**ПРИЛОЖЕНИЕ А**

***(обязательное)***

**Листинг кода**

Файл JdbcColorDao.java

package by.bsuir.phoneshop.core.dao.impl;

import java.util.Optional;

import javax.annotation.Resource;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Component;

import by.bsuir.phoneshop.core.dao.ColorDAO;

import by.bsuir.phoneshop.core.models.Color;

@Component

public class JdbcColorDAO implements ColorDAO

{

@Resource

private JdbcTemplate jdbcTemplate;

private static final String SQL\_GET\_COLOR = "select \* from colors where id = ";

private static final String SQL\_UPDATE\_COLOR = "insert into colors(id,code) values (?,?)";

@Override

public Optional<Color> get(final Long key)

{

return Optional.ofNullable(jdbcTemplate.queryForObject(SQL\_GET\_COLOR + key,

new BeanPropertyRowMapper<Color>(Color.class)));

}

@Override

public void save(final Color color)

{

jdbcTemplate.update(SQL\_UPDATE\_COLOR, color.getId(), color.getCode());

}

}

Файл JdbcOrderDao.java

package by.bsuir.phoneshop.core.dao.impl;

import java.util.List;

import java.util.Optional;

import javax.annotation.Resource;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;

import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;

import org.springframework.stereotype.Component;

import by.bsuir.phoneshop.core.dao.OrderDao;

import by.bsuir.phoneshop.core.models.Order;

import by.bsuir.phoneshop.core.models.OrderItem;

import by.bsuir.phoneshop.core.dao.extractors.OrderListResultSetExtractor;

import by.bsuir.phoneshop.core.dao.extractors.OrderResultSetExtractor;

import by.bsuir.phoneshop.core.models.enums.OrderStatus;

@Component

public class JdbcOrderDao implements OrderDao

{

public static final String SELECT\_ORDER\_SQL = "select \* from orders left join orderItems on orders.id = orderItems.orderId where orders.id= ";

public static final String INSERT\_ORDER\_SQL = " insert into orders(id,subtotal,deliveryPrice,totalPrice,firstName," +

"lastName,deliveryAddress,contactPhoneNo,additionalInfo,status,date) values (:id,:subtotal,:deliveryPrice," +

":totalPrice,:firstName,:lastName,:deliveryAddress,:contactPhoneNo,:additionalInfo,:status,:date)";

public static final String SQL\_INSERT\_INTO\_ORDER\_ITEMS = "insert into orderItems (phoneId,orderId,quantity) " +

"values (?,?,?)";

public static final String SELECT\_ORDERS = "select \* from orders left join orderItems on orders.id = orderItems.orderId";

public static final String UPDATE\_ORDER\_STATUS\_SQL = "update orders set orders.status = '";

public static final String SQL\_NEEDED\_ORDER = "' where orders.id = ";

@Resource

private NamedParameterJdbcTemplate namedParameterJdbcTemplate;

@Resource

private JdbcTemplate jdbcTemplate;

@Resource

private OrderResultSetExtractor orderResultSetExtractor;

@Resource

private OrderListResultSetExtractor orderListResultSetExtractor;

@Override

public Optional<Order> get(final Long key)

{

String query = SELECT\_ORDER\_SQL + key;

return Optional.ofNullable(jdbcTemplate.query(query, orderResultSetExtractor));

}

@Override

public Long save(final Order order)

{

MapSqlParameterSource in = new MapSqlParameterSource();

in.addValue("id", order.getId());

in.addValue("subtotal", order.getSubtotal());

in.addValue("deliveryPrice", order.getDeliveryPrice());

in.addValue("totalPrice", order.getTotalPrice());

in.addValue("firstName", order.getFirstName());

in.addValue("lastName", order.getLastName());

in.addValue("deliveryAddress", order.getDeliveryAddress());

in.addValue("contactPhoneNo", order.getContactPhoneNo());

in.addValue("additionalInfo", order.getAdditionalInfo());

in.addValue("status", order.getStatus().name());

in.addValue("date", order.getDate());

namedParameterJdbcTemplate.update(INSERT\_ORDER\_SQL, in);

List<OrderItem> orderItemList = order.getOrderItems();

orderItemList.forEach(orderItem -> jdbcTemplate

.update(SQL\_INSERT\_INTO\_ORDER\_ITEMS, orderItem.getPhone().getId(), order.getId(), orderItem.getQuantity()));

return order.getId();

}

@Override

public List<Order> getOrders(final int limit, final int offset)

{

String query = SELECT\_ORDERS + " order by orders.date DESC " + " limit " + limit + " offset " + offset;

return jdbcTemplate.query(query, orderListResultSetExtractor);

}

@Override

public void updateStatus(OrderStatus orderStatus, Long key)

{

String query = UPDATE\_ORDER\_STATUS\_SQL + orderStatus.name() + SQL\_NEEDED\_ORDER + key;

jdbcTemplate.update(query);

}

}

Файл JdbcPhoneDao.java

package by.bsuir.phoneshop.core.dao.impl;

import java.sql.Types;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Optional;

import java.util.Set;

import javax.annotation.Resource;

import org.springframework.dao.EmptyResultDataAccessException;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;

import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;

import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;

import org.springframework.stereotype.Component;

import org.springframework.transaction.annotation.Transactional;

import by.bsuir.phoneshop.core.models.Color;

import by.bsuir.phoneshop.core.models.ParamsForSearch;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.dao.ColorDAO;

import by.bsuir.phoneshop.core.dao.PhoneDao;

import by.bsuir.phoneshop.core.dao.extractors.PhoneResultSetExtractor;

@Component

public class JdbcPhoneDao implements PhoneDao

{

private static final String SQL\_INSERT\_PHONE = "insert into phones (id,brand,model,price,discountPercent,displaySizeInches,weightGr," +

"lengthMm,widthMm,heightMm,deviceType,displayResolution,pixelDensity,displayTechnology," +

"backCameraMegapixels,frontCameraMegapixels,internalStorageGb,batteryCapacityMah,talkTimeHours," +

"standByTimeHours,bluetooth,imageUrl,description) values (:id,:brand,:model,:price,:discountPercent,:displaySizeInches,:weightGr," +

":lengthMm,:widthMm,:heightMm,:deviceType,:displayResolution,:pixelDensity," +

":displayTechnology,:backCameraMegapixels,:frontCameraMegapixels,:internalStorageGb," +

":batteryCapacityMah,:talkTimeHours,:standByTimeHours,:bluetooth,:imageUrl,:description)";

private static final String SQL\_GET\_ALL\_PHONES = "select \* from phones left join phone2color " +

"on phones.id=phone2color.phoneId left join colors on colors.id = phone2color.colorId ";

private static final String SQL\_GET\_MAX\_DISCOUNTED\_PHONES = "select \* from phones left join phone2color " +

"on phones.id=phone2color.phoneId left join colors on colors.id = phone2color.colorId where phones.discountPercent!=0 ORDER BY phones.discountPercent DESC";

public static final String SQL\_GET\_PHONE = "select \* from phones where id= ";

private static final String SQL\_SELECT\_COUNT\_FIND\_ALL\_EXTENDED = "select count(\*) from phones ";

private static final String SQL\_WHERE\_SEARCH = "where (phones.id in (select phoneId from stocks) and " +

