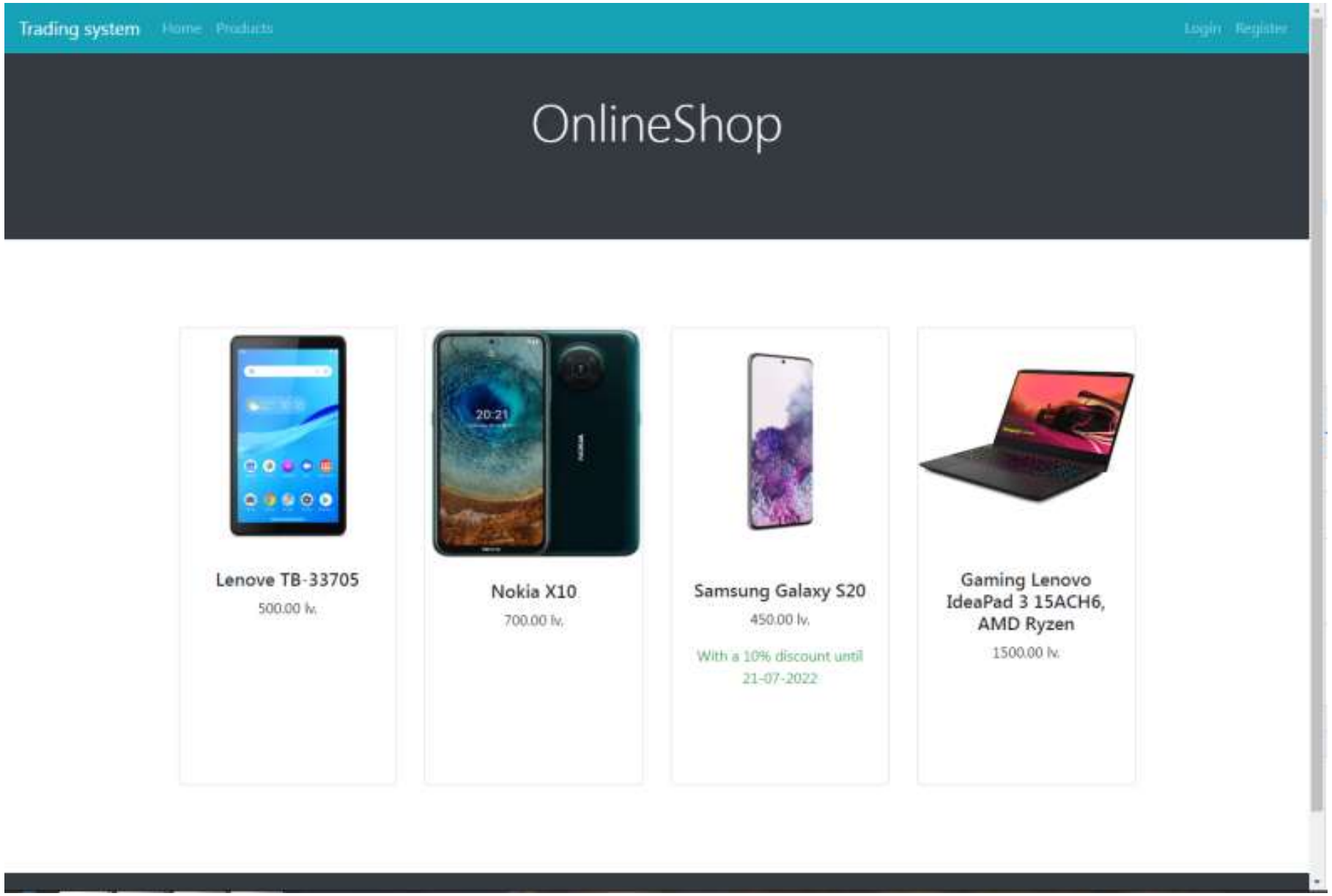
Проект „Онлайн търговия на дребно“

Разработен от: Димитър Найденов, Димана Стоянова и Антонио Сотиров

Използвани технологии

- Java
- SpringBoot
- MySQL
- HTML
- CSS
- JavaScript
- Thymeleaf

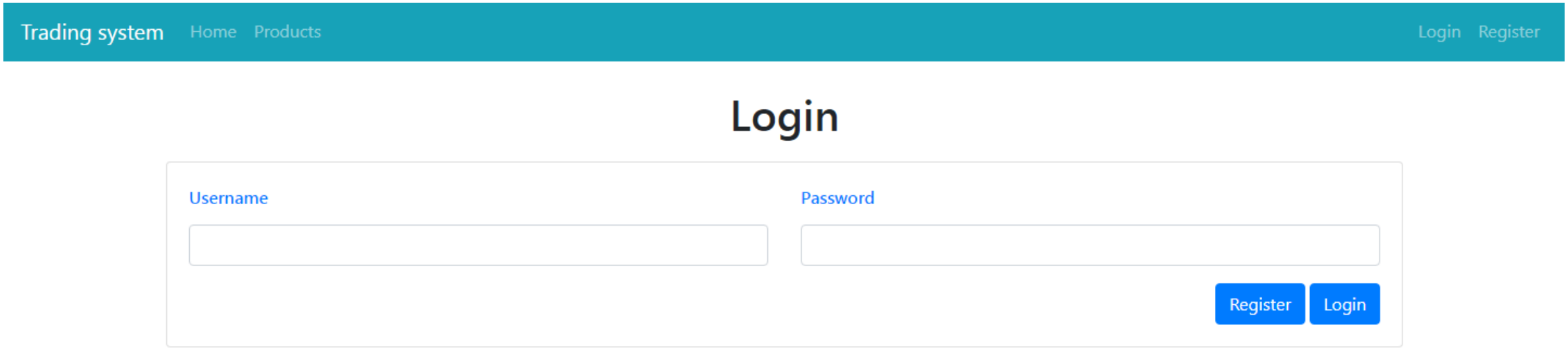
Начална страница



Всеки потребител, който влезе в сайта, може да разгледа началната страница без значение дали е регистриран или не. На нея са изобразени всички продукти, които се предлагат в сайта.

Ако потребителят е регистриран, той има възможност да влезе в акаунта си чрез опцията Login, в противен случай може да си създаде акаунт чрез опцията Register.

