoryContains(String nameCategory);

}

34. main\java\com\example\demo\repo\StatusRepository.java

package com.example.demo.repo;

import com.example.demo.models.Status;

import org.springframework.data.repository.CrudRepository;

public interface StatusRepository extends CrudRepository<Status, Long> {

Status findFirstByOrderByIdAsc();

}

35. main\java\com\example\demo\repo\UserRepos.java

package com.example.demo.repo;

import com.example.demo.models.User;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface UserRepos extends CrudRepository<User,Long> {

User findByUsername(String username);

List<User> findByActive(boolean active);

default List<User> findActive() {

return this.findByActive(true);

}

}

36. main\resources\application.properties

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/doubleF

spring.datasource.username=root

spring.datasource.password=

spring.main.allow-circular-references=true

server.error.whitelabel.enabled=false

server.port=8081

upload.path=/IdeaProjects/fastfood/src/main/resources/static/img

37. main\resources\static\jees.js

$(function(){

$(".fold-table tr.view").on("click", function(){

$(this).toggleClass("open").next(".fold").toggleClass("open");

});

});

38. main\resources\static\toggleCheckbox.scss

$toggle-indicator-size: 24px; // changing this number will resize the whole toggle

$track-height: $toggle-indicator-size + 6;

$track-width: $toggle-indicator-size \* 2.5;

$highContrastModeSupport: solid 2px transparent;

/\*

The following vars come from my theme.

You'll need to replace with your own color values.

- "$light"

- "$mid"

- "$dark"

\*/

$track-border: $mid;

$track-background: $light;

$focus-ring: 0px 0px 0px 2px $dark;

// Toggle specific styles

.toggle {

align-items: center;

border-radius: 100px;

display: flex;

font-weight: 700;

margin-bottom: 16px;

&:last-of-type {

margin: 0;

}

}

// Since we can't style the checkbox directly, we "hide" it so we can draw the toggle.

.toggle\_\_input {

clip: rect(0 0 0 0);

clip-path: inset(50%);

height: 1px;

overflow: hidden;

position: absolute;

white-space: nowrap;

width: 1px;

// This style sets the focus ring. The ":not([disabled])" prevents the focus ring from creating a flash when the element is clicked.

&:not([disabled]):active + .toggle-track,

&:not([disabled]):focus + .toggle-track {

border: 1px solid transparent;

box-shadow: $focus-ring;

}

&:disabled + .toggle-track {

cursor: not-allowed;

opacity: 0.7;

}

}

.toggle-track {

background: $track-background;

border: 1px solid $track-border;

border-radius: 100px;

cursor: pointer;

display: flex;

height: $track-height;

margin-right: 12px;

position: relative;

width: $track-width;

}

.toggle-indicator {

align-items: center;

background: $dark;

border-radius: $toggle-indicator-size;

bottom: 2px;

display: flex;

height: $toggle-indicator-size;

justify-content: center;

left: 2px;

outline: $highContrastModeSupport;

position: absolute;

transition: $speed;

width: $toggle-indicator-size;

}

// The check mark is optional

.checkMark {

fill: #fff;

height: $toggle-indicator-size - 4;

width: $toggle-indicator-size - 4;

opacity: 0;

transition: opacity $speed ease-in-out;

}

.toggle\_\_input:checked + .toggle-track .toggle-indicator {

background: $dark;

transform: translateX($track-width - $track-height);

.checkMark {

opacity: 1;

transition: opacity $speed ease-in-out;

}

}

@media screen and (-ms-high-contrast: active) {

.toggle-track {

border-radius: 0;

}

}

39. main\resources\static\wb.css

@charset "UTF-8";

@import url("https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css");

\* {

margin: 0;

padding: 0;

}

.pcs:after {

content: " pcs";

}

.cur:before {

content: "$";

}

.per:after {

content: "%";

}

table {

width: 100%;

}

table th {

text-align: center;

border-bottom: 1px solid #ccc;

}

table th,

table td {

padding: 0.4em;

}

table.fold-table>tbody>tr.view td,

table.fold-table>tbody>tr.view th {

cursor: pointer;

}

table.fold-table>tbody>tr.view td:first-child,

table.fold-table>tbody>tr.view th:first-child {

position: relative;

padding-left: 20px;

}

table.fold-table>tbody>tr.view td:first-child:before,

table.fold-table>tbody>tr.view th:first-child:before {

position: absolute;

top: 50%;

left: 5px;

width: 9px;

height: 16px;

margin-top: -8px;

font: 16px fontawesome;

color: #999;

content: "";

transition: all 0.3s ease;

}

table.fold-table>tbody>tr.view:nth-child(4n-1) {

background: #eee;

}

table.fold-table>tbody>tr.view:hover {

background: gray;

}

table.fold-table>tbody>tr.view.open {

background: gold;

color: white;

}

table.fold-table>tbody>tr.view.open td:first-child:before,

table.fold-table>tbody>tr.view.open th:first-child:before {

transform: rotate(-180deg);

color: #333;

}

table.fold-table>tbody>tr.fold {

display: none;

}

table.fold-table>tbody>tr.fold.open {

display: table-row;

}

.fold-content {

padding: 0.5em;

}

.fold-content h3 {

margin-top: 0;

}

.fold-content>table {

border: 2px solid #ccc;

}

.fold-content>table>tbody tr:nth-child(even) {

background: #eee;

}

40. main\resources\templates\accept-user\policy-terms.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<title>Условия политики конфиденциальности</title>

</head>

<body class="bg">

<div>

<div>

<h1>УСЛОВИЯХ ПОЛИТИКИ КОНФИДЕНЦИАЛЬНОСТИ</h1>

</div>

<div>

<h2>1.Общие положения и основные термины</h2>

<h4>

Настоящая политика обработки персональных данных составлена в соответствии с требованиями

Федерального закона от 27.07.2006. №152-ФЗ «О персональных данных» (далее – Закон о персональных данных)

и определяет порядок обработки персональных данных и меры по обеспечению безопасности персональных данных.<br>

Настоящая политика Оператора в отношении обработки персональных данных (далее – Политика) применяется ко

всей информации, которую Оператор может получить о посетителях данного веб-сайта.<br>

</h4>

</div>

<div>

<h2>2. ОПРЕДЕЛЕНИЯ ТЕРМИНОВ</h2>

<h4>

2.1. Автоматизированная обработка персональных данных – обработка персональных данных с помощью

средств вычислительной техники.<br>

2.2. Блокирование персональных данных – временное прекращение обработки персональных данных

(за исключением случаев, если обработка необходима для уточнения персональных данных).<br>

2.3. Веб-сайт – совокупность графических и информационных материалов, а также программ для ЭВМ и

баз данных, обеспечивающих их доступность в сети Интернет.<br>

2.4. Информационная система персональных данных — совокупность содержащихся в базах данных персональных

данных, и обеспечивающих их обработку информационных технологий и технических средств.<br>

2.5. Обезличивание персональных данных — действия, в результате которых невозможно определить без

использования дополнительной информации принадлежность персональных данных конкретному

Пользователю или иному субъекту персональных данных.<br>

2.6. Обработка персональных данных – любое действие (операция) или совокупность действий (операций),

совершаемых с использованием средств автоматизации или без использования таких средств с персональными

данными, включая сбор, запись, систематизацию, накопление, хранение, уточнение (обновление, изменение),

извлечение, использование, передачу (распространение, предоставление, доступ), обезличивание, блокирование,

удаление, уничтожение персональных данных.<br>

2.7. Оператор – государственный орган, муниципальный орган, юридическое или физическое лицо, самостоятельно

или совместно с другими лицами организующие и (или) осуществляющие обработку персональных данных, а также

определяющие цели обработки персональных данных, состав персональных данных, подлежащих обработке, действия

(операции), совершаемые с персональными данными.<br>

2.8. Персональные данные – любая информация, относящаяся прямо или косвенно к определенному или определяемому

Пользователю данного сайта.<br>

2.9. Персональные данные, разрешенные субъектом для распространения – персональные данные, доступ

неограниченного круга лиц к которым предоставлен субъектом персональных данных путем дачи согласия на

обработку персональных данных, разрешенных субъектом персональных данных для распространения в порядке,

предусмотренном Законом о персональных данных (далее - персональные данные, разрешенные для распространения).<br>

2.10. Пользователь – любой посетитель этого сайта.<br>

2.11. Предоставление персональных данных – действия, направленные на раскрытие персональных данных определенному

лицу или определенному кругу лиц.<br>

2.12. Распространение персональных данных – любые действия, направленные на раскрытие персональных данных

неопределенному кругу лиц (передача персональных данных) или на ознакомление с персональными данными неограниченного

круга лиц, в том числе обнародование персональных данных в средствах массовой информации, размещение в

информационно-телекоммуникационных сетях или предоставление доступа к персональным данным каким-либо иным способом.<br>

2.13. Трансграничная передача персональных данных – передача персональных данных на территорию иностранного государства

органу власти иностранного государства, иностранному физическому или иностранному юридическому лицу.<br>

2.14. Уничтожение персональных данных – любые действия, в результате которых персональные данные уничтожаются

безвозвратно с невозможностью дальнейшего восстановления содержания персональных данных в информационной системе

персональных данных и (или) уничтожаются материальные носители персональных данных.<br>

</h4>

</div>

<div>

<h2>3. Основные права и обязанности Оператора</h2>

<h4>3.1. Оператор имеет право:<br>

<h5>

3.1.1. Получать от субъекта персональных данных достоверные информацию и/или документы, содержащие персональные данные;<br>

3.1.2. В случае отзыва субъектом персональных данных согласия на обработку персональных данных Оператор вправе продолжить

обработку персональных данных без согласия субъекта персональных данных при наличии оснований, указанных в Законе

о персональных данных;<br>

3.1.3. Самостоятельно определять состав и перечень мер, необходимых и достаточных для обеспечения выполнения

обязанностей, предусмотренных Законом о персональных данных и принятыми в соответствии с ним нормативными правовыми

актами, если иное не предусмотрено Законом о персональных данных или другими федеральными законами.<br>

</h5>

</h4>

<h4>3.2. Оператор обязан:<br>

<h5>

3.2.1. Предоставлять субъекту персональных данных по его просьбе информацию, касающуюся обработки его персональных данных;<br>

3.2.2. Организовывать обработку персональных данных в порядке, установленном действующим законодательством РФ;<br>

3.2.3. Отвечать на обращения и запросы субъектов персональных данных и их законных представителей в соответствии с

требованиями Закона о персональных данных;<br>

3.2.4. Сообщать в уполномоченный орган по защите прав субъектов персональных данных по запросу этого органа необходимую

информацию в течение 30 дней с даты получения такого запроса;<br>

3.2.5. Публиковать или иным образом обеспечивать неограниченный доступ к настоящей Политике в отношении обработки

персональных данных;<br>

3.2.6. Принимать правовые, организационные и технические меры для защиты персональных данных от неправомерного

или случайного доступа к ним, уничтожения, изменения, блокирования, копирования, предоставления, распространения

персональных данных, а также от иных неправомерных действий в отношении персональных данных;<br>

3.2.7. Прекратить передачу (распространение, предоставление, доступ) персональных данных, прекратить обработку и

уничтожить персональные данные в порядке и случаях, предусмотренных Законом о персональных данных;<br>

3.2.8. Исполнять иные обязанности, предусмотренные Законом о персональных данных.<br>

</h5>

</h4>

</div>

</div>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

/\*color: #A7A1AE;\*/

color: #E6E4E4;

background-color: #1F2739;

}

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#auth {

margin-left: 30px;

}

}

h1 {

text-align: center;

}

h2{

margin-left: 50px;

}

h4{

margin-left: 80px;

}

h5{

margin-left: 100px;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 100px;

padding-right: 100px;

text-transform: uppercase;

-webkit-transition: all 1s ease;

transition: all 1s ease;

}

a:hover, button:hover {

color: black;

background-color: whitesmoke;

}

input {

margin-top: 15px;

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

41. main\resources\templates\accept-user\userAccept.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<title>Пользовательское соглашение</title>

</head>

<body class="bg">

<div>

<div>

<h1>ПОЛЬЗОВАТЕЛЬСКОЕ СОГЛАШЕНИЕ</h1>

</div>

<div>

<h2>1. ОБЩИЕ ПОЛОЖЕНИЯ</h2>

<h4>1.1. Настоящее Пользовательское соглашение (далее - Соглашение) относится к сайту «НАЗВАНИЕ САЙТА»,

расположенному по адресу АДРЕС САЙТА.