"(SELECT STOCK FROM STOCKS WHERE PHONEID = PHONES.ID AND STOCK > 0) and price is not null) ";

public static final String SQL\_SELECT\_COLOR\_IDS = "select colorId from phone2color where phoneId= ";

public static final String SQL\_GET\_PHONE\_BY\_MODEL = "select \* from phones where phones.model = '";

public static final String DELETE\_PHONE\_SQL = "delete from phones where phones.id=?";

public static final String SQL\_UPDATE\_PHONE = "update phones set brand= :brand,model= :model,price= :price,discountPercent= :discountPercent," +

"displaySizeInches= :displaySizeInches,weightGr= :weightGr,lengthMm= :lengthMm,widthMm= :widthMm,heightMm= :heightMm,deviceType= :deviceType,displayResolution= :displayResolution,pixelDensity= :pixelDensity," +

"displayTechnology= :displayTechnology,backCameraMegapixels= :backCameraMegapixels,frontCameraMegapixels= :frontCameraMegapixels,internalStorageGb= :internalStorageGb,batteryCapacityMah= :batteryCapacityMah," +

"talkTimeHours= :talkTimeHours,standByTimeHours= :standByTimeHours,bluetooth= :bluetooth,imageUrl= :imageUrl,description= :description where id= :id";

@Resource

private JdbcTemplate jdbcTemplate;

@Resource

private NamedParameterJdbcTemplate namedParameterJdbcTemplate;

@Resource

private PhoneResultSetExtractor phoneResultSetExtractor;

@Resource

private ColorDAO jdbcColorDAO;

@Override

public Optional<Phone> get(final Long key)

{

final String query = SQL\_GET\_PHONE + key;

return getPhoneAndColors(query);

}

@Override

public Optional<Phone> get(final String model)

{

final String query = SQL\_GET\_PHONE\_BY\_MODEL + model + "'";

return getPhoneAndColors(query);

}

private Optional<Phone> getPhoneAndColors(final String query)

{

Phone phone;

try

{

phone = jdbcTemplate.queryForObject(query, new BeanPropertyRowMapper<>(Phone.class));

phone.setActualPrice();

}

catch (EmptyResultDataAccessException e)

{

return Optional.empty();

}

final RowMapper<Long> idRowMapper = (rs, rowNum) -> rs.getLong("colorId");

final String colorIdQuery = SQL\_SELECT\_COLOR\_IDS + phone.getId();

final List<Long> colorIds = jdbcTemplate.query(colorIdQuery, idRowMapper);

if (colorIds != null && !colorIds.isEmpty())

{

final Set<Color> colorSet = new HashSet<>();

for (Long colorId : colorIds)

{

final Optional<Color> colorOptional = jdbcColorDAO.get(colorId);

colorOptional.ifPresent(colorSet::add);

}

phone.setColors(colorSet);

}

return Optional.ofNullable(phone);

}

@Override

@Transactional

public void save(final Phone phone)

{

namedParameterJdbcTemplate.update(SQL\_INSERT\_PHONE, populateNamedParams(phone));

}

@Override

@Transactional(readOnly = true)

public List<Phone> findAll(final int offset, final int limit)

{

String query = SQL\_GET\_ALL\_PHONES + " offset " + offset + " limit " + limit;

return jdbcTemplate.query(query, phoneResultSetExtractor);

}

@Override

@Transactional(readOnly = true)

public List<Phone> findAll(final ParamsForSearch paramsForSearch)

{

final String search = paramsForSearch.getSearch();

final String sortField = paramsForSearch.getSortField();

final String order = paramsForSearch.getOrder();

final int offset = paramsForSearch.getOffset();

final int limit = paramsForSearch.getLimit();

final List<Object> objects = new ArrayList<>();

final List<Integer> types = new ArrayList<>();

String query = SQL\_GET\_ALL\_PHONES + SQL\_WHERE\_SEARCH;

if (search != null && !search.isEmpty())

{

query = query + "and lower(model) like lower(?) ";

objects.add("%" + search + "%");

types.add(Types.VARCHAR);

}

if (sortField != null && order != null)

{

query = query + String.format("group by phones.id,phone2color.colorId order by %s %s ", sortField, order);

}

query = query + " limit " + limit + " offset " + offset;

final int[] typesArray = types.stream()

.mapToInt(i -> i)

.toArray();

return jdbcTemplate.query(query, objects.toArray(), typesArray, phoneResultSetExtractor);

}

@Override

public Long count(final ParamsForSearch paramsForSearch)

{

final String search = paramsForSearch.getSearch();

String request = SQL\_SELECT\_COUNT\_FIND\_ALL\_EXTENDED + SQL\_WHERE\_SEARCH;

if (search != null && !search.isEmpty())

{

request = request + "and (lower(model) like lower(?))";

return jdbcTemplate.queryForObject(request, new Object[]{"%" + search +

"%"}, new int[]{Types.VARCHAR}, Long.class);

}

else

{

return jdbcTemplate.queryForObject(request, Long.class);

}

}

@Override

public List<Phone> findMaxDiscountPercentPhones()

{

return jdbcTemplate.query(SQL\_GET\_MAX\_DISCOUNTED\_PHONES, phoneResultSetExtractor);

}

@Override

public void deletePhone(final Long id)

{

jdbcTemplate.update(DELETE\_PHONE\_SQL, id);

}

@Override

@Transactional

public void updateDevice(final Phone phone)

{

namedParameterJdbcTemplate.update(SQL\_UPDATE\_PHONE, populateNamedParams(phone));

}

private MapSqlParameterSource populateNamedParams(final Phone phone)

{

MapSqlParameterSource in = new MapSqlParameterSource();

in.addValue("id", phone.getId());

in.addValue("brand", phone.getBrand());

in.addValue("model", phone.getModel());

in.addValue("price", phone.getPrice());

in.addValue("discountPercent", phone.getDiscountPercent());

in.addValue("displaySizeInches", phone.getDisplaySizeInches());

in.addValue("weightGr", phone.getWeightGr());

in.addValue("lengthMm", phone.getLengthMm());

in.addValue("widthMm", phone.getWidthMm());

in.addValue("heightMm", phone.getHeightMm());

in.addValue("deviceType", phone.getDeviceType());

in.addValue("displayResolution", phone.getDisplayResolution());

in.addValue("pixelDensity", phone.getPixelDensity());

in.addValue("displayTechnology", phone.getDisplayTechnology());

in.addValue("backCameraMegapixels", phone.getBackCameraMegapixels());

in.addValue("frontCameraMegapixels", phone.getFrontCameraMegapixels());

in.addValue("internalStorageGb", phone.getInternalStorageGb());

in.addValue("batteryCapacityMah", phone.getBatteryCapacityMah());

in.addValue("talkTimeHours", phone.getTalkTimeHours());

in.addValue("standByTimeHours", phone.getStandByTimeHours());

in.addValue("bluetooth", phone.getBluetooth());

in.addValue("imageUrl", phone.getImageUrl());

in.addValue("description", phone.getDescription());

return in;