Login

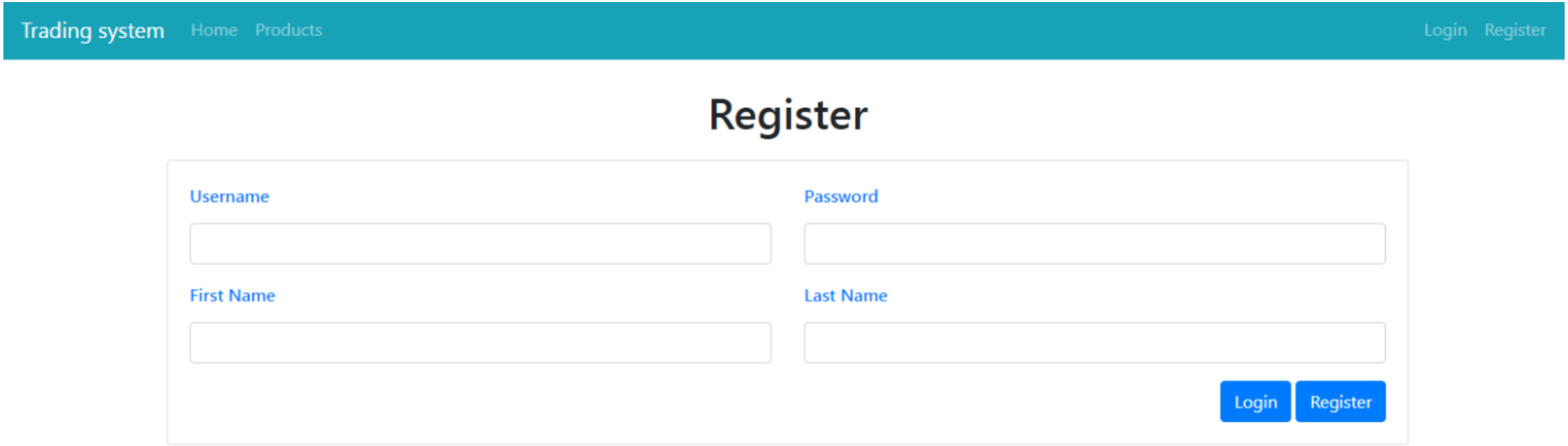


Формата за Login има две задължителни полета Username и Password. Ако полетата са празни, потребителят не може да продължи напред.

```java
@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    Optional<User> user = userRepository.findByUsername(username);
    if (!user.isPresent()) {
        throw new UsernameNotFoundException("Invalid username or password.");
    }
    return new org.springframework.security.core.userdetails.User(user.get().getUsername(),
            user.get().getPassword(),
            mapRolesToAuthorities(user.get().getRole())
           );
}
```

```java
@GetMapping("/login")
public String getLoginPage(Model model, @RequestParam Optional<String> id) {
    if (id.isPresent() && id.get().equals("error")) model.addAttribute( attributeName: "error", attributeValue: true);
    model.addAttribute( attributeValue: "loginRequest");
    return "login";
}
```