<br>

1.2. Сайт «НАЗВАНИЕ САЙТА» (далее - Сайт) является собственностью юридического лица ОРГАНИЗАЦИЯ (ОГРН:

НОМЕР, ИНН: НОМЕР, адрес регистрации: ЮР.АДРЕС)<br>

1.3. Настоящее Соглашение регулирует отношения между Администрацией сайта «НАЗВАНИЕ САЙТА» (далее -

Администрация сайта) и Пользователем данного Сайта.<br>

1.4. Администрация сайта оставляет за собой право в любое время изменять, добавлять или удалять пункты

настоящего Соглашения без уведомления Пользователя.<br>

1.5. Использование Сайта Пользователем означает принятие Соглашения и изменений, внесенных в настоящее

Соглашение.<br>

1.6. Пользователь несет персональную ответственность за проверку настоящего Соглашения на наличие изменений

в нем.<br>

</h4>

</div>

<div>

<h2>2. ОПРЕДЕЛЕНИЯ ТЕРМИНОВ</h2>

<h4>2.1. Перечисленные ниже термины имеют для целей настоящего Соглашения следующее значение: <br>

<h5>

2.1.1 «НАЗВАНИЕ САЙТА» - Интернет-ресурс, расположенный на доменном имени АДРЕС САЙТА, осуществляющий

свою деятельность посредством Интернет-ресурса и сопутствующих ему сервисов (далее - Сайт).<br>

2.1.2. «НАЗВАНИЕ САЙТА» - сайт, содержащий информацию о Товарах и/или Услугах и/или Иных ценностях для

пользователя, Продавце и/или Исполнителе услуг, позволяющий осуществить выбор, заказ и (или)

приобретение Товара, и/или получение услуги.<br>

2.1.3. Администрация сайта - уполномоченные сотрудники на управления Сайтом, действующие от имени

юридического лица, ОРГАНИЗАЦИЯ.<br>

2.1.4. Пользователь сайта (далее - Пользователь) - лицо, имеющее доступ к Сайту, посредством сети

Интернет и использующее Сайт.<br>

2.1.5. Содержание сайта (далее - Содержание) - охраняемые результаты интеллектуальной деятельности,

включая тексты литературных произведений, их названия, предисловия, <br>

аннотации, статьи, иллюстрации, обложки, музыкальные произведения с текстом или без текста, графические,

текстовые, фотографические, производные, составные и иные произведения, <br>

пользовательские интерфейсы, визуальные интерфейсы, названия товарных знаков, логотипы, программы для

ЭВМ, базы данных, а также дизайн, структура, выбор, координация, внешний вид,<br>

общий стиль и расположение данного Содержания, входящего в состав Сайта и другие объекты

интеллектуальной собственности все вместе и/или по отдельности, содержащиеся на сайте АДРЕС САЙТА<br>

<br>

</h5>

</h4>

</div>

<div>

<h2>3. ПРЕДМЕТ СОГЛАШЕНИЯ</h2>

<h4>3.1. Предметом настоящего Соглашения является предоставление Пользователю доступа к содержащимся на Сайте

Товарам и/или оказываемым услугам. <br>

<h5>

3.1.1. Сайт предоставляет Пользователю следующие виды услуг (сервисов): • предоставление Пользователю

возможности размещения сообщений, комментариев,<br>

рецензий Пользователей, выставления оценок контенту сайта; • ознакомление с товарами/услугами,

размещенными на Сайте; • выбор и заказ товаров/услуг для осуществления<br>

последующей покупки или оформления на данном Сайте.<br>

3.1.2. Под действие настоящего Соглашения подпадают все существующие (реально функционирующие) на данный

момент услуги (сервисы) Сайта, а также любые их последующие модификации<br>

и появляющиеся в дальнейшем дополнительные услуги (сервисы).<br>

3.2. Доступ к сайту предоставляется на бесплатной основе.<br>

3.3. Настоящее Соглашение является публичной офертой. Получая доступ к Сайту Пользователь считается

присоединившимся к настоящему Соглашению. 3.4. Использование материалов и сервисов<br>

Сайта регулируется нормами действующего законодательства Российской Федерации

</h5>

</h4>

</div>

<div>

<h2>4. ПРАВА И ОБЯЗАННОСТИ СТОРОН</h2>

<h4>4.1. Администрация сайта вправе: <br>

<h5>

4.1.1. Изменять правила пользования Сайтом, а также изменять содержание данного Сайта.

Изменения вступают в силу с момента публикации новой редакции Соглашения на Сайте.

</h5>

</h4>

<h4>4.2. Пользователь вправе: <br>

<h5>

4.2.1. Пользоваться всеми имеющимися на Сайте услугами, а также приобретать любые Товары и/или Услуги, предлагаемые на Сайте.<br>

4.2.2. Задавать любые вопросы, относящиеся к услугам сайта: • по телефону: ТЕЛЕФОН • по электронной почте: ПОЧТА • через Форму обратной связи, расположенную по адресу: АДРЕС САЙТА<br>

4.2.3. Пользоваться Сайтом исключительно в целях и порядке, предусмотренных Соглашением и не запрещенных законодательством Российской Федерации. <br>

4.2.5. Требовать от администрации скрытия любой информации о пользователе. <br>

4.2.6. Использовать информацию сайта в коммерческих целях без специального разрешения.<br>

</h5>

</h4>

<h4>4.3. Пользователь Сайта обязуется: <br>

<h5>

4.3.1. Предоставлять по запросу Администрации сайта дополнительную информацию, которая имеет непосредственное отношение к предоставляемым услугам данного Сайта.<br>

4.3.2. Соблюдать имущественные и неимущественные права авторов и иных правообладателей при использовании Сайта.<br>

4.3.3. Не предпринимать действий, которые могут рассматриваться как нарушающие нормальную работу Сайта.<br>

4.3.4. Не распространять с использованием Сайта любую конфиденциальную и охраняемую законодательством Российской Федерации информацию о физических либо юридических лицах.<br>

4.3.5. Избегать любых действий, в результате которых может быть нарушена конфиденциальность охраняемой законодательством Российской Федерации информации.<br>

4.3.6. Не использовать Сайт для распространения информации рекламного характера, иначе как с согласия Администрации сайта.<br>

4.3.7. Не использовать сервисы с целью:

<h6>

4.3.7.1. нарушения прав несовершеннолетних лиц и (или) причинение им вреда в любой форме.<br>

4.3.7.2. ущемления прав меньшинств.<br>

4.3.7.3. представления себя за другого человека или представителя организации и (или) сообщества без достаточных на то прав, в том числе за сотрудников данного сайта.<br>

4.3.7.4. введения в заблуждение относительно свойств и характеристик какого-либо Товара и/или услуги, размещенных на Сайте.<br>

4.3.7.5. некорректного сравнения Товара и/или Услуги, а также формирования негативного отношения к лицам, (не) пользующимся определенными Товарами и/или услугами, или осуждения таких лиц.<br>

4.3.7.6. загрузки контента, который является незаконным, нарушает любые права третьих лиц; пропагандирует насилие, жестокость, ненависть и (или) дискриминацию по расовому, национальному, половому, <br>

религиозному, социальному признакам; содержит недостоверные сведения и (или) оскорбления в адрес конкретных лиц, организаций, органов власти.<br>

4.3.7.7. побуждения к совершению противоправных действий, а также содействия лицам, действия которых направлены на нарушение ограничений и запретов, действующих на территории Российской Федерации.<br>

</h6>

<h5>

4.3.8. Обеспечить достоверность предоставляемой информации<br>

4.3.9. Обеспечивать сохранность личных данных от доступа третьих лиц.<br>

</h5>

</h5>

</h4>

<h4>4.4. Пользователю запрещается: <br>

<h5>

4.4.1. Использовать любые устройства, программы, процедуры, алгоритмы и методы, автоматические устройства или эквивалентные ручные процессы для доступа, приобретения, копирования или отслеживания содержания Сайта.<br>

4.4.2. Нарушать надлежащее функционирование Сайта.<br>

4.4.3. Любым способом обходить навигационную структуру Сайта для получения или попытки получения любой информации, документов или материалов любыми средствами, которые специально не представлены сервисами данного Сайта.<br>

4.4.4. Несанкционированный доступ к функциям Сайта, любым другим системам или сетям, относящимся к данному Сайту, а также к любым услугам, предлагаемым на Сайте.<br>

4.4.4. Нарушать систему безопасности или аутентификации на Сайте или в любой сети, относящейся к Сайту.<br>

4.4.5. Выполнять обратный поиск, отслеживать или пытаться отслеживать любую информацию о любом другом Пользователе Сайта.<br>

4.4.6. Использовать Сайт и его Содержание в любых целях, запрещенных законодательством Российской Федерации, а также подстрекать к любой незаконной деятельности или другой деятельности, нарушающей права Сайта или других лиц.<br>

</h5>

</h4>

</div>

</div>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

/\*color: #A7A1AE;\*/

color: #E6E4E4;

background-color: #1F2739;

}

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#auth {

margin-left: 30px;

}

}

h1 {

text-align: center;

}

h2 {

margin-left: 50px;

}

h4 {

margin-left: 80px;

}

h5 {

margin-left: 100px;

}

h6 {

margin-left: 120px;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 100px;

padding-right: 100px;

text-transform: uppercase;

-webkit-transition: all 1s ease;

transition: all 1s ease;

}

a:hover, button:hover {

color: black;

background-color: whitesmoke;

}

input {

margin-top: 15px;

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

42. main\resources\templates\admin\userEdit.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<title>Редактирование пользователя</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

</head>

<body th:align="center" class="bg">

<header>

<section class="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</link>

</div>

</div>

</section>

</header>

<div class="empty"></div>

<div align="center">

<form th:action="${'/admin'}" method="post" th:object="${user}">

<div class="text-field">

<input type="hidden" th:field="\*{id}">

<input type="hidden" th:field="\*{password}">

<input type="hidden" th:field="\*{active}">

<p class="text-field\_\_label">Номер телефона</p>

<input class="text-field\_\_input" type="text" th:field="\*{username}">

<div>

<p class="text-field\_\_label">Роль</p>

<td class="text-center">

<div class="btn-group">

<select th:field="\*{roles}" class="form-control">

<div th:each="role: ${rolesList}">

<option th:selected="${user.roles.contains(role)}" th:value="${role}" th:text="${role.roleName}"/>