}

}

Файл JdbcStockDao.java

package by.bsuir.phoneshop.core.dao.impl;

import java.util.List;

import java.util.Optional;

import javax.annotation.Resource;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Component;

import by.bsuir.phoneshop.core.models.Stock;

import by.bsuir.phoneshop.core.dao.StockDao;

@Component

public class JdbcStockDao implements StockDao

{

@Resource

private JdbcTemplate jdbcTemplate;

private static final String SQL\_GET\_STOCK = "select \* from stocks where phoneId = ";

private static final String SQL\_UPDATE = "update stocks set stock = %d, reserved = %d where phoneId = %d";

private static final String SQL\_INSERT = "insert into stocks(phoneId,stock,reserved) values (?,?,?)";

@Override

public Optional<Stock> get(final Long key)

{

final List<Stock> stocks = jdbcTemplate.query(SQL\_GET\_STOCK + key, new BeanPropertyRowMapper<>(Stock.class));

return stocks.stream().findFirst();

}

@Override

public void update(final Long key, final Long stock, final Long reserved, boolean isNewPhone)

{

String query;

if (!isNewPhone)

{

query = String.format(SQL\_UPDATE, stock, reserved, key);

jdbcTemplate.update(query);

}

else

{

jdbcTemplate.update(SQL\_INSERT, key, stock, reserved);

}

}

}

Файл JdbcUserDao.java

package by.bsuir.phoneshop.core.dao.impl;

import java.util.List;

import java.util.Optional;

import javax.annotation.Resource;

import org.springframework.dao.EmptyResultDataAccessException;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Service;

import by.bsuir.phoneshop.core.models.Employee;

import by.bsuir.phoneshop.core.models.User;

import by.bsuir.phoneshop.core.dao.UserDao;

import by.bsuir.phoneshop.core.dao.extractors.EmployeeListResultSetExtractor;

import by.bsuir.phoneshop.core.dao.extractors.UserListResultSetExtractor;

@Service

public class JdbcUserDao implements UserDao

{

@Resource

private JdbcTemplate jdbcTemplate;

@Resource

private UserListResultSetExtractor resultSetExtractor;

@Resource

private EmployeeListResultSetExtractor employeeListResultSetExtractor;

private static final String SQL\_GET\_EMPLOYEES = "select \* from employees";

private static final String SQL\_GET\_USER = "select \* from users where userName=";

private static final String SQL\_GET\_USERS\_EXCEPT\_ADMIN = "select \* from users where userName not like 'admin'";

private static final String SQL\_UPDATE\_USER\_STATUS = "update users set isAccountNonLocked= ? where userName = ?";

private static final String SQL\_INSERT\_NEW\_USER = "insert into users(userName,password,isAccountNonLocked,role) values (?,?,?,?)";

@Override

public Optional<User> getUserByUsername(final String username) throws EmptyResultDataAccessException

{

return Optional.ofNullable(jdbcTemplate.queryForObject(SQL\_GET\_USER + "'" + username + "'", new BeanPropertyRowMapper<User>(User.class)));

}

@Override

public List<User> getUsersExceptAdmin(int limit, int offset)

{

return jdbcTemplate.query(SQL\_GET\_USERS\_EXCEPT\_ADMIN + " limit " + limit + " offset " + offset, resultSetExtractor);

}

@Override

public void updateUserStatus(int status, String username)

{

jdbcTemplate.update(SQL\_UPDATE\_USER\_STATUS, status, username);

}

@Override

public void addNewUserToDb(final User user)

{

jdbcTemplate.update(SQL\_INSERT\_NEW\_USER, user.getUserName(), user.getPassword(), user.getIsAccountNonLocked(), user.getRole());

}

@Override

public List<Employee> getMainEmployees()

{

return jdbcTemplate.query(SQL\_GET\_EMPLOYEES, employeeListResultSetExtractor);

}

}

Файл AuthProviver.java

package by.bsuir.phoneshop.core.service.impl;

import java.util.Arrays;

import javax.annotation.Resource;

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

import org.springframework.security.authentication.AuthenticationProvider;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.LockedException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

import org.springframework.stereotype.Component;

import by.bsuir.phoneshop.core.models.User;

import by.bsuir.phoneshop.core.service.UserService;

@Component

public class AuthProvider implements AuthenticationProvider

{

@Resource

private UserService userDetailsService;

@Value("${user.password.encryption.strength}")

private int ENCRYPTION\_STRENGTH;

@Override

public Authentication authenticate(final Authentication authentication) throws AuthenticationException

{

final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder(ENCRYPTION\_STRENGTH);

final String username = authentication.getName();

User user;

try

{

user = userDetailsService.loadUserByUsername(username);

}

catch (final Exception e)

{

throw new BadCredentialsException("Username or password is invalid");

}

if (user.getIsAccountNonLocked() == 0)

{

throw new LockedException("You are blocked!");

}

else

{

if (passwordEncoder.matches(authentication.getCredentials().toString(), user.getPassword()))

{

return new UsernamePasswordAuthenticationToken(authentication.getName(), authentication.getCredentials()

.toString(), Arrays.asList(new SimpleGrantedAuthority(user.getRole())));

}

else throw new BadCredentialsException("Username or password is invalid");

}

}

@Override

public boolean supports(final Class<?> aClass)

{

return aClass.equals(UsernamePasswordAuthenticationToken.class);

}

}

Файл HttpSessionCartService.java

package by.bsuir.phoneshop.core.service.impl;

import java.math.BigDecimal;

import java.util.Map;

import java.util.Optional;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.dao.DataAccessException;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import by.bsuir.phoneshop.core.dao.PhoneDao;

import by.bsuir.phoneshop.core.dao.StockDao;

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.models.CartItem;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.models.Stock;

import by.bsuir.phoneshop.core.service.CartService;

@Service

public class HttpSessionCartService implements CartService

{

public static final String CART\_SESSION\_ATTR = "cart";

@Resource

private PhoneDao jdbcPhoneDao;

@Resource

private StockDao jdbcStockDao;

@Override

@Transactional(readOnly = true)

public synchronized Cart getCart(final HttpSession httpSession)

{

Cart cart = (Cart) httpSession.getAttribute(CART\_SESSION\_ATTR);

if (cart == null)

{

cart = new Cart();

httpSession.setAttribute(CART\_SESSION\_ATTR, cart);

}

return cart;

}

@Override

@Transactional(rollbackFor = DataAccessException.class)

public synchronized void addPhone(final Long phoneId, final Long quantity, final Cart cart) throws RuntimeException

{

final Optional<Phone> optionalPhone = jdbcPhoneDao.get(phoneId);

final Optional<Stock> optionalStock = jdbcStockDao.get(phoneId);

if (optionalPhone.isPresent() && optionalStock.isPresent())

{

final Stock stock = optionalStock.get();

final Phone phone = optionalPhone.get();

if (stock.getStock() - stock.getReserved() >= quantity)

{

addToCart(quantity, stock, phone, cart);

}

else

{

throw new RuntimeException();