Register



Register формата има 4 задължителни полета, без които не може да се продължи.

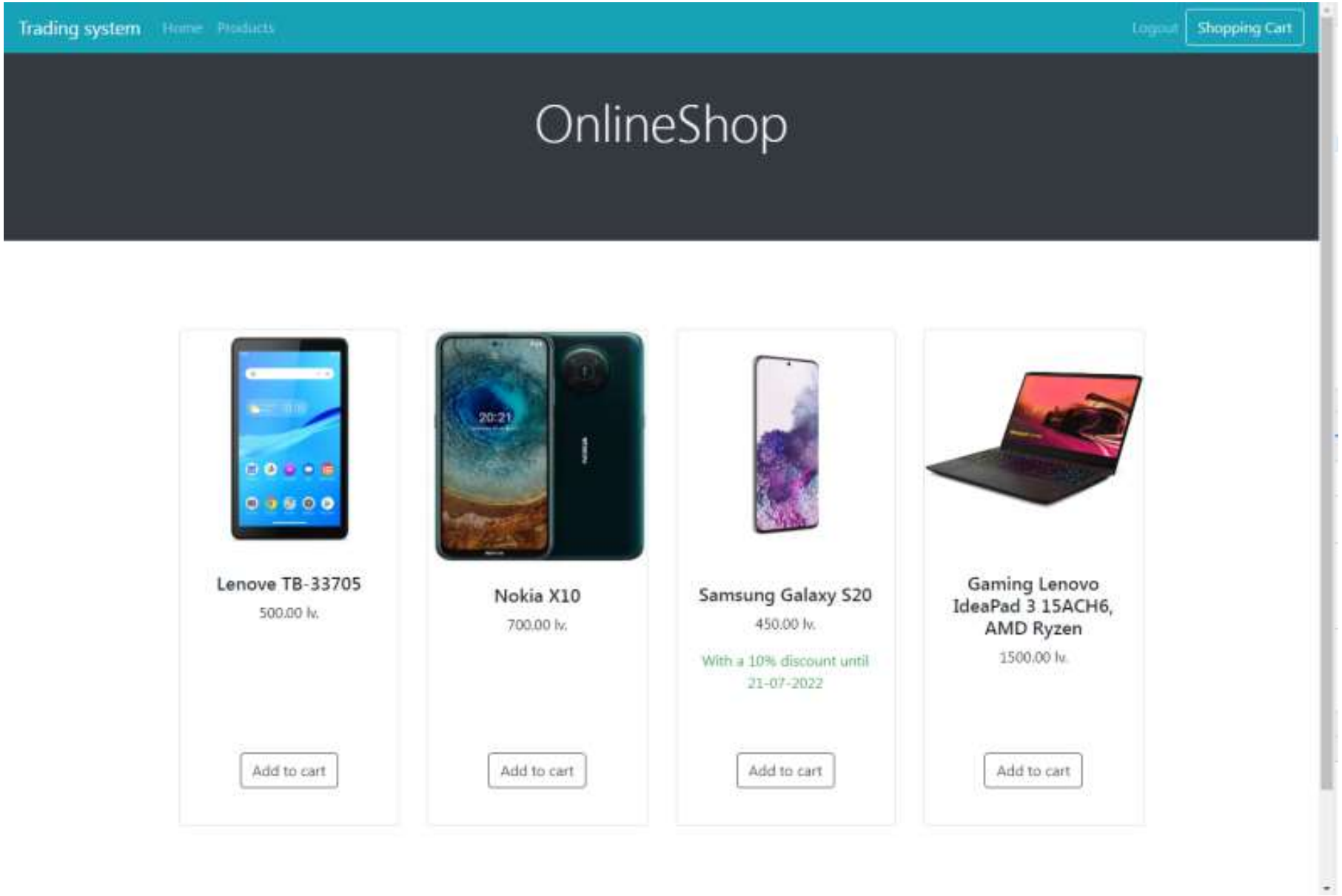
```java
public User registerUser(String firstName, String lastName, String username, String password) {
    boolean userExists = userRepository.findByUsername(username).isPresent();

    if(userExists) {
        return null;
    } else if (firstName == null || lastName == null || username == null || password == null) {
        return null;
    } else {
        User user = new User();
        user.setFirstName(firstName);
        user.setLastName(lastName);
        user.setUsername(username);
        user.setPassword(passwordEncoder.encode(password));
        user.setRole(roleRepository.findByName("USER"));
        return userRepository.save(user);
    }
}
public User auth(String username,String password){
    return userRepository.findByUsernameAndPassword(username, password).orElse( other: null);
}
```