</div>

</select>

</div>

</td>

</div>

<br>

<button type="submit">Применить</button>

</div>

</form>

</div>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 300;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

.header {

padding: 10px;

height: 60px;

background-color: rgba(128, 128, 128, 50%);

}

select {

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

.text-field {

padding: 10px;

width: 300px;

height: 250px;

background-color: #1F2739;

border-radius: 30px;

}

.shapka {

display: flex;

}

\* {

margin: 0;

padding: 0;

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.empty {

height: 60px;

}

.bg {

background-image: url("../../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

input {

margin-top: 15px;

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

43. main\resources\templates\admin\userList.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml" lang="ru">

<head>

<title>Список пользователей</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

</head>

<body bgcolor="6600CC" th:align="center" class="bg">

<header>

<section class="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</link>

</div>

</div>

</section>

</header>

<div class="empty"></div>

<section>

<table th:align="center" class="container">

<tr>

<th><h1 th:text="${'Номер телефона'}"></h1></th>

<th><h1 th:text="${'Роль'}"></h1></th>

<th><h1 th:text="${'Редактирование'}"></h1></th>

<th><h1 th:text="${'Удаление'}"></h1></th>

</tr>

<div th:each="el : ${users}">

<tr>

<td><p th:text="${el.username}"></p></td>

<td><p th:text="${el.roles[0].roleName}"></p></td>

<th>

<a th:href="'/admin/' + ${el.id} + '/edit'" >Редактировать</a>

</th>

<th>

<form th:action="@{/admin/delete}" method="post">

<button type="submit">Удалить</button>

<input type="hidden" th:value="${el.id}" name="user">

</form>

</th>

</tr>

</div>

</table>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 300;

line-height: 1.42em;

color:#A7A1AE;

background-color:#1F2739;

}

\* {

margin: 0;

padding: 0;

}

.empty{

height: 60px;

}

.header {

padding: 10px;

height: 60px;

background-color: rgba(128, 128, 128, 50%);

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

h1 {

font-size:3em;

font-weight: 300;

line-height:1em;

text-align: center;

color: #4DC3FA;

}

h2 {

font-size:1em;

font-weight: 300;

text-align: center;

display: block;

line-height:1em;

padding-bottom: 2em;

color: #FB667A;

}

h2 a {

font-weight: 700;

text-transform: uppercase;

color: #FB667A;

text-decoration: none;

}

.blue { color: #185875; }

.yellow { color: #FFF842; }

.container th h1 {

font-weight: bold;

font-size: 1em;

text-align: left;

color: #185875;

}

.container td {

font-weight: normal;

font-size: 1em;

-webkit-box-shadow: 0 2px 2px -2px #0E1119;

-moz-box-shadow: 0 2px 2px -2px #0E1119;

box-shadow: 0 2px 2px -2px #0E1119;

}

.container {

text-align: left;

overflow: hidden;

width: 50%;

margin: 0 auto;

display: table;

padding: 0 0 8em 0;

}

.container td, .container th {

padding-bottom: 1%;

padding-top: 1%;

padding-left:1%;

}

/\* Background-color of the odd rows \*/

.container tr:nth-child(odd) {

background-color: #323C50;

}

/\* Background-color of the even rows \*/

.container tr:nth-child(even) {

background-color: #2C3446;

}

.container th {

background-color: #1F2739;

}

.container td:first-child { color: #FB667A; }

.container tr:hover {

background-color: #464A52;

-webkit-box-shadow: 0 6px 6px -6px #0E1119;

-moz-box-shadow: 0 6px 6px -6px #0E1119;

box-shadow: 0 6px 6px -6px #0E1119;

}

.container td:hover {

background-color: #FFF842;

color: #403E10;

font-weight: bold;

box-shadow: #7F7C21 -1px 1px, #7F7C21 -2px 2px, #7F7C21 -3px 3px, #7F7C21 -4px 4px, #7F7C21 -5px 5px, #7F7C21 -6px 6px;

transform: translate3d(6px, -6px, 0);

transition-delay: 0s;

transition-duration: 0.4s;

transition-property: all;

transition-timing-function: line;

}

@media (max-width: 800px) {

.container td:nth-child(4),

.container th:nth-child(4) { display: none; }

}

/\*th {\*/

/\* padding: 10px;\*/

/\* border-style: solid;\*/

/\* border-color: limegreen;\*/

/\* background: rgba(0, 255, 0, 0.3);\*/

/\*}\*/

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.rest {

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

color: white;

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

44. main\resources\templates\category\category-add.html

<!DOCTYPE html>

<html lang="en" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Добавление</title>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="70" height="60"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/category">Категория</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<form th:action="${'/category/add'}" method="post" th:object="${category}">

<div class="back">

<label>Наименование категории</label>

<input required th:field="\*{nameCategory}" type="text" name="nameCategory">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameCategory')}" th:errors="\*{nameCategory}"></div>

<br><br>

<button type="submit">Добавить</button>

</div>

<br>

</form>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.bg {

background-image: url("../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.back {

margin:0 40% 0 40%;

width:17%;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

/\*width: 250px;\*/

height: 60px;

}

input {

width:100%;

margin-bottom: 20px;

height: 30px;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

45. main\resources\templates\category\category-edit.html

<!DOCTYPE html>

<html lang="en" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Добавление</title>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="70" height="60"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/category">Категория</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<form th:action="${'/category/'+category.id+'/edit'}" method="post" th:object="${category}">

<div class="back">

<label>Наименование</label>

<input required th:field="\*{nameCategory}" type="text" name="nameCategory">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameCategory')}" th:errors="\*{nameCategory}"></div>

<br><br>

<button type="submit">Добавить</button>

</div>

</form>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.bg {

background-image: url("../../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.back {

margin:0 40% 0 40%;

width:17%;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

/\*width: 250px;\*/

height: 60px;

}

input {

margin-bottom: 20px;

height: 30px;

width: 250px;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

46. main\resources\templates\category\category.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>

<title>Категория</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/category">Категория</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a href="/category/add">Добавить</a>

</div>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<table class="container">

<thead>

<tr>

<th><h1 th:text="${'Наименование'}"></h1></th>

<th><h1 th:text="${'Изменение'}"></h1></th>

<th><h1 th:text="${'Удаление'}"></h1></th>

</tr>

</thead>

<tbody>

<div th:each="el : ${categories}">

<tr>

<td>

<p th:text="${el.nameCategory}"></p>

</td>

<div class="left">

<th>

<a th:href=" '/category/' + ${el.id} + '/edit'">Редактировать</a>

</th>

<th>

<form th:action=" '/category/' + ${el.id} + '/remove'" method="post" id="delForm" onchange="checkButton()">

<button id="dell" name="dell" value="Delete" type="button" >Удалить</button>

</form>

</th>

</div>

</tr>

</div>

</tbody>

</table>

</section>

<div id="hidden" class="show-none">

<div class="warn-window">

<div class="warn-title">Предупреждение</div>

<span>Вы действительно хотите удалить?</span>

<div class="warn-act">

<input id="yes" type="button" name="yes" value="Да">

<input id="no" type="button" name="no" value="Нет">

</div>

</div>

</div>

<script>

function checkButton()

{

var hid = document.getElementById('hidden'); //Находим id всасывающегося окна

var yes = document.getElementById('yes'); // id кнопки ДА

var no = document.getElementById('no'); // id кнопки НЕТ

yes.onclick = function(e) //если будет нажата кнопка ДА

{

var frm = document.getElementById('delForm'); //Находим id формы

this.onclick = null;

hid.className = ''; // скрывается окно подтверждения

frm.submit(); //// отправка на сервер происходит

e.stopPropagation();

e.preventDefault();

};

no.onclick = function(e) // Если будет нажата кнопка НЕТ

{

var hid = document.getElementById('hidden'); //Находим id всасывающегося окна

this.onclick = null;

hid.className = ''; //скрывается окно подтверждения

// отправка на сервер не происходит

e.stopPropagation();

e.preventDefault();

}

// Oтображается окно подтверждения.

hid.className = 'substrate';

}

window.onload = function(e)

{

var myf = document.getElementById('dell');

myf.onclick = checkButton; //Подписка на событие клика по кнопке УДАЛИТЬ

e.stopPropagation();

e.preventDefault();

}

</script>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.left{

margin-left: 0;

}

.substrate{

z-index:1;

position: absolute;

background-color: #000;

opacity: 0.8;

filter: alpha(opacity=80);

width: 100%;

height: 100%;

top: 0;

left: 0;

display: block;

}

.show-none{

display:none;

}

/\*.warn-window{\*/

/\* z-index:2;\*/

/\* background: #fff;\*/

/\* height: 180px;\*/

/\* width: 300px;\*/

/\* border: 1px #3CCB34 solid;\*/

/\* border-radius:25px;\*/

/\* margin:200px auto;\*/

/\*}\*/

.warn-window{

margin-top: 450px;

margin-left: 620px;

padding: 10px;

border-radius: 25px;

width: 200px;

height: 120px;

background: #2C3446;

}

h1 {

font-size:3em;

font-weight: 300;

line-height:1em;

text-align: center;

color: #4DC3FA;

}

h2 {

font-size:1em;

font-weight: 300;

text-align: center;

display: block;

line-height:1em;

padding-bottom: 2em;

color: #FB667A;

}

h2 a {

font-weight: 700;

text-transform: uppercase;

color: #FB667A;

text-decoration: none;

}

.blue { color: #185875; }

.yellow { color: #FFF842; }

.container th h1 {

font-weight: bold;

font-size: 1em;

text-align: left;

color: #185875;

}

.container td {

font-weight: normal;

font-size: 1em;

-webkit-box-shadow: 0 2px 2px -2px #0E1119;

-moz-box-shadow: 0 2px 2px -2px #0E1119;

box-shadow: 0 2px 2px -2px #0E1119;

}

.container {

text-align: left;

overflow: hidden;

width: 40%;

margin: 0 auto;

display: table;

padding: 0 0 8em 0;

}

.container td, .container th {

padding-bottom: 1%;

padding-top: 1%;

padding-left:1%;

}

/\* Background-color of the odd rows \*/

.container tr:nth-child(odd) {

background-color: #323C50;

}

/\* Background-color of the even rows \*/

.container tr:nth-child(even) {

background-color: #2C3446;

}

.container th {

background-color: #1F2739;

}

.container td:first-child { color: #FB667A; }

.container tr:hover {

background-color: #464A52;

-webkit-box-shadow: 0 6px 6px -6px #0E1119;

-moz-box-shadow: 0 6px 6px -6px #0E1119;

box-shadow: 0 6px 6px -6px #0E1119;

}

.container td:hover {

background-color: #FFF842;

color: #403E10;

font-weight: bold;

box-shadow: #7F7C21 -1px 1px, #7F7C21 -2px 2px, #7F7C21 -3px 3px, #7F7C21 -4px 4px, #7F7C21 -5px 5px, #7F7C21 -6px 6px;

transform: translate3d(6px, -6px, 0);

transition-delay: 0s;

transition-duration: 0.4s;

transition-property: all;

transition-timing-function: line;

}

@media (max-width: 800px) {

.container td:nth-child(4),

.container th:nth-child(4) { display: none; }

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

input{

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

</style>

47. main\resources\templates\login.html

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:th="https://www.thymeleaf.org">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<script src="https://unpkg.com/imask"></script>