}

}

else

{

throw new IllegalArgumentException("No stock or phone in data base");

}

}

private void addToCart(final Long quantity, final Stock stock, final Phone phone, Cart cart)

{

jdbcStockDao.update(phone.getId(), stock.getStock().longValue(), stock.getReserved() + quantity, false);

final Optional<CartItem> cartItem = findCartItem(phone.getId(), cart);

if (cartItem.isPresent())

{

final CartItem existingCartItem = cartItem.get();

existingCartItem.setQuantity(existingCartItem.getQuantity() + quantity);

}

else

{

final BigDecimal price = phone.getActualPrice().multiply(BigDecimal.valueOf(quantity));

cart.getCartItems().add(new CartItem(phone, quantity, price));

}

calculateCart(cart);

}

@Override

@Transactional(rollbackFor = DataAccessException.class)

public synchronized void update(final Map<Long, Long> items, final Cart cart)

{

items.keySet().stream()

.map(phoneId -> findCartItem(phoneId, cart))

.filter(Optional::isPresent)

.map(Optional::get)

.forEach(cartItem ->

{

final Long phoneId = cartItem.getPhone().getId();

final Long quantity = items.get(cartItem.getPhone().getId());

final Long quantityDifference = quantity - cartItem.getQuantity();

if (checkQuantity(phoneId, quantityDifference))

{

cartItem.setQuantity(cartItem.getQuantity() + quantityDifference);

}

});

calculateCart(cart);

}

private boolean checkQuantity(final Long phoneId, final Long quantityDifference)

{

final Optional<Stock> optionalStock = jdbcStockDao.get(phoneId);

if (optionalStock.isPresent())

{

final Stock stock = optionalStock.get();

if (stock.getStock() - stock.getReserved() >= quantityDifference)

{

jdbcStockDao.update(phoneId, stock.getStock().longValue(),

stock.getReserved() + quantityDifference, false);

return true;

}

}

return false;

}

@Override

@Transactional(rollbackFor = DataAccessException.class)

public synchronized void remove(final Long phoneId, final Cart cart)

{

final Optional<CartItem> optionalCartItem = findCartItem(phoneId, cart);

if (optionalCartItem.isPresent())

{

cart.getCartItems().remove(optionalCartItem.get());

}

else

{

throw new IllegalArgumentException(phoneId.toString());

}

final Optional<Stock> optionalStock = jdbcStockDao.get(phoneId);

if (optionalStock.isPresent() && optionalCartItem.isPresent())

{

final Stock stock = optionalStock.get();

final CartItem cartItem = optionalCartItem.get();

jdbcStockDao.update(phoneId, stock.getStock() + cartItem.getQuantity(),

stock.getReserved() - cartItem.getQuantity(), false);

}

calculateCart(cart);

}

@Override

public void deleteCart(final HttpSession httpSession)

{

httpSession.removeAttribute("cart");

}

private Optional<CartItem> findCartItem(final Long phoneId, final Cart cart)

{

return cart.getCartItems().stream()

.filter(cartItem -> cartItem.getPhone().getId().equals(phoneId))

.findFirst();

}

private void calculateCart(final Cart cart)

{

final Long totalQuantity = cart.getCartItems().stream()

.mapToLong(CartItem::getQuantity)

.sum();

cart.setTotalQuantity(totalQuantity);

final BigDecimal totalCost = cart.getCartItems().stream()

.map(cartItem -> cartItem.getPhone().getActualPrice().multiply(BigDecimal.valueOf(cartItem.getQuantity())))

.reduce(BigDecimal.ZERO, BigDecimal::add);

cart.setTotalCost(totalCost);

}

}

Файл OrderServiceImpl.java

package by.bsuir.phoneshop.core.service.impl;

import java.math.BigDecimal;

import java.util.List;

import java.util.Optional;

import java.util.UUID;

import java.util.concurrent.atomic.AtomicBoolean;

import javax.annotation.Resource;

import org.springframework.dao.DataAccessException;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.models.Order;

import by.bsuir.phoneshop.core.dao.OrderDao;

import by.bsuir.phoneshop.core.dto.OrderDataDto;

import by.bsuir.phoneshop.core.models.OrderItem;

import by.bsuir.phoneshop.core.models.enums.OrderStatus;

import by.bsuir.phoneshop.core.models.Stock;

import by.bsuir.phoneshop.core.dao.StockDao;

import by.bsuir.phoneshop.core.service.OrderService;

@Service

public class OrderServiceImpl implements OrderService

{

@Resource

private OrderDao jdbcOrderDao;

@Resource

private StockDao jdbcStockDao;

@Override

@Transactional(rollbackFor = DataAccessException.class)

public Long placeOrder(final Cart cart, OrderDataDto orderDataDto, final Long deliveryPrice) throws RuntimeException

{

final AtomicBoolean isOutOfStock = new AtomicBoolean(false);

final Order order = createOrderFromCart(cart, orderDataDto, deliveryPrice);

order.getOrderItems().forEach(orderItem ->

{

final Stock stock = jdbcStockDao.get(orderItem.getPhone().getId()).orElse(null);

if (stock != null && stock.getStock() - orderItem.getQuantity() > 0)

{

jdbcStockDao.update(orderItem.getPhone().getId(), stock.getStock() - orderItem.getQuantity(), stock.getReserved() - orderItem.getQuantity(), false);

}

else

{

isOutOfStock.set(true);

}

});

if (isOutOfStock.get())

{

throw new RuntimeException();

}

order.setId(UUID.randomUUID().getMostSignificantBits() & Long.MAX\_VALUE);

return jdbcOrderDao.save(order);

}

@Override

@Transactional(readOnly = true)

public List<Order> getOrders(final int limit,final int offset)

{

return jdbcOrderDao.getOrders(limit, offset);

}

@Override

public void updateStatus(final OrderStatus orderStatus, final Long orderId)

{

jdbcOrderDao.updateStatus(orderStatus, orderId);

}

@Override

public Optional<Order> getOrderById(final Long id)

{

return jdbcOrderDao.get(id);

}

private Order createOrderFromCart(final Cart cart, final OrderDataDto orderDataDto, final Long deliveryPrice)

{

final Order order = new Order();

order.setSubtotal(cart.getTotalCost());

order.setDeliveryPrice(BigDecimal.valueOf(deliveryPrice));

order.setTotalPrice(order.getSubtotal().add(order.getDeliveryPrice()));

order.setStatus(OrderStatus.NEW);

order.setFirstName(orderDataDto.getFirstName());

order.setLastName(orderDataDto.getLastName());

order.setContactPhoneNo(orderDataDto.getPhone());

order.setDeliveryAddress(orderDataDto.getAddress());

order.setAdditionalInfo(orderDataDto.getAdditionalInfo());

cart.getCartItems().forEach(cartItem ->

{

final OrderItem orderItem = new OrderItem(cartItem, order);

order.getOrderItems().add(orderItem);

});

return order;