Функции на различните потребители

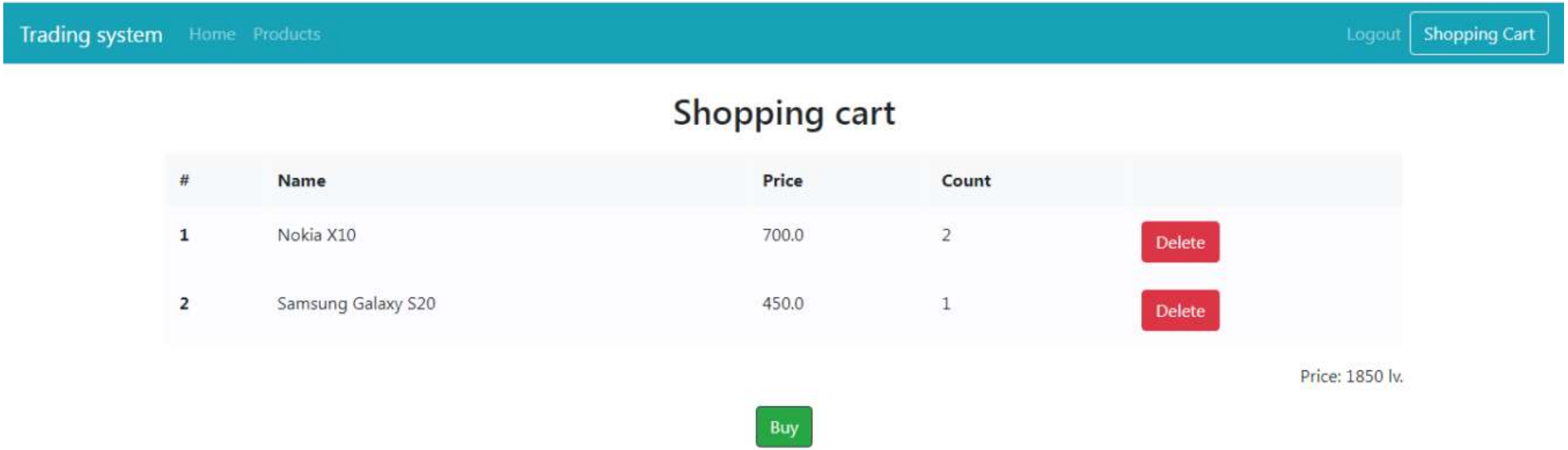
Всеки потребител има достъп до различни функционалности. В зависимост от ролята на потребителя се зарежда различна страница.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
            .antMatchers( ...antPatterns: "/cart").authenticated()
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/statistics").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/users").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/editUser").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.POST,  ...antPatterns: "/editUser").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/store").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.POST,  ...antPatterns: "/store").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/deleteUser").hasAuthority("ADMIN")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/suppliers").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/editSupplier").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/editProduct").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.POST,  ...antPatterns: "/editSupplier").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.POST,  ...antPatterns: "/editProduct").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/addProduct").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.POST,  ...antPatterns: "/addProduct").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/deleteSupplier").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
            .antMatchers(HttpMethod.GET,  ...antPatterns: "/deleteProduct").hasAnyAuthority( ...authorities: "ADMIN", "MODERATOR")
```

User interface