<link rel="stylesheet" type="text/css" href="toggleCheckbox.scss">

<title>Вход</title>

</head>

<body th:align="center" class="bg">

<!--<div class="con2">-->

<!-- <div th:if="${param.error}" class="error">-->

<!-- Неверный пароль или логин!-->

<!-- </div>-->

<!-- <br>-->

<!-- <div th:if="${param.logout}" class="logout">-->

<!-- Выход выполнен!-->

<!-- </div>-->

<!--</div>-->

<div class="con" id="auth">

<h1>Авторизация</h1>

<form th:action="@{/login}" method="post" id="form">

<div>

<label>Номер телефона</label>

<input required type="text" name="username" data-mask="phoneNumber"/><br>

</div>

<div>

<label class="password">Пароль</label><br>

<input type="password" name="password" class="form-control password1" placeholder=""/><br>

</div>

<div class="div-show-password">

<label class="label-show-password">Показать пароль</label>

<input type="checkbox" class="show-password">

</div>

<div>

<button type="submit" value="Войти">Войти</button>

<a class="button" href="/registration">Регистрация</a>

</div>

</form>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => {

const mask = (dataValue, options) => {

const elements = document.querySelectorAll(`[data-mask="${dataValue}"]`)

if (!elements) return

elements.forEach(el => {

IMask(el, options)

})

}

mask('phoneNumber', {

mask: '+{7}(000)000-00-00'

})

})

</script>

</body>

</html>

<script>

window.addEventListener("load", function() {

// иконка для отображения пароля

showPassword = document.querySelector('.show-password');

showPassword.addEventListener('click', () => {

// элементы ввода типа пароля

password1 = document.querySelector('.password1');

password2 = document.querySelector('.password2');

if ( password1.type === "text" ) {

password1.type = "password"

password2.type = "password"

} else {

password1.type = "text"

password2.type = "text"

}

})

});

</script>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#auth{

/\*margin-top: 200px;\*/

margin-left: 30px;

}

}

.div-show-password{

text-align: left;

margin: 10px 0 0 40px;

}

.label-show-password{

margin: 0;

}

h1 {

padding-top: 20px;

}

.show-password{

margin: 0;

width: 15px;

height: 15px;

}

.logout {

text-align: center;

vertical-align: center;

line-height: 70px;

font-size: 20px;

color: yellow;

width: 230px;

height: 70px;

background-color: black;

border-radius: 25px;

border: solid yellow 1px;

}

.error {

font-size: 20px;

color: red;

}

.con {

margin-top: 130px;

margin-left: 890px;

background-color: #1F2739;

width: 330px;

height: 420px;

border-radius: 20px;

}

.con2 {

margin-left: 1250px;

}

.bg {

background-image: url("backНАСАЙТ.jpg");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.button {

margin-left: 38px;

display: block;

width: 235px;

height: 30px;

background-color: #323C50;

padding: 15px 12px 10px 12px;

text-align: center;

border-radius: 15px;

color: white;

font-size: 15px;

line-height: 20px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-top: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 100px;

padding-right: 100px;

text-transform: uppercase;

-webkit-transition: all 1s ease;

transition: all 1s ease;

}

a:hover, button:hover {

color: black;

background-color: whitesmoke;

}

input {

margin-top: 15px;

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

48. main\resources\templates\orders\details.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<title>Покупка</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<meta name="viewport" content="width=device-width; initial-scale=1.0">

</head>

<header>

<section class="header" id="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</link>

</div>

<div th:each="el : ${booking}">

<a th:if="${cartCount != 0}" class="cart" id="cart" th:href=" '/orders/' + ${el.id} + '/edit'">

<img class="cart\_\_image" src="/cart.png" alt="Cart" width="45"

height="45"/>

<div class="cart\_\_num" id="cart\_num" th:text="${cartCount}"></div>

</a>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<body class="bg" >

<section>

<div class="flex-box">

<form th:action="@{/orders/return}" method="post">

<input value="Все блюда" type="submit" class="href-dish">

<input th:value="${booking.id}" name="booking" type="hidden">

</form>

<label class="href-dish-bird">></label>

<input th:value="${product.nameProduct}" type="button" class="href-dish">

</div>

<div class="element">

<div class="flex-box">

<img class="image" style="width: 300px;height: 300px;" th:if="${product.image!=null}"

th:src="'/image/' + ${product.image.idImage}" alt="">

<div class="block">

<h3 th:text="${product.nameProduct}"></h3>

<h3 th:text="${product.price+ ' руб.'}"></h3>

<h3 th:text="${product.weight+ ' г.'}"></h3>

</div>

</div>

<h3 th:text="${product.details}"></h3>

<form th:action="@{/cart/add}" method="post" th:object="${booking}">

<input type="hidden" th:field="\*{id}">

<input type="hidden" th:field="\*{tableNumber}">

<input type="hidden" th:field="\*{status}">

<input id="button" value="В корзину" type="submit" class="buttons-elements">

<input th:value="${product.id}" name="product" type="hidden">

</form>

</div>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#header {

width: 100%;

}

#button{

height: 20px;

}

#navbar-sub-list {

width: 400px;

}

}

\* {

margin: 0;

padding: 0;

}

.href-dish-bird{

margin-top: 15px;

}

.href-dish{

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

background: none;

text-decoration: none;

color: white;

font-size: 15px;

margin: 0 0 0 2px;

padding: 15px 15px 15px 15px;

}

.href-dish:hover{

color: #4DC3FA;

background: none;

}

.flex-box{

display: inline-flex;

}

.block{

display: block;

}

.buttons-elements{

margin-top: 10px;

}

#inner {

display: table;

margin: 0 auto;

}

#outer {

width: 100%

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

/\*color: #A7A1AE;\*/

color: lightyellow;

background-color: #1F2739;

}

image:hover {

transform: scale(1.2);

}

.element {

padding: 20px;

margin: 50px 0 0 420px;

width: 480px;

height: 460px;

background-color: rgb(31, 39, 57, 70%);

border-radius: 20px;

}

.empty {

padding-bottom: 20px;

}

.header {

position: fixed;

padding: 10px;

width: 100%;

height: 60px;

background-color: rgba(128, 128, 128, 100%);

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.empty {

height: 60px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.ErrorMassage {

color: red;

}

.bg {

background-image: url("../../backНАСАЙТ.jpg");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #1F2739;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: white;

background-color: #2C3446;

}

input {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #1F2739;

margin: 0 0 0 15px;

padding: 15px 30px 15px 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

input:hover {

color: white;

background-color: #2C3446;

}

input::placeholder {color: white;}

.cart {

width: 30px;

height: 30px;

border-radius: 50%;

background-color: #364364;

transition: 0.1s;

cursor: pointer;

position: fixed; /\* Фиксированное расположение \*/

top: 10px; /\* в правом верхнем углу \*/

right: 20px;

display: flex;

justify-content: center;

align-items: center;

object-fit: contain;

padding: 15px;

box-shadow: 1px 2px 4px rgba(0, 0, 0, 0.1);

}

/\* Увеличиваем кнопку при наведении на нее \*/

.cart:hover {

transform: scale(1.1);

}

/\* Стилизуем счетчик товаров \*/

.cart\_\_num {

position: absolute;

background-color: #d62240;

width: 30px;

height: 30px;

border-radius: 50%;

display: flex;

justify-content: center;

align-items: center;

color: #fff;

font-size: 18px;

font-weight: 500;

top: -5px;

right: -5px;

}

</style>

49. main\resources\templates\orders\orders-add.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<title>Покупка</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<meta name="viewport" content="width=device-width; initial-scale=1.0">

</head>

<body th:align="center" class="bg">

<header>

<section class="header" id="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/withdrawals">Заказы</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</link>

</div>

<div>

<form th:action="@{/orders/search}" method="get">

<!-- <img src="/zoom.png" alt="" style="width: 20px;height: 20px;">-->

<input type="text" name="search" placeholder="Поиск блюд">

<button type="submit">Найти</button>

<input type="hidden" name="booking" th:value="${booking.id}">

</form>

</div>

<div th:each="el : ${booking}">

<a th:if="${cartCount != 0}" class="cart" id="cart" th:href=" '/orders/' + ${el.id} + '/edit'">

<img class="cart\_\_image" src="/cart.png" alt="Cart" width="45"

height="45"/>

<div class="cart\_\_num" id="cart\_num" th:text="${cartCount}"></div>

</a>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<div th:each="category : ${categories}" th:if="${productCount != 0}">

<h2 th:text="${category.nameCategory}"></h2>

<div th:each="product : ${products}" th:if="${product.category==category && product.hiddenDish == false}" class="flex-box">

<div class="element">

<img class="image" style="width: 150px;height: 150px;" th:if="${product.image!=null}"

th:src="'/image/' + ${product.image.idImage}" alt="">

<h3 th:text="${product.nameProduct}"></h3>

<div class="ErrorMassage" th:if="${ErrorMassage != null}" th:text="${ErrorMassage}"></div>

<span><h3 th:text="${product.price+ ' руб.'}"></h3></span>

<span><h3 th:text="${product.weight+ ' г.'}"></h3></span>

<form th:action="@{/cart/add}" method="post" th:object="${booking}">

<input type="hidden" th:field="\*{id}">

<input type="hidden" th:field="\*{tableNumber}">

<input type="hidden" th:field="\*{status}">

<input id="button" value="В корзину" type="submit" class="buttons-elements">

<input th:value="${product.id}" name="product" type="hidden">

</form>

<form th:action="@{/orders/details}" method="post">

<input value="Подробнее" type="submit" class="buttons-elements">

<input th:value="${product.id}" type="hidden" name="product">

<input th:value="${booking.id}" type="hidden" name="booking">

</form>

</div>

</div>

</div>

<div th:if="${productCount == 0}">

<h2>

Такого блюда или категории пока нет, но мы их обязательно добавим!

</h2>

</div>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#header {

width: 100%;

}

#button{

height: 20px;

}

#navbar-sub-list {

width: 400px;

}

}

\* {

margin: 0;

padding: 0;

}

.flex-box{

display: inline-flex;

}

.buttons-elements{

margin-top: 10px;

}

#inner {

display: table;

margin: 0 auto;

}

#outer {

width: 100%

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

/\*color: #A7A1AE;\*/

color: lightyellow;

background-color: #1F2739;

}

image:hover {

transform: scale(1.2);

}

.element {

margin: 10px;

width: 180px;

height: 360px;

background-color: rgb(31, 39, 57, 70%);

border-radius: 20px;

}

.empty {

padding-bottom: 50px;

}

.header {

position: fixed;

padding: 10px;

width: 100%;

height: 60px;

background-color: rgba(128, 128, 128, 100%);

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.empty {

height: 60px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.ErrorMassage {

color: red;

}

.bg {

background-image: url("../../backНАСАЙТ.jpg");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #1F2739;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: white;

background-color: #2C3446;

}

input {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #1F2739;

margin: 0 0 0 15px;

padding: 15px 30px 15px 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

input:hover {

color: white;

background-color: #2C3446;

}

input::placeholder {color: white;}

.cart {

width: 30px;

height: 30px;

border-radius: 50%;

background-color: #364364;

transition: 0.1s;

cursor: pointer;

position: fixed; /\* Фиксированное расположение \*/

top: 10px; /\* в правом верхнем углу \*/

right: 20px;

display: flex;

justify-content: center;

align-items: center;

object-fit: contain;

padding: 15px;

box-shadow: 1px 2px 4px rgba(0, 0, 0, 0.1);

}

/\* Увеличиваем кнопку при наведении на нее \*/

.cart:hover {

transform: scale(1.1);