}

}

Файл PhoneServiceImpl.java

package by.bsuir.phoneshop.core.service.impl;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import javax.annotation.Resource;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Component;

import by.bsuir.phoneshop.core.models.ParamsForSearch;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.dao.PhoneDao;

import by.bsuir.phoneshop.core.dao.StockDao;

import by.bsuir.phoneshop.core.dto.PhoneAddDto;

import by.bsuir.phoneshop.core.service.PhoneService;

@Component

public class PhoneServiceImpl implements PhoneService

{

@Resource

private PhoneDao jdbcPhoneDao;

@Resource

private StockDao jdbcStockDao;

@Resource

private JdbcTemplate jdbcTemplate;

@Override

public Optional<Phone> getPhoneByKey(final Long key)

{

return jdbcPhoneDao.get(key);

}

@Override

public Optional<Phone> getPhoneByModel(final String model)

{

return jdbcPhoneDao.get(model);

}

@Override

public void savePhone(final Phone phone, final Long quantity, final String[] colors)

{

jdbcPhoneDao.save(phone);

final Phone savedPhone = getPhoneByModel(phone.getModel()).get();

jdbcStockDao.update(savedPhone.getId(), quantity, 0L, true);

for (final String color : colors)

{

jdbcTemplate.update("insert into phone2color(phoneId,colorId) values (?,?)", savedPhone.getId(), color);

}

}

@Override

public List<Phone> getPhones(final int offset, final int limit)

{

return jdbcPhoneDao.findAll(offset, limit);

}

@Override

public List<Phone> getPhones(final ParamsForSearch paramsForSearch)

{

return jdbcPhoneDao.findAll(paramsForSearch);

}

@Override

public Long countPhoneQuantity(final ParamsForSearch paramsForSearch)

{

return jdbcPhoneDao.count(paramsForSearch);

}

@Override

public List<Phone> getHotPricedPhones(final int amount)

{

final List<Phone> phones = jdbcPhoneDao.findMaxDiscountPercentPhones();

final List<Phone> limitedPhoneList = new ArrayList<>();

for (int i = 0; i < phones.size(); i++)

{

if (i < amount)

{

limitedPhoneList.add(phones.get(i));

}

else

{

break;

}

}

return limitedPhoneList;

}

@Override

public void deletePhone(final Long id)

{

jdbcPhoneDao.deletePhone(id);

}

@Override

public Phone populateFields(final PhoneAddDto phoneAddDto)

{

final Phone phone = new Phone();

phone.setBrand(phoneAddDto.getBrand());

phone.setModel(phoneAddDto.getModel());

phone.setPrice(phoneAddDto.getPrice());

if (phoneAddDto.getDiscountPercent() != null)

{

phone.setDiscountPercent(phoneAddDto.getDiscountPercent());

}

else

{

phone.setDiscountPercent(0);

}

phone.setImageUrl(phoneAddDto.getImageUrl());

phone.setDescription(phoneAddDto.getDescription());

phone.setDisplaySizeInches(phoneAddDto.getDisplaySizeInches());

phone.setDisplayResolution(phoneAddDto.getDisplayResolution());

phone.setPixelDensity(phoneAddDto.getPixelDensity());

phone.setDisplayTechnology(phoneAddDto.getDisplayTechnology());

phone.setWeightGr(phoneAddDto.getWeightGr());

phone.setLengthMm(phoneAddDto.getLengthMm());

phone.setWidthMm(phoneAddDto.getWidthMm());

phone.setHeightMm(phoneAddDto.getHeightMm());

phone.setDeviceType(phoneAddDto.getDeviceType());

phone.setBackCameraMegapixels(phoneAddDto.getBackCameraMegapixels());

phone.setFrontCameraMegapixels(phoneAddDto.getFrontCameraMegapixels());

phone.setInternalStorageGb(phoneAddDto.getInternalStorageGb());

phone.setBatteryCapacityMah(phoneAddDto.getBatteryCapacityMah());

phone.setTalkTimeHours(phoneAddDto.getTalkTimeHours());

phone.setStandByTimeHours(phoneAddDto.getStandByTimeHours());

phone.setBluetooth(phoneAddDto.getBluetooth());

return phone;

}

@Override

public void updateDevice(final Phone phone, final Long quantity, final String[] colors, final Long id)

{

phone.setId(id);

jdbcPhoneDao.updateDevice(phone);

final Phone savedPhone = getPhoneByModel(phone.getModel()).get();

jdbcStockDao.update(savedPhone.getId(), quantity, 0L, false);

for (final String color : colors)

{

jdbcTemplate.update("delete from phone2color where phoneId=?", savedPhone.getId());

jdbcTemplate.update("insert into phone2color(phoneId,colorId) values (?,?)", savedPhone.getId(), color);

}

}

}

Файл AddDeviceController.java

package by.bsuir.phoneshop.web.controller.pages.admin.devicemanagement;

import javax.annotation.Resource;

import org.springframework.context.MessageSource;

import org.springframework.context.i18n.LocaleContextHolder;

import org.springframework.http.MediaType;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.Validator;

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.PostMapping;

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

import by.bsuir.phoneshop.core.dto.PhoneAddDto;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping("/admin/device-operations/add")

public class AddDeviceController

{

@Resource

private PhoneService phoneServiceImpl;

@Resource(name = "phoneAddDtoValidator")

private Validator phoneAddDtoValidator;

@Resource

private MessageSource messageSource;

@GetMapping

public String showAddDevicePage(final Model model)

{

model.addAttribute("phoneAddDto", new PhoneAddDto());

return PhoneshopPages.UserPages.AddDevicePage;

}

@PostMapping(consumes = MediaType.APPLICATION\_FORM\_URLENCODED\_VALUE)

public String addDeviceToDataBase(final @Validated @ModelAttribute(name = "phoneAddDto") PhoneAddDto phoneAddDto, final BindingResult bindingResult, final Model model)

{

phoneAddDtoValidator.validate(phoneAddDto, bindingResult);

final String[] stringColors = phoneAddDto.getColors().split(",");

if (bindingResult.hasErrors())

{

model.addAttribute("recentData", phoneAddDto);

model.addAttribute("errors", bindingResult);

model.addAttribute("error", messageSource.getMessage("page.admin.error.while.adding.device.to.db", null, LocaleContextHolder.getLocale()));

return PhoneshopPages.UserPages.AddDevicePage;

}

phoneServiceImpl.savePhone(phoneServiceImpl.populateFields(phoneAddDto), phoneAddDto.getQuantity(), stringColors);

model.addAttribute("success", messageSource.getMessage("page.admin.success.while.adding.device.to.db", null,

LocaleContextHolder.getLocale()));

return PhoneshopPages.UserPages.AddDevicePage;

}

}

Файл PhoneDtoValidator.java

package by.bsuir.phoneshop.web.controller.pages.admin;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

import by.bsuir.phoneshop.core.models.Order;

import by.bsuir.phoneshop.core.service.OrderService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/admin/orders")

public class OrdersPageController

{

@Resource

private OrderService orderServiceImpl;

@RequestMapping(method = RequestMethod.GET)

public String getOrders(final Model model)

{

final List<Order> orders = orderServiceImpl.getOrders(Integer.MAX\_VALUE, 0);

if (orders != null)

{

model.addAttribute(orders);

}

return PhoneshopPages.UserPages.OrdersPage;

}

}

Файл UserManagementController.java

package by.bsuir.phoneshop.web.controller.pages.admin;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

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

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

import by.bsuir.phoneshop.core.models.User;

import by.bsuir.phoneshop.core.exception.errors.ResponseErrors;

import by.bsuir.phoneshop.core.dto.UserDto;

import by.bsuir.phoneshop.core.service.UserService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/admin/user-management")

public class UserManagementController

{

@Resource

private UserService userServiceImpl;

private static final int LIMIT\_ON\_PAGE = 10;

@GetMapping

public String showUserManagementPage(final Model model, @RequestParam(name = "page", required = false) Integer page)

{

if (page == null)

{

page = 0;

}

int offset = page \* LIMIT\_ON\_PAGE;

final List<User> users = userServiceImpl.getUsersExceptAdmin(LIMIT\_ON\_PAGE, offset);

model.addAttribute("users", users);

return PhoneshopPages.UserPages.UserManagementPage;

}

@RequestMapping(method = RequestMethod.POST, value = "/update-status", consumes = "application/json")

public ResponseEntity<?> updateUserStatus(final @RequestBody UserDto userDto)

{

int statusToUpdate = userDto.getStatus() == 0 ? 1 : 0;

try

{

userServiceImpl.updateUserStatus(statusToUpdate, userDto.getUserName());

return ResponseEntity.ok().build();

}

catch (Exception e)

{

final ResponseErrors errors = new ResponseErrors(e.getMessage());

return ResponseEntity.badRequest().body(errors);

}

}

}

Файл AboutUsPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

import by.bsuir.phoneshop.core.service.UserService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/about-us")

public class AboutUsPageController

{

@Resource

private UserService userService;

@GetMapping

public String showAboutUsPage(final Model model)

{

model.addAttribute("employees", userService.getEmployees());

return PhoneshopPages.UserPages.AboutUsPage;

}

}

Файл CartPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.util.HashMap;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collectors;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.FieldError;

import org.springframework.validation.annotation.Validated;

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

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

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

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

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

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.dto.PhoneArrayDto;

import by.bsuir.phoneshop.core.service.CartService;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.core.validator.PhoneArrayDtoValidator;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/cart")

public class CartPageController

{

public static final String REDIRECT\_TO\_CART\_PAGE = "redirect:/" + PhoneshopPages.UserPages.CartPage;

@Resource

private CartService cartService;

@Resource

private HttpSession httpSession;

@Resource

private PhoneService phoneServiceImpl;

@Resource

private PhoneArrayDtoValidator phoneArrayDtoValidator;

@RequestMapping(method = RequestMethod.GET)

public String getCart(final @RequestParam(value = "successDelete", required = false) boolean successDelete,

final @RequestParam(value = "successUpdate", required = false) boolean successUpdate,

final @RequestParam(value = "isOutOfStock", required = false) boolean isOutOfStock,

final @RequestParam(value = "error", required = false) boolean error,

final @RequestParam(value = "errorsId", required = false) List<Long> errorsId,

final Model model)

{

final Cart cart = cartService.getCart(httpSession);

model.addAttribute("successDelete", successDelete);

model.addAttribute("successUpdate", successUpdate);

model.addAttribute("isOutOfStock", isOutOfStock);

model.addAttribute("error", error);

model.addAttribute("errorsId", errorsId);

model.addAttribute("cart", cart);

return PhoneshopPages.UserPages.CartPage;

}

@RequestMapping(method = RequestMethod.POST)

public String updateCart(final @Validated @ModelAttribute(name = "phoneArrayDto") PhoneArrayDto phoneArrayDto, final Model model, final BindingResult bindingResult)

{

final Cart cart = cartService.getCart(httpSession);

if (cart.getCartItems().isEmpty())

{

return prepareModelForEmptyCart(cart, model);

}

phoneArrayDtoValidator.validate(phoneArrayDto, bindingResult);

if (bindingResult.hasErrors())

{

return validationFailed(cart, bindingResult, model);

}

final HashMap<Long, Long> idAndQuantity = new HashMap<>();

for (int i = 0; i < phoneArrayDto.getQuantity().length; i++)

{

idAndQuantity.put(Long.parseLong(phoneArrayDto.getPhoneId()[i]),

Long.parseLong(phoneArrayDto.getQuantity()[i]));

}

cartService.update(idAndQuantity, cart);

model.addAttribute("successUpdate", true);

model.addAttribute("cart", cart);

return REDIRECT\_TO\_CART\_PAGE;

}

@RequestMapping(value = "{id}", method = RequestMethod.POST)

public String deleteFromCart(final @PathVariable("id") Long id, final Model model) throws IllegalArgumentException

{

final Optional<Phone> optionalPhone = phoneServiceImpl.getPhoneByKey(id);

final Cart cart = cartService.getCart(httpSession);

if (cart.getCartItems().isEmpty())

{

return prepareModelForEmptyCart(cart, model);

}

if (optionalPhone.isPresent())

{

cartService.remove(id, cart);

model.addAttribute("successDelete", true);

model.addAttribute("cart", cart);

}

else

{

model.addAttribute("error", true);

throw new IllegalArgumentException(String.valueOf(id));

}

return REDIRECT\_TO\_CART\_PAGE;

}

private String prepareModelForEmptyCart(final Cart cart, final Model model)

{

model.addAttribute("error", true);

model.addAttribute("cart", cart);

return REDIRECT\_TO\_CART\_PAGE;

}

private String validationFailed(final Cart cart, final BindingResult bindingResult, final Model model)

{

final List<FieldError> errorList = bindingResult.getFieldErrors("quantity");

final List<Long> errorsId = errorList.stream()

.map(item -> Long.parseLong(item.getCode()))

.collect(Collectors.toList());

model.addAttribute("cart", cart);

model.addAttribute("errorsId", errorsId);

model.addAttribute("error", true);

return REDIRECT\_TO\_CART\_PAGE;

}

}

Файл ContactsPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import org.springframework.stereotype.Controller;

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

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

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping("/contacts")

public class ContactsPageController

{

@GetMapping

public String getContactsPage()

{

return PhoneshopPages.UserPages.ContactsPage;

}

}

Файл GlobalController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.nio.file.AccessDeniedException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.http.HttpStatus;

import org.springframework.ui.Model;

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

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

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

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

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

import org.springframework.web.servlet.NoHandlerFoundException;

@ControllerAdvice

@RequestMapping("/error")

public class GlobalController

{

private static final String ADMIN\_MAIL="mailto:antosa2972@gmail.com";

@ExceptionHandler(NoHandlerFoundException.class)

public String exception404(final HttpServletRequest httpRequest, final HttpServletResponse httpServletResponse, final Model model)

{

model.addAttribute("adminMail", ADMIN\_MAIL);

model.addAttribute("status", "404");

return "error";

}

@ExceptionHandler(AccessDeniedException.class)

@GetMapping("/access-denied")

public String accessDenied(final HttpServletRequest httpRequest, final HttpServletResponse httpServletResponse, final Model model)