}

/\* Стилизуем счетчик товаров \*/

.cart\_\_num {

position: absolute;

background-color: #d62240;

width: 30px;

height: 30px;

border-radius: 50%;

display: flex;

justify-content: center;

align-items: center;

color: #fff;

font-size: 18px;

font-weight: 500;

top: -5px;

right: -5px;

}

</style>

50. main\resources\templates\orders\orders-edit.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<meta charset="UTF-8">

<title>Корзина</title>

</head>

<body th:align="center" class="bg">

<h1>Корзина</h1>

<div class="table-wrapper">

<table class="fl-table" th:if="${cartCount != 0}">

<thead>

<tr>

<th>Изображение</th>

<th>Наименование</th>

<th>Цена</th>

<th>Вес</th>

<th>Количество</th>

</tr>

</thead>

<tbody>

<form th:action="${'/orders/'+booking.id+'/edit'}" method="post" th:object="${booking}" id="container2">

<button type="submit">Подтвердить</button>

<br><br>

<div th:each="cart : ${carts}">

<tr>

<td><img style="width: 75px;height: 75px;" th:if="${cart.product.image!=null}"

th:src="'/image/' + ${cart.product.image.idImage}" alt=""></td>

<td><h3 th:text="${cart.product.nameProduct}"></h3></td>

<div class="ErrorMassage" th:if="${ErrorMassage != null}" th:text="${ErrorMassage}"></div>

<td><h3 th:text="${cart.product.price+ ' руб.'}"></h3></td>

<td><h3 th:text="${cart.product.weight+ ' г.'}"></h3></td>

<td>

<form th:action="@{/cart/add}" method="post" th:object="${booking}">

<input type="hidden" th:field="\*{id}">

<input type="hidden" th:field="\*{tableNumber}">

<input type="hidden" th:field="\*{status}">

<input th:value="${cart.product.id}" name="product" type="hidden">

<div class="cart\_\_num" id="cart\_num" th:text="${cart.count+' шт.'}"></div>

</form>

</td>

<td>

<form th:action="@{/cart/remove1}" method="post">

<input type="hidden" name="cart" th:value="${cart.id}">

<input type="hidden" name="booking" th:value="${booking.id}">

<button type="submit" class="plus-minus">-</button>

</form>

</td>

<td>

<form th:action="@{/cart/plus1}" method="post">

<input type="hidden" name="cart" th:value="${cart.id}">

<input type="hidden" name="booking" th:value="${booking.id}">

<button type="submit" class="plus-minus">+</button>

</form>

</td>

<td>

<form th:action="@{/cart/delete}" method="post">

<input type="hidden" name="cart" th:value="${cart.id}">

<input type="hidden" name="booking" th:value="${booking.id}">

<button type="submit" class="delete">✕</button>

</form>

</td>

</tr>

</div>

</form>

<tr>

<td><h3>Итого</h3></td>

<td><h3 th:text="${fullPrice + ' р.'}"></h3></td>

</tr>

<tbody>

</table>

</div>

<div th:if="${cartCount == 0}">

<form th:action="@{/cart/returnToOrder}" method="get">

<input type="hidden" name="booking" th:value="${booking.id}">

<img src="/EmptyCartBag.svg" width="80" height="70">

<h4>В вашей корзине пока пусто</h4>

<h4>Тут появятся товары, которые вы закажете.</h4>

<button type="submit">В каталог</button>

</form>

</div>

</body>

</html>

<style>

\* {

box-sizing: border-box;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

}

.bg {

background-image: url("../../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

h1 {

text-align: center;

font-size: 25px;

text-transform: uppercase;

letter-spacing: 1px;

color: white;

padding: 30px 0;

}

h2 {

text-align: center;

font-size: 18px;

text-transform: uppercase;

letter-spacing: 1px;

color: white;

padding: 30px 0;

}

h4{

color: white;

line-height: 10px;

}

/\* Table Styles \*/

.delete:hover{

background-color: red;

}

.table-wrapper {

margin: 10px 70px 70px;

box-shadow: 0px 35px 50px rgba(0, 0, 0, 0.2);

}

.fl-table {

border-radius: 15px;

font-size: 12px;

font-weight: normal;

border: none;

border-collapse: collapse;

width: 100%;

max-width: 100%;

white-space: nowrap;

background-color: white;

}

.fl-table td, .fl-table th {

text-align: center;

padding: 8px;

}

.fl-table td {

border-right: 1px solid #f8f8f8;

font-size: 12px;

}

.fl-table thead th {

color: #ffffff;

background: #4FC3A1;

}

.fl-table thead th:nth-child(odd) {

color: #ffffff;

background: #324960;

}

.fl-table tr:nth-child(even) {

background: #F8F8F8;

}

/\* Responsive \*/

@media (max-width: 767px) {

.fl-table {

display: block;

width: 100%;

}

.plus-minus{

padding: 11px 20px 11px 20px;

background: #324960;

}

.delete{

background-color: red;

}

.table-wrapper:before {

display: block;

text-align: right;

font-size: 11px;

color: white;

padding: 0 0 10px;

}

.fl-table thead, .fl-table tbody, .fl-table thead th {

display: block;

}

.fl-table thead th:last-child {

border-bottom: none;

}

.fl-table thead {

float: left;

}

.fl-table tbody {

width: auto;

position: relative;

overflow-x: auto;

}

.fl-table td, .fl-table th {

padding: 10px .625em .625em .625em;

height: 60px;

vertical-align: middle;

box-sizing: border-box;

overflow-x: hidden;

overflow-y: auto;

width: 120px;

font-size: 13px;

text-overflow: ellipsis;

}

.fl-table thead th {

text-align: left;

border-bottom: 1px solid #f7f7f9;

}

.fl-table tbody tr {

display: table-cell;

}

.fl-table tbody tr:nth-child(odd) {

background: none;

}

.fl-table tr:nth-child(even) {

background: transparent;

}

.fl-table tr td:nth-child(odd) {

background: #F8F8F8;

border-right: 1px solid #E6E4E4;

}

.fl-table tr td:nth-child(even) {

border-right: 1px solid #E6E4E4;

}

.fl-table tbody td {

display: block;

text-align: center;

}

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

51. main\resources\templates\orders\orders.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>

<title>Главная</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<meta name="viewport" content="width=device-width; initial-scale=1.0">

</head>

<body th:align="center" class="bg">

<header>

<section class="header" id="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<div>

<a th:if="${isAdmin}" href="/admin">Пользователи</a>

</div>

<div>

<a th:if="${isSaler}" href="/products">Блюда</a>

</div>

<div>

<a th:if="${isSaler}" href="/category">Категория</a>

</div>

<div>

<a th:if="${isSaler}" href="/status">Статус</a>

</div>

<div >

<a th:if="${isUser}" href="/withdrawals">Заказы</a>

</div>

<div>

<a th:if="${isSaler}" href="/withdrawals">Все заказы</a>

</div>

</div>

<div class="exit">

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</link>

</div>

<!-- <div th:if="${isUser}" class="rest">-->

<!-- <div th:each="el : ${snackbars}">-->

<!-- <p th:text="${'Ресторан: ' +el.nameSnackbar}"></p>-->

<!-- <p th:text="${'Время работы: '+el.workingTime}"></p>-->

<!-- <p th:text="${'Адрес: '+el.address}"></p>-->

<!-- </div>-->

<!-- </div>-->

<div class="role">

<p th:if="${isSaler}">Сотрудник отдела Продаж</p>

</div>

<div class="role">

<p th:if="${isAdmin}">Администратор</p>

</div>

</div>

</section>

</header>

<section>

<!-- <div class="greeting" id="greeting">-->

<!-- <h1>Welcome!</h1>-->

<!-- <h3>Добро пожаловать! Разрабатывай и трапезничай с Электронное меню ООО "ЛинТех"</h3>-->

<!-- -->

<!-- <h3>Добро пожаловать в Электронное меню ООО "ЛинТех"</h3>-->

<!-- <h3>После усердной разработки или долгой починки робота всегда хочется перекусить.</h3>-->

<!-- <h3>Но это не проблема для нашего меню, где Вы можете выбрать любые блюда, </h3>-->

<!-- <h3>предложенные столовой, находясь за своим столиком и не отходя от него!</h3>-->

<!-- </div>-->

<h1 id="hello">Разрабатывай, <span class="underlined underline-clip">Изучай</span>

<br>& Обедай <span class="underlined underline-mask">Вместе с </span>

<br><span class="underlined underline-overflow">Электронным меню от ООО "ЛинТех"</span></h1>

</section>

<section>

<div class="stolik" >

<h2>Введите номер своего стола</h2>

<br>

<!-- <div th:each="freeNumberTable : ${freeNumberTables}" class="flex-errors">-->

<!-- <label th:text="${freeNumberTable}"></label>-->

<!-- </div>-->

<form th:action="${'/orders/add'}" method="post" th:object="${booking}">

<div class="flex-errors">

<input required th:field="\*{tableNumber}" type="number" min="1" max="100" id="tableNumber">

</div>

<div class="ErrorMassage" th:if="${ErrorMassage != null}" th:text="${ErrorMassage[0]}"></div>

<div class="ErrorMassage" th:if="${ErrorMassage != null}" th:text="${ErrorMassage[1]}"></div>

<input type="hidden" th:value="${booking.id}" name="booking">

<button type="submit">Продолжить</button>

</form>

</div>

</section>

<!--<section>-->

<!-- <footer>-->

<!-- <div th:each="el : ${snackbars}">-->

<!-- <p th:text="${'Время работы: '+el.workingTime}"></p>-->

<!-- <p th:text="${'Адрес: '+el.address}"></p>-->

<!-- </div>-->

<!-- </footer>-->

<!--</section>-->

<section>

<footer id="footer">

<h3>© 2023 ООО «ЛинТех»® - все права защищены. Mofelz Production</h3>

</footer>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#header{

width: 100%;

}

#footer{

height: 50px;

width: 100%;

}

#hello{

font-size: clamp(2rem, 5vmin, 2rem);

/\*font-size: 10px\*/

}

#tableNumber{

}

}

@media screen and (max-width: 1080px) {

html {

-webkit-text-size-adjust: none;

}

#header{

width: 100%;

}

#footer{

height: 50px;

width: 100%;

}

#hello{

font-size: clamp(2rem, 5vmin, 2rem);

/\*font-size: 10px\*/

}

#tableNumber{

}

}

body {

display: grid;

font-family: 'Open Sans', sans-serif;

font-weight: 700;

color: #A7A1AE;

background-color: #1F2739;

position: relative;

height: 100%;

width: 100%;

}

h1 {

font-size: clamp(2rem, 7vmin, 4rem);

color: hsl(0 0% 98%);

}

.flex-errors{

display: inline-flex;

}

.underlined {

position: relative;

}

.underline-mask:after {

content: '';

position: absolute;

top: 95%;

width: 150%;

aspect-ratio: 3 / 1;

left: 50%;

transform: translate(-50%, 0);

border-radius: 50%;

border: 6px solid hsl(280 80% 50%);

/\* Use a conic gradient mask to hide and show the bits you want \*/

--spread: 140deg;

--start: 290deg;

mask: conic-gradient(from var(--start), white 0 var(--spread), transparent var(--spread));

}

.underline-overflow {

display: inline-block;

/\* A little extra padding and overflow hidden \*/

overflow: hidden;

padding-bottom: clamp(1rem, 2vmin, 2rem);

}

.underline-overflow:after {

content: '';

position: absolute;

top: 100%;

height: 150%;

aspect-ratio: 2.5 / 1;

left: 50%;

transform: translate(-50%, -10%);

border-radius: 50%;

border: 6px solid hsl(10 80% 50%);

}

.underline-clip:after {

content: '';

position: absolute;

top: 95%;

width: 150%;

aspect-ratio: 3 / 1;

left: 50%;

transform: translate(-50%, 0);

border-radius: 50%;

border: 6px solid hsl(130 80% 50%);

/\* Use a clip-path to hide and show the bits you want \*/

clip-path: polygon(0 0, 50% 50%, 100% 0);

}

html{height: 100%;}

\* {

margin: 0;

padding: 0;

}

.greeting{

}

footer{

padding: 10px;

width: 100%;

background-color: rgba(128, 128, 128, 50%);

position: fixed;

left: 0;

bottom: 0;

height: 25px;

}

.header {

width: 100%;

padding: 10px;

height: 60px;

background-color: rgba(128, 128, 128, 50%);

}

.ErrorMassage {

color: red;

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.bg {

background-image: url("../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p {

margin-top: 15px;

font-size: 25px;

}

.stolik {

margin-top: 70px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #1F2739;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: white;

background-color: #2C3446;

}

input {

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

52. main\resources\templates\orders\withdrawals.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"

xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<!-- <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">-->

<title>Заказы</title>

<script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>

<link th:href="@{/wb.css}" rel="stylesheet">

</head>

<body th:align="center" class="bg">

<header>

<section class="header">

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div th:if="${isSaler}">-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div th:if="${isSaler}">-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a th:if="${isSaler}" href="/category">Категория</a>-->

<!-- </div>-->

<div th:if="${isSaler}">

<a class="rest" href="/withdrawals">Все заказы</a>

</div>

<div>

<a th:href="@{/report/excel}" th:if="${isSaler}">🡇 Отчет</a>

</div>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div th:if="${isSaler}">

<form th:action="@{/hiddenBoughtOrders}" method="post">

<label>Показать купленные заказы</label>

<input name="payment" th:checked="${payment}" type="checkbox" style="width: 20px;height: 20px;" onchange="this.form.submit()">

</form>

</div>

<div class="role" th:if="${isSaler}">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<table class="fold-table" th:if="${bookingCount != 0 || isSaler}">

<thead>

<tr>

<th th:if="${isSaler}">Логин пользователя</th>

<th>Номер заказа</th>

<th>Номер столика</th>

<th>Статус заказа</th>

<th th:if="${isSaler}">Удаление</th>

</tr>

</thead>

<tbody>

<div th:each="el : ${bookings}" th:if="${el.user.username == isAuth || isSaler && (el.payment == payment || payment == true)}">

<tr class="view">

<td th:if="${isSaler}"><p th:text="${el.user.username}"></p></td>

<td><p th:text="${el.id}"></p></td>

<td><p th:text="${el.tableNumber}"></p></td>

<td>

<form th:action="@{/orders/save}" method="post" th:if="${isSaler}">

<div th:if="${el.payment == false}">

<select class="form-select" name="status" onchange="this.form.submit()">

<div th:each="status: ${statuses}">

<option th:value="${status.id}" th:text="${status.nameStatus}"

th:selected="${status.id==el.status.id}"/>

</div>

</select>

</div>

<input type="hidden" th:value="${el.id}" name="booking">

</form>

<p th:text="${el.status.nameStatus}" th:if="${isUser || el.payment == true}"></p>

</td>

<td th:if="${isSaler}">

<form th:action=" '/orders/' + ${el.id} + '/remove'" method="post" >

<button type="submit" class="btn btn-del">Удалить</button>

</form>

</td>

</tr>

<tr class="fold">

<td colspan="7">

<div class="fold-content">

<table>

<thead>

<tr>

<th>Наименование</th>

<th>Категория</th>

<th>Вес</th>

<th>Цена</th>

<th>Количество</th>

<th>Итоговая цена</th>

</tr>

</thead>

<tbody>

<div th:each="cart : ${carts}" th:if="${cart.booking == el}">

<tr class="view">

<td><p th:text="${cart.product.nameProduct}"></p></td>

<td><p th:text="${cart.product.category.nameCategory}"></p></td>

<td><p th:text="${cart.product.weight + ' г.'}"></p></td>

<td><p th:text="${cart.product.price + ' р.'}"></p></td>

<td><p th:text="${cart.count + ' шт.'}"></p></td>

<td><p th:text="${cart.count \* cart.product.price + ' р.'}"></p></td>

</tr>

</div>

<tr>

<td colspan="4"></td>

<td><p></p>Итого</td>

<td><p th:text="${fullPrice[elStat.index] + ' р.'}"></p></td>

<td th:if="${el.status.nameStatus == 'Готов' && el.payment == false && isUser}">

<form th:action="@{/pay}" method="post">

<input type="hidden" name="booking" th:value="${el.id}">

<button type="submit">Оплатить</button>

</form>

</td>

</tr>

</tbody>

</table>

</div>

</td>

</tr>

</div>

</tbody>

</table>

<div th:if="${bookingCount == 0 && isUser}">

<h2>Вы еще не сделали ни одного заказа</h2>

<h2>Либо оплатили все новые заказы</h2>

<h2>Для создания нового заказа перейдите на главное меню</h2>

</div>

<script th:src="@{/jees.js}"></script>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin: 0;

padding: 0;

}

.header {

padding: 10px;

height: 60px;

background-color: rgba(128, 128, 128, 50%);

}

select {

font-size: 20px;

color: #0E1119;

/\*color: rgba(128, 128, 128, 50%);\*/

/\*background-color: #2C3446;\*/

background-color: #F8F8F8;

}

@media (min-resolution: 150dpi) {

td {

font-size: 14px;

}

}

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#header {

width: 100%;

}

#container {

font-size: 60%;

width: 100%;

}

#p {

margin-top: 15px;

font-size: 15px;

}

}

@media (max-width: 800px) {

.container td:nth-child(4),

.container th:nth-child(4) {

display: none;

}

}

.empty {

height: 50px;

}

body {

font-size: 15px;

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

.rest {

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

h1 {

font-size: 3em;

font-weight: 300;

line-height: 1em;

text-align: center;

color: #4DC3FA;

}

h2 {

font-size: 1em;

font-weight: 300;

text-align: center;

display: block;

line-height: 1em;

padding-bottom: 2em;

color: #FB667A;

}

h2 a {

font-weight: 700;

text-transform: uppercase;

color: #FB667A;

text-decoration: none;

}

.blue {

color: #185875;

}

.yellow {

color: #FFF842;

}

.container th h1 {

font-weight: bold;

font-size: 1em;

text-align: left;

color: #185875;

}

.container td {

font-weight: normal;

font-size: 0.5em;

-webkit-box-shadow: 0 2px 2px -2px #0E1119;

-moz-box-shadow: 0 2px 2px -2px #0E1119;

box-shadow: 0 2px 2px -2px #0E1119;

}

.container {

text-align: left;

overflow: hidden;

width: 60%;

margin: 0 auto;

display: table;

padding: 0 0 8em 0;

}

.container td, .container th {

padding-bottom: 1%;

padding-top: 1%;

padding-left: 1%;

}

/\* Background-color of the odd rows \*/

.container tr:nth-child(odd) {

background-color: #323C50;

}

/\* Background-color of the even rows \*/

.container tr:nth-child(even) {

background-color: #2C3446;

}

.container th {

background-color: #1F2739;

}

.container td:first-child {

color: #FB667A;

}

.container tr:hover {

background-color: #464A52;

-webkit-box-shadow: 0 6px 6px -6px #0E1119;

-moz-box-shadow: 0 6px 6px -6px #0E1119;

box-shadow: 0 6px 6px -6px #0E1119;

}

.container td:hover {

background-color: #FFF842;

color: #403E10;

font-weight: bold;

box-shadow: #7F7C21 -1px 1px, #7F7C21 -2px 2px, #7F7C21 -3px 3px, #7F7C21 -4px 4px, #7F7C21 -5px 5px, #7F7C21 -6px 6px;

transform: translate3d(6px, -6px, 0);

transition-delay: 0s;

transition-duration: 0.4s;

transition-property: all;

transition-timing-function: line;

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p {

margin-top: 15px;

font-size: 20px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

53. main\resources\templates\products\products-add.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Добавление</title>

</head>

<body th:align="center" class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<div>

<a class="rest" href="">Блюда</a>

</div>

<!-- <div>-->

<!-- <a href="/category">Категория</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<form th:action="${'/products/add'}" method="post" th:object="${products}" enctype="multipart/form-data">

<div class="back">

<div>

<label>Наиманование блюда</label>

<input required th:field="\*{nameProduct}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameProduct')}"

th:errors="\*{nameProduct}">

</div>

<label>Изображение</label>

<div class="form-group mt-2">

<input placeholder="Изображение" type="file" class="form-control" name="file"

accept="image/png, image/jpg, image/jpeg">

<div class="error-text-danger" th:if="errorMessageImage!=null" th:text="${errorMessageImage}"></div>

</div>

<label>Вес</label>

<br>

<input required th:field="\*{weight}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('weight')}"

th:errors="\*{weight}">

</div>

<br>

<label>Цена</label>

<br>

<input required th:field="\*{price}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('price')}" th:errors="\*{price}"></div>

<br>

<label>Категория</label>

<select th:field="\*{category}" class="border">

<div th:each="el : ${categories}">

<option th:value="${el.id}" th:text="${el.nameCategory}" style="background-color: darkblue;color: white;border-radius: 20px"/>

</div>

</select>

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('category')}" th:errors="\*{category}"></div>

<br>

<label>Подробная информация</label>

<br>

<input required th:field="\*{details}" type="text" class="text-for-detail-info">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('details')}" th:errors="\*{details}"></div>

</div>

</div>

<br>

<button type="submit">Добавить</button>

</form>

</body>

</html>

<script>

var uploadField = document.querySelector("input[name=file]");

uploadField.onchange = function() {

if(this.files[0].size > 1048576){

alert("Размер файла превышает 1МБ!");

this.value = "";

};

};

</script>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.text-for-detail-info{

height: 70px;

border-radius: 10px;

}

select {

height: 30px;

width: 250px;

border-radius: 30px;

/\*-webkit-appearance: button;\*/

/\*-moz-appearance: button;\*/

/\*-webkit-user-select: none;\*/

/\*-moz-user-select: none;\*/

/\*-webkit-padding-end: 20px;\*/

/\*-moz-padding-end: 20px;\*/

/\*-webkit-padding-start: 2px;\*/

/\*-moz-padding-start: 2px;\*/

/\*background-position: center right;\*/

/\*background-repeat: no-repeat;\*/

/\*border: 1px solid #AAA;\*/

/\*box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.1);\*/

/\*color: #555;\*/

/\*font-size: inherit;\*/

/\*margin: 0;\*/

/\*overflow: hidden;\*/

/\*padding-top: 2px;\*/

/\*padding-bottom: 2px;\*/

/\*text-overflow: ellipsis;\*/

/\*white-space: nowrap;\*/

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

.empty{

height: 100px;

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.rest {

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

background-image: url("../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.back {

margin-top: 50px;

margin-left: 600px;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

width: 250px;

height: 430px;

}

input {

margin-bottom: 20px;

height: 30px;

width: 250px;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

54. main\resources\templates\products\products-edit.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Редактирование</title>

</head>

<body th:align="center" class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<div>