{

model.addAttribute("adminMail", ADMIN\_MAIL);

model.addAttribute("status", "403");

return "error";

}

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

@ExceptionHandler(Exception.class)

public String internalServerError(final HttpServletRequest httpRequest, final HttpServletResponse httpServletResponse, final Model model)

{

model.addAttribute("adminMail", ADMIN\_MAIL);

model.addAttribute("status", "500");

return "error";

}

}

Файл HotPricesController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.util.List;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.util.CollectionUtils;

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

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

public class HotPricesPageController

{

private static final int AMOUNT\_TO\_GET = 10;

@Resource

private PhoneService phoneServiceImpl;

@GetMapping("/hot-prices")

public String getHotPricesPage(final Model model)

{

final List<Phone> phonesWithHotPrices = phoneServiceImpl.getHotPricedPhones(AMOUNT\_TO\_GET);

if (phonesWithHotPrices != null && !CollectionUtils.isEmpty(phonesWithHotPrices))

{

model.addAttribute("phones", phonesWithHotPrices);

}

else

{

model.addAttribute("isEmptyPhoneList", true);

}

return PhoneshopPages.UserPages.HotPricesPage;

}

}

Файл OrderOverviewPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

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

import by.bsuir.phoneshop.core.service.OrderService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/order-overview/{id}")

public class OrderOverviewPageController

{

@Resource

OrderService orderServiceImpl;

@RequestMapping(method = RequestMethod.GET)

public String getOrderOverview(final @PathVariable("id") Long id, final Model model)

{

orderServiceImpl.getOrderById(id).ifPresent(order -> model.addAttribute("order", order));

model.addAttribute("id", id);

return PhoneshopPages.UserPages.OrderOverviewPage;

}

@ExceptionHandler(RuntimeException.class)

public String handleError()

{

return "redirect:/error?message=" + "no such order";

}

}

Файл OrderPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.math.BigDecimal;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.context.MessageSource;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.i18n.LocaleContextHolder;

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.ExceptionHandler;

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

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

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

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.dto.OrderDataDto;

import by.bsuir.phoneshop.core.service.CartService;

import by.bsuir.phoneshop.core.service.OrderService;

import by.bsuir.phoneshop.core.validator.OrderDataDtoValidator;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@PropertySource("classpath:/messages\_en.properties")

@RequestMapping(value = "/order")

public class OrderPageController

{

@Resource

private OrderService orderService;

@Resource

private CartService cartService;

@Resource

private HttpSession httpSession;

@Resource

private OrderDataDtoValidator orderDataDtoValidator;

@Resource

private MessageSource messageSource;

@RequestMapping(method = RequestMethod.GET)

public String getOrder(final Model model)

{

model.addAttribute("errors", model.asMap().get("errors"));

model.addAttribute("error", model.asMap().get("error"));

Cart cart = cartService.getCart(httpSession);

showCartAsOrder(cart, model);

return PhoneshopPages.UserPages.OrderPage;

}

@RequestMapping(method = RequestMethod.POST)

public String placeOrder(final @Validated @ModelAttribute(name = "orderDataDto") OrderDataDto orderDataDto,

final BindingResult bindingResult,

final RedirectAttributes redirectAttributes) throws RuntimeException

{

final Cart cart = cartService.getCart(httpSession);

if (cart.getCartItems().isEmpty())

{

return prepareModelForEmptyCart(redirectAttributes);

}

orderDataDtoValidator.validate(orderDataDto, bindingResult);

if (bindingResult.hasErrors())

{

return prepareModelForValidationErrors(bindingResult, redirectAttributes);

}

final Long id = orderService.placeOrder(cart, orderDataDto, Long.parseLong(messageSource.getMessage("delivery.price", null, LocaleContextHolder.getLocale())));

cartService.deleteCart(httpSession);

return "redirect:/" + PhoneshopPages.UserPages.OrderOverviewPage + "/" + id;

}

private String prepareModelForValidationErrors(final BindingResult bindingResult,

final RedirectAttributes redirectAttributes)

{

redirectAttributes.addFlashAttribute("error", true);

redirectAttributes.addFlashAttribute("errors", bindingResult);

return "redirect:/" + PhoneshopPages.UserPages.OrderPage;

}

private String prepareModelForEmptyCart(final RedirectAttributes redirectAttributes)

{

redirectAttributes.addFlashAttribute("error", true);

return "redirect:/" + PhoneshopPages.UserPages.OrderPage;

}

private void showCartAsOrder(final Cart cart, final Model model)

{

model.addAttribute("cart", cart);

final BigDecimal deliveryPrice = BigDecimal.valueOf(Long.parseLong(

messageSource.getMessage("delivery.price", null, LocaleContextHolder.getLocale())));

model.addAttribute("deliveryPrice", deliveryPrice);

model.addAttribute("totalCost", cart.getTotalCost().add(deliveryPrice));

}

@ExceptionHandler(RuntimeException.class)

public String handleError()

{

return "redirect:/error?message=" + messageSource.getMessage("exception.outOfStock", null, LocaleContextHolder.getLocale());

}

}

Файл ProductDetailsPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.service.CartService;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/product-details/{id}")

public class ProductDetailsPageController

{

@Resource

private PhoneService phoneServiceImpl;

@Resource

private CartService httpSessionCartService;

@Resource

private HttpSession httpSession;

@RequestMapping(method = RequestMethod.GET)

public String showProductDetailsInfo(final @PathVariable("id") Long phoneId, Model model)

{

final Phone phone = phoneServiceImpl.getPhoneByKey(phoneId).orElse(null);

model.addAttribute("phone", phone);

model.addAttribute("cart", httpSessionCartService.getCart(httpSession));

return PhoneshopPages.UserPages.ProductDetailsPage;

}

}

Файл ProductListController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.util.List;

import java.util.Locale;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import by.bsuir.phoneshop.core.models.ParamsForSearch;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.models.enums.SortField;

import by.bsuir.phoneshop.core.models.enums.SortOrder;

import by.bsuir.phoneshop.core.service.CartService;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/product-list")

public class ProductListPageController

{

public static final Long QUANTITY\_ON\_PAGE = 10L;

@Resource

private PhoneService phoneServiceImpl;

@Resource

private CartService cartService;

@Resource

private HttpSession httpSession;

@RequestMapping(method = RequestMethod.GET)

public String showProductList(final @RequestParam(required = false) String search,

@RequestParam(required = false) String field,

@RequestParam(required = false) String order,

@RequestParam(required = false) Long page, final Model model)

{

if (page == null)

{

page = 1L;

}

if (field != null && order != null)

{

try

{

field = SortField.valueOf(field.toUpperCase(Locale.ROOT)).name();

order = SortOrder.valueOf(order.toUpperCase(Locale.ROOT)).name();

}

catch (IllegalArgumentException e)

{

field = null;

order = null;