<a class="rest" href="">Блюда</a>

</div>

<!-- <div>-->

<!-- <a href="/category">Категория</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a href="/products/add">Добавить</a>

</div>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<form th:action="${'/products/'+products.id+'/edit'}" method="post" th:object="${products}" enctype="multipart/form-data">

<div class="back">

<label>Наиманование продукции</label>

<input required th:field="\*{nameProduct}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameProduct')}"

th:errors="\*{nameProduct}">

</div>

<label>Изображение</label>

<div class="form-group mt-2">

<input placeholder="Изображение" type="file" class="form-control" name="file"

accept="image/png, image/jpg, image/jpeg">

<div class="error-text-danger" th:if="errorMessageImage!=null" th:text="${errorMessageImage}"></div>

</div>

<label>Вес</label>

<br>

<input required th:field="\*{weight}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('weight')}"

th:errors="\*{weight}">

</div>

<br>

<label>Цена</label>

<br>

<input required th:field="\*{price}" type="text">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('price')}"

th:errors="\*{price}">

</div>

<label>Категория</label>

<br>

<select th:field="\*{category}" class="border">

<div th:each="el : ${categories}">

<option th:value="${el.id}" th:text="${el.nameCategory}"/>

</div>

</select>

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('category')}" th:errors="\*{category}"></div>

<br>

<label>Подробная информация</label>

<br>

<input required th:field="\*{details}" type="text" class="text-for-detail-info">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('details')}" th:errors="\*{details}"></div>

</div>

<br>

<button type="submit">Применить</button>

</form>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

select {

height: 30px;

width: 250px;

border-radius: 30px;

/\*-webkit-appearance: button;\*/

/\*-moz-appearance: button;\*/

/\*-webkit-user-select: none;\*/

/\*-moz-user-select: none;\*/

/\*-webkit-padding-end: 20px;\*/

/\*-moz-padding-end: 20px;\*/

/\*-webkit-padding-start: 2px;\*/

/\*-moz-padding-start: 2px;\*/

/\*background-position: center right;\*/

/\*background-repeat: no-repeat;\*/

/\*border: 1px solid #AAA;\*/

/\*box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.1);\*/

/\*color: #555;\*/

/\*font-size: inherit;\*/

/\*margin: 0;\*/

/\*overflow: hidden;\*/

/\*padding-top: 2px;\*/

/\*padding-bottom: 2px;\*/

/\*text-overflow: ellipsis;\*/

/\*white-space: nowrap;\*/

}

.text-for-detail-info{

height: 70px;

border-radius: 10px;

}

.empty{

height: 100px;

}

.error-text-danger{

color: red;

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.rest {

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

background-image: url("../../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.back {

margin-top: 50px;

margin-left: 600px;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

width: 250px;

height: 430px;

}

input {

margin-bottom: 20px;

height: 30px;

width: 250px;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

55. main\resources\templates\products\products.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Продукция</title>

</head>

<body th:align="center" class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<div>

<a class="rest" href="">Блюда</a>

</div>

<!-- <div>-->

<!-- <a href="/category">Категория</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a href="/products/add">Добавить</a>

</div>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<table class="container">

<tr>

<th><h1 th:text="${'Скрыть блюдо'}"></h1></th>

<th><h1 th:text="${'Наименование'}"></h1></th>

<th><h1 th:text="${'Изображение'}"></h1></th>

<th><h1 th:text="${'Цена'}"></h1></th>

<th><h1 th:text="${'Вес'}"></h1></th>

<th><h1 th:text="${'Категория'}"></h1></th>

<th><h1 th:text="${'Редактирование'}"></h1></th>

<th><h1 th:text="${'Удаление'}"></h1></th>

</tr>

<div th:each="el : ${products}">

<tr>

<td>

<form th:action="@{/hiddenDish}" method="post">

<input th:checked="${el.hiddenDish}" type="checkbox" style="width: 20px;height: 20px;" onchange="this.form.submit()">

<input th:value="${el.id}" name="product" type="hidden">

</form>

</td>

<td><p th:text="${el.nameProduct}"></p></td>

<td><img style="width: 75px;height: 75px;" class="rounded img-fluid pt-4 align-center"

th:if="${el.image!=null}" th:src="'/image/' + ${el.image.idImage}" alt=""></td>

<td><p th:text="${el.price+ ' руб.'}"></p></td>

<td><p th:text="${el.weight+ ' г.'}"></p></td>

<td><p th:text="${el.category.nameCategory}"></p></td>

<th><a class="btn-edit" th:href=" '/products/' + ${el.id} + '/edit'">Редактировать</a></th>

<th>

<form th:action=" '/products/' + ${el.id} + '/remove'" method="post">

<button type="submit" class="btn-del">Удалить</button>

</form>

</th>

</tr>

</div>

</table>

<div class="buttons-inline-flex">

<form th:action="@{/productsSort}">

<div>

<button type="submit">Назад</button>

<input name="offset" th:value="${offset}" type="hidden">

</div>

</form>

<form th:action="@{/productsSort}">

<div>

<input name="offset" th:value="${offset+2}" type="hidden">

<button type="submit">Вперед</button>

</div>

</form>

</div>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.buttons-inline-flex{

display: inline-flex;

}

.back{

position: fixed;

width: 1460px;

height: 100px;

background: black;

}

.empty{

height: 100px;

}

h1 {

font-size:3em;

font-weight: 300;

line-height:1em;

text-align: center;

color: #4DC3FA;

}

h2 {

font-size:1em;

font-weight: 300;

text-align: center;

display: block;

line-height:1em;

padding-bottom: 2em;

color: #FB667A;

}

h2 a {

font-weight: 700;

text-transform: uppercase;

color: #FB667A;

text-decoration: none;

}

.blue { color: #185875; }

.yellow { color: #FFF842; }

.container th h1 {

font-weight: bold;

font-size: 1em;

text-align: left;

color: #185875;

}

.container td {

font-weight: normal;

font-size: 1em;

-webkit-box-shadow: 0 2px 2px -2px #0E1119;

-moz-box-shadow: 0 2px 2px -2px #0E1119;

box-shadow: 0 2px 2px -2px #0E1119;

}

.container {

text-align: left;

overflow: hidden;

width: 75%;

margin: 0 auto;

display: table;

padding: 0 0 1em 0;

}

.container td, .container th {

padding-bottom: 1%;

padding-top: 1%;

padding-left:1%;

}

/\* Background-color of the odd rows \*/

.container tr:nth-child(odd) {

background-color: #323C50;

}

/\* Background-color of the even rows \*/

.container tr:nth-child(even) {

background-color: #2C3446;

}

.container th {

background-color: #1F2739;

}

.container td:first-child { color: #FB667A; }

.container tr:hover {

background-color: #464A52;

-webkit-box-shadow: 0 6px 6px -6px #0E1119;

-moz-box-shadow: 0 6px 6px -6px #0E1119;

box-shadow: 0 6px 6px -6px #0E1119;

}

.container td:hover {

background-color: #FFF842;

color: #403E10;

font-weight: bold;

box-shadow: #7F7C21 -1px 1px, #7F7C21 -2px 2px, #7F7C21 -3px 3px, #7F7C21 -4px 4px, #7F7C21 -5px 5px, #7F7C21 -6px 6px;

transform: translate3d(6px, -6px, 0);

transition-delay: 0s;

transition-duration: 0.4s;

transition-property: all;

transition-timing-function: line;

}

@media (max-width: 800px) {

.container td:nth-child(4),

.container th:nth-child(4) { display: none; }

}

.imageMain {

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

background: none;

}

.rest {

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

/\*position: fixed;\*/

display: flex;

/\*padding-bottom: 100px;\*/

/\*margin-bottom: 100px;\*/

}

.shapka2 {

display: flex;

margin-top: 15px;

}

/\*th {\*/

/\* width: 150px;\*/

/\* border-style: solid;\*/

/\* border-color: limegreen;\*/

/\* !\*background-color: green;\*!\*/

/\* background: rgba(0, 255, 0, 0.3);\*/

/\*}\*/

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

56. main\resources\templates\registration.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta name="viewport" content="width=device-width; initial-scale=1.0">

<script src="https://unpkg.com/imask" ></script>

<title>Регистрация</title>

</head>

<body th:align="center" class="bg">

<form th:object="${user}" th:action="@{/registration}" method="post" class="con2">

<div th:if="${#fields.hasErrors('username')}" th:errors="\*{username}" ></div>

<div th:if="errorMessageUsername!=null"

th:text="${errorMessageUsername}" class="error"></div>

<br>

</form>

<div class="con" align="center" id="auth">

<div class="includediv">

<!-- <div th:if="${errorMessageUsername}">-->

<!-- <p th:text="${errorMessageUsername}"></p>-->

<!-- </div>-->

<h1>Регистрация</h1>

<form th:object="${user}" th:action="@{/registration}" method="post">

<div>

<label>Номер телефона</label><br>

<input required type="text" th:field="\*{username}" data-mask="phoneNumber"/>

</div>

<!-- <div class="error text-danger" th:if="${#fields.hasErrors('username')}" th:errors="\*{username}"></div>-->

<!-- <div style="color: red; font-size: 20px" class="error text-danger" th:if="errorMessageUsername!=null"-->

<!-- th:text="${errorMessageUsername}"></div>-->

<label>Пароль</label><br>

<div style="width: 250px; height: 40px; margin: 1px">

<input required type="password" th:field="\*{password}" name="password" class="form-control password1"/>

</div>

<div th:if="${#fields.hasErrors('password')}" th:errors="\*{password}"></div>

<div class="div-show-password">

<label class="label-show-password">Показать пароль</label>

<input type="checkbox" class="show-password">

</div>

<div>

<button type="submit">Далее</button>

<br><br>

<a href="/login">Назад</a>

</div>

<div class="accept">

<label>

<span class="UserEntryFlow-description">Нажимая «Далее», вы принимаете

<a class="imageMain" href="/userAccept" target="\_blank">пользовательское соглашение</a>

и соглашаетесь на обработку вашей персональной информации на

<a class="imageMain" href="/policy-terms" target="\_blank">

условиях политики конфиденциальности</a>

</span>

</label>

</div>

</form>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', () => {

const mask = (dataValue, options) => {

const elements = document.querySelectorAll(`[data-mask="${dataValue}"]`)

if (!elements) return

elements.forEach(el => {

IMask(el, options)

})

}

mask('phoneNumber', {

mask: '+{7}(000)000-00-00'

})

})

</script>

</body>

</html>

<script>

window.addEventListener("load", function() {

// иконка для отображения пароля

showPassword = document.querySelector('.show-password');

showPassword.addEventListener('click', () => {

// элементы ввода типа пароля

password1 = document.querySelector('.password1');

password2 = document.querySelector('.password2');

if ( password1.type === "text" ) {

password1.type = "password"

password2.type = "password"

} else {

password1.type = "text"

password2.type = "text"

}

})

});

</script>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

.accept{

line-height: 16px;

text-align: left;

margin: 20px 0 0 40px;

}

.imageMain {

font-size: 10px;

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover {

color: blue;

background: none;

}

.UserEntryFlow-description{

margin-top: 20px;

padding-top: 10px;

font-size: 13px;

}

@media screen and (max-width: 480px) {

html {

-webkit-text-size-adjust: none;

}

#auth{

margin-left: 30px;

}

}

.div-show-password{

text-align: left;

margin: 40px 0 10px 40px;

}

.label-show-password{

margin: 0;

}

h1 {

padding-top: 20px;

}

.show-password{

margin: 0;

width: 15px;

height: 15px;

}

.logout {

font-size: 20px;

color: yellow;

}

.error {

/\*text-align: center;\*/

/\*vertical-align: center;\*/

/\*line-height: 70px;\*/

font-size: 20px;

color: red;

/\*width: 330px;\*/

/\*height: 70px;\*/

/\*background-color: black;\*/

/\*border-radius: 25px;\*/

/\*border: solid red 1px;\*/

}

.con {

margin-top: 100px;

margin-left: 890px;

background-color: #1F2739;

width: 330px;

height: 510px;

border-radius: 20px;

}

.con2 {

margin-left: 1150px;

}

.bg {

background-image: url("backНАСАЙТ.jpg");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.button {

margin-top: 10px;

display: block;

width: 240px;

height: 30px;

background-color: #323C50;

padding: 10px;

text-align: center;

border-radius: 15px;

color: white;

font-size: 15px;

line-height: 20px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 100px;

padding-right: 100px;

text-transform: uppercase;

-webkit-transition: all 1s ease;

transition: all 1s ease;

}

a:hover, button:hover {

color: black;

background-color: whitesmoke;

}

input {

margin-top: 15px;

color: black;

width: 250px;

height: 50px;

border-radius: 15px;

border-width: 1px;

}

</style>

57. main\resources\templates\status\status-add.html

<!DOCTYPE html>

<html lang="en" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Добавление</title>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="70" height="60"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a class="rest" href="/category">Категория</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/status">Статус</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/status/add">Добавить</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<form th:action="${'/status/add'}" method="post" th:object="${status}">

<div class="back">

<label>Наименование статуса</label>

<input required th:field="\*{nameStatus}" type="text" name="nameStatus">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameStatus')}" th:errors="\*{nameStatus}"></div>

<br><br>

<button type="submit">Добавить</button>

</div>

</form>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.bg {

background-image: url("../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.back {

margin:0 40% 0 40%;

width:17%;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

/\*width: 250px;\*/

height: 60px;

}

input {

margin-bottom: 20px;

height: 30px;

width: 100%;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

58. main\resources\templates\status\status-edit.html

<!DOCTYPE html>

<html lang="en" xmlns:sec="http://www.w3.org/1999/xhtml">

<head>

<meta charset="UTF-8">

<title>Редактирование</title>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="70" height="60"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a class="rest" href="/category">Категория</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/status">Статус</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/status/add">Добавить</a>-->

<!-- </div>-->

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<div class="role">

<p>Сотрудник Отдела продаж</p>

</div>

</div>

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<form th:action="${'/status/'+status.id+'/edit'}" method="post" th:object="${status}">

<div class="back">

<label>Наименование ресторана</label>

<input required th:field="\*{nameStatus}" type="text" name="nameStatus">

<div style="width: 250px; height: 40px; margin-left: 600px" th:if="\*{#fields.hasErrors('nameStatus')}" th:errors="\*{nameStatus}"></div>

<br><br>

<button type="submit">Применить</button>

</div>

</form>

</section>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

.bg {

background-image: url("../../black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.back {

margin:0 40% 0 40%;

width:17%;

display: block;

font-weight: 900;

text-align: left;

border-radius: 30px;

padding: 20px;

background-color: #323C50;

/\*width: 250px;\*/

height: 60px;

}

input {

margin-bottom: 20px;

height: 30px;

width: 100%;

color: black;

border-style: none;

border-radius: 30px;

background-color: white;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

</style>

59. main\resources\templates\status\status.html

<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>

<title>Статус</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

</head>

<body class="bg">

<header>

<section>

<div class="shapka">

<div>

<a class="imageMain" href="/"><img src="/ava.png" width="80" height="70"></a>

</div>

<div class="shapka2">

<!-- <div>-->

<!-- <a href="/snackbar">Рестораны</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/products">Блюда</a>-->

<!-- </div>-->

<!-- <div>-->

<!-- <a href="/category">Категория</a>-->

<!-- </div>-->

<div>

<a class="rest" href="/status">Статус</a>

</div>

<!-- <div>-->

<!-- <a href="/withdrawals">Все заказы</a>-->

<!-- </div>-->

<div>

<a href="/status/add">Добавить</a>

</div>

</div>

<div>

<link sec:authorize="isAuthenticated()">

<form th:action="@{/logout}" method="post">

<button type="submit">Выйти</button>

</form>

</div>

<!-- <div class="role">-->

<!-- <p>Сотрудник Отдела продаж</p>-->

<!-- </div>-->

</div>

</section>

</header>

<section>

<div class="empty"></div>

</section>

<section>

<table class="container">

<thead>

<tr>

<th><h1 th:text="${'Наименование'}"></h1></th>

<th><h1 th:text="${'Изменение'}"></h1></th>

<th><h1 th:text="${'Удаление'}"></h1></th>

</tr>

</thead>

<tbody>

<div th:each="el : ${status}">

<tr>

<td>

<p th:text="${el.nameStatus}"></p>

</td>

<div class="left">

<th>

<a th:href=" '/status/' + ${el.id} + '/edit'">Редактировать</a>

</th>

<th>

<form th:action=" '/status/' + ${el.id} + '/remove'" method="post" id="delForm" onchange="checkButton()">

<button id="dell" name="dell" value="Delete" type="button" >Удалить</button>

</form>

</th>

</div>

</tr>

</div>

</tbody>

</table>

</section>

<div id="hidden" class="show-none">

<div class="warn-window">

<div class="warn-title">Предупреждение</div>

<span>Вы действительно хотите удалить?</span>

<div class="warn-act">

<input id="yes" type="button" name="yes" value="Да">

<input id="no" type="button" name="no" value="Нет">

</div>

</div>

</div>

<script>

function checkButton()

{

var hid = document.getElementById('hidden'); //Находим id всасывающегося окна

var yes = document.getElementById('yes'); // id кнопки ДА

var no = document.getElementById('no'); // id кнопки НЕТ

yes.onclick = function(e) //если будет нажата кнопка ДА

{

var frm = document.getElementById('delForm'); //Находим id формы

this.onclick = null;

hid.className = ''; // скрывается окно подтверждения

frm.submit(); //// отправка на сервер происходит

e.stopPropagation();

e.preventDefault();

};

no.onclick = function(e) // Если будет нажата кнопка НЕТ

{

var hid = document.getElementById('hidden'); //Находим id всасывающегося окна

this.onclick = null;

hid.className = ''; //скрывается окно подтверждения

// отправка на сервер не происходит

e.stopPropagation();

e.preventDefault();

}

// Oтображается окно подтверждения.

hid.className = 'substrate';

}

window.onload = function(e)

{

var myf = document.getElementById('dell');

myf.onclick = checkButton; //Подписка на событие клика по кнопке УДАЛИТЬ

e.stopPropagation();

e.preventDefault();

}

</script>

</body>

</html>

<style>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,700);

\* {

margin:0;

padding:0;

}

body {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

color: #A7A1AE;

background-color: #1F2739;

}

header{

margin: 0;

padding: 10px;

width: 100%;

background: rgba(78,87,84,100%);

height: 60px;

position: fixed;

}

.empty{

height: 100px;

}

.left{

margin-left: 0;

}

.substrate{

z-index:1;

position: absolute;

background-color: #000;

opacity: 0.8;

filter: alpha(opacity=80);

width: 100%;

height: 100%;

top: 0;

left: 0;

display: block;

}

.show-none{

display:none;

}

/\*.warn-window{\*/

/\* z-index:2;\*/

/\* background: #fff;\*/

/\* height: 180px;\*/

/\* width: 300px;\*/

/\* border: 1px #3CCB34 solid;\*/

/\* border-radius:25px;\*/

/\* margin:200px auto;\*/

/\*}\*/

.warn-window{

margin-top: 450px;

margin-left: 620px;

padding: 10px;

border-radius: 25px;

width: 200px;

height: 120px;

background: #2C3446;

}

h1 {

font-size:3em;

font-weight: 300;

line-height:1em;

text-align: center;

color: #4DC3FA;

}

h2 {

font-size:1em;

font-weight: 300;

text-align: center;

display: block;

line-height:1em;

padding-bottom: 2em;

color: #FB667A;

}

h2 a {

font-weight: 700;

text-transform: uppercase;

color: #FB667A;

text-decoration: none;

}

.blue { color: #185875; }

.yellow { color: #FFF842; }

.container th h1 {

font-weight: bold;

font-size: 1em;

text-align: left;

color: #185875;

}

.container td {

font-weight: normal;

font-size: 1em;

-webkit-box-shadow: 0 2px 2px -2px #0E1119;

-moz-box-shadow: 0 2px 2px -2px #0E1119;

box-shadow: 0 2px 2px -2px #0E1119;

}

.container {

text-align: left;

overflow: hidden;

width: 40%;

margin: 0 auto;

display: table;

padding: 0 0 8em 0;

}

.container td, .container th {

padding-bottom: 1%;

padding-top: 1%;

padding-left:1%;

}

/\* Background-color of the odd rows \*/

.container tr:nth-child(odd) {

background-color: #323C50;

}

/\* Background-color of the even rows \*/

.container tr:nth-child(even) {

background-color: #2C3446;

}

.container th {

background-color: #1F2739;

}

.container td:first-child { color: #FB667A; }

.container tr:hover {

background-color: #464A52;

-webkit-box-shadow: 0 6px 6px -6px #0E1119;

-moz-box-shadow: 0 6px 6px -6px #0E1119;

box-shadow: 0 6px 6px -6px #0E1119;

}

.container td:hover {

background-color: #FFF842;

color: #403E10;

font-weight: bold;

box-shadow: #7F7C21 -1px 1px, #7F7C21 -2px 2px, #7F7C21 -3px 3px, #7F7C21 -4px 4px, #7F7C21 -5px 5px, #7F7C21 -6px 6px;

transform: translate3d(6px, -6px, 0);

transition-delay: 0s;

transition-duration: 0.4s;

transition-property: all;

transition-timing-function: line;

}

@media (max-width: 800px) {

.container td:nth-child(4),

.container th:nth-child(4) { display: none; }

}

.imageMain{

border: none;

background: none;

margin: 0;

padding: 0;

}

.imageMain:hover{

background: none;

}

.rest{

pointer-events: none; /\* делаем ссылку некликабельной \*/

cursor: default; /\* устанавливаем курсор в виде стрелки \*/

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

-o-user-select: none;

user-select: none;

background-color: green;

}

.bg {

background-image: url("black2.png");

background-repeat: no-repeat;

background-position: center center;

background-attachment: fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

p{

margin-top: 15px;

font-size: 25px;

}

.role {

display: flex;

margin-left: 15px;

}

.shapka {

display: flex;

}

.shapka2 {

display: flex;

margin-top: 15px;

}

a, button {

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

a:hover, button:hover {

color: black;

background-color: #FFF842;

}

input{

font-family: 'Open Sans', sans-serif;

font-weight: 700;

line-height: 1.42em;

text-decoration: none;

color: white;

font-size: 15px;

border-width: 0px;

border-style: solid;

border-radius: 15px;

background-color: #323C50;

margin-left: 10px;

padding-top: 15px;

padding-bottom: 15px;

padding-left: 30px;

padding-right: 30px;

text-transform: uppercase;

-webkit-transition: all 0.5s ease;

transition: all 0.5s ease;

}

</style>

60. test\java\com\example\demo\DemoApplicationTests.java

package com.example.demo;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class DemoApplicationTests {

@Test

void contextLoads() {

}

}