}

}

long offset = (page - 1) \* QUANTITY\_ON\_PAGE;

final ParamsForSearch paramsForSearch = new ParamsForSearch(search,

field, order, (int) offset,

QUANTITY\_ON\_PAGE.intValue());

final List<Phone> phoneList = phoneServiceImpl.getPhones(paramsForSearch);

final Long phoneQuantity = phoneServiceImpl.countPhoneQuantity(paramsForSearch);

long numOfPages = phoneQuantity / QUANTITY\_ON\_PAGE;

long lastPage;

if (phoneQuantity % QUANTITY\_ON\_PAGE != 0)

{

lastPage = numOfPages + 2;

}

else

{

lastPage = numOfPages + 1;

}

model.addAttribute("phones", phoneList);

model.addAttribute("cart", cartService.getCart(httpSession));

model.addAttribute("pages", lastPage);

model.addAttribute("phoneQuantity", phoneQuantity);

return PhoneshopPages.UserPages.ProductListPage;

}

}

Файл QuickOrderController.java

package by.bsuir.phoneshop.web.controller.pages;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

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.ModelAttribute;

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

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

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.models.Phone;

import by.bsuir.phoneshop.core.dto.QuickOrderElement;

import by.bsuir.phoneshop.core.dto.QuickOrderElementsDto;

import by.bsuir.phoneshop.core.service.CartService;

import by.bsuir.phoneshop.core.service.PhoneService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/quick-order")

public class QuickOrderController

{

@Resource

private HttpSession httpSession;

@Resource

private PhoneService phoneServiceImpl;

@Resource

private CartService httpSessionCartService;

private final List<Phone> phoneList;

private QuickOrderController()

{

this.phoneList = new ArrayList<>();

}

@RequestMapping(method = RequestMethod.GET)

public String getQuickOrderPage(final Model model)

{

model.addAttribute("success", model.asMap().get("success"));

model.addAttribute("quickOrderElementsDto", new QuickOrderElementsDto());

return "quick-order";

}

@RequestMapping(method = RequestMethod.POST)

public String makeQuickOrder(final @ModelAttribute(name = "quickOrderElementsDto") @Validated QuickOrderElementsDto quickOrderElementsDto,

final BindingResult bindingResult,

final RedirectAttributes redirectAttributes,

final Model model)

{

Cart cart = httpSessionCartService.getCart(httpSession);

quickOrderElementsDto.getQuickOrderElements().forEach(quickOrderElement ->

{

int indexInList = quickOrderElementsDto.getQuickOrderElements().indexOf(quickOrderElement);

addToCart(quickOrderElementsDto, cart, bindingResult, indexInList);

});

if (bindingResult.hasFieldErrors())

{

model.addAttribute("successfulPhones", phoneList);

model.addAttribute("errors", bindingResult);

return PhoneshopPages.UserPages.QuickOrderPage;

}

redirectAttributes.addFlashAttribute("success", true);

return "redirect:" + PhoneshopPages.UserPages.QuickOrderPage;

}

private void addToCart(final QuickOrderElementsDto quickOrderElementsDto,

final Cart cart, final BindingResult bindingResult, final int indexInList)

{

final QuickOrderElement quickOrderElement = quickOrderElementsDto.getQuickOrderElements().get(indexInList);

final Long quantity = quickOrderElement.getQuantity();

final String modelPhone = quickOrderElement.getModel();

if (quantity == null && modelPhone.isEmpty())

{

return;

}

final Optional<Phone> optionalPhone = phoneServiceImpl.getPhoneByModel(modelPhone);

optionalPhone.ifPresent(phone ->

{

if (quantity != null)

{

try

{

httpSessionCartService.addPhone(phone.getId(), quantity, cart);

quickOrderElementsDto.getQuickOrderElements().set(indexInList, new QuickOrderElement());

}

catch (RuntimeException e)

{

bindingResult.rejectValue("quickOrderElements[" + indexInList + "].quantity",

"errors.quantity.outOfStock", "Out of stock!");

}

phoneList.add(phone);

}

else

{

bindingResult.rejectValue("quickOrderElements[" + indexInList + "].quantity",

"errors.emptyField", "Empty field!");

}

});

if (!optionalPhone.isPresent())

{

bindingResult.rejectValue("quickOrderElements[" + indexInList + "].model",

"errors.product.notFound ", "No such product!");

}

}

}

Файл RegisterPageController.java

package by.bsuir.phoneshop.web.controller.pages;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

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

import by.bsuir.phoneshop.core.service.UserService;

import by.bsuir.phoneshop.web.controller.constants.PhoneshopPages;

@Controller

@RequestMapping(value = "/register")

public class RegisterPageController

{

@Resource

private UserService userServiceImpl;

@GetMapping

public String showRegisterPage()

{

return PhoneshopPages.UserPages.RegisterPage;

}

@PostMapping

public String registerUser(final @RequestParam String username, @RequestParam final String password1, final Model model)

{

try

{

if (username == null || username.isEmpty() || password1 == null || password1.isEmpty() || username.length() < 5 || password1.length() < 5)

{

model.addAttribute("errorEmpty", true);

return PhoneshopPages.UserPages.RegisterPage;

}

userServiceImpl.registerUser(username, password1);

}

catch (final Exception e)

{

model.addAttribute("error", true);

return PhoneshopPages.UserPages.RegisterPage;

}

model.addAttribute("isRegistered", true);

return PhoneshopPages.UserPages.LoginPage;

}

}

Файл AjaxCartController.java

package by.bsuir.phoneshop.web.controller;

import static org.springframework.http.ResponseEntity.badRequest;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.validation.Validator;

import org.springframework.validation.annotation.Validated;

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

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

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

import by.bsuir.phoneshop.core.models.Cart;

import by.bsuir.phoneshop.core.exception.errors.ResponseErrors;

import by.bsuir.phoneshop.core.exception.errors.ValidationErrors;

import by.bsuir.phoneshop.core.dto.PhoneDto;

import by.bsuir.phoneshop.core.service.CartService;

@Controller

@RequestMapping(value = "/ajax-cart")

public class AjaxCartController

{

@Resource(name = "phoneDtoValidator")

private Validator quantityValidator;

@Resource

private CartService cartService;

@Resource

private HttpSession httpSession;

@RequestMapping(method = RequestMethod.POST, consumes = "application/json")

public ResponseEntity<?> addPhone(final @Validated @RequestBody PhoneDto phoneDto, final BindingResult bindingResult)

{

quantityValidator.validate(phoneDto, bindingResult);

if (bindingResult.hasErrors())

{

ValidationErrors validationErrors = new ValidationErrors(bindingResult.getAllErrors());

return badRequest().body(validationErrors.getErrors().get(0).getCode());

}

try

{

final Cart currentCart = cartService.getCart(httpSession);

cartService.addPhone(phoneDto.getId(), phoneDto.getQuantity(), currentCart);

return ResponseEntity.ok(cartService.getCart(httpSession));

}

catch (final RuntimeException e)

{

final ResponseErrors errors = new ResponseErrors(e.getMessage());

return ResponseEntity.badRequest().body(errors);

}

catch (final Exception e)

{

return ResponseEntity.status(500).build();

}

}

}

**ПРИЛОЖЕНИЕ Б**

***(обязательное)***

**Спецификация программного дипломного проекта**

**ПРИЛОЖЕНИЕ В**

***(обязательное)***

**Ведомость документов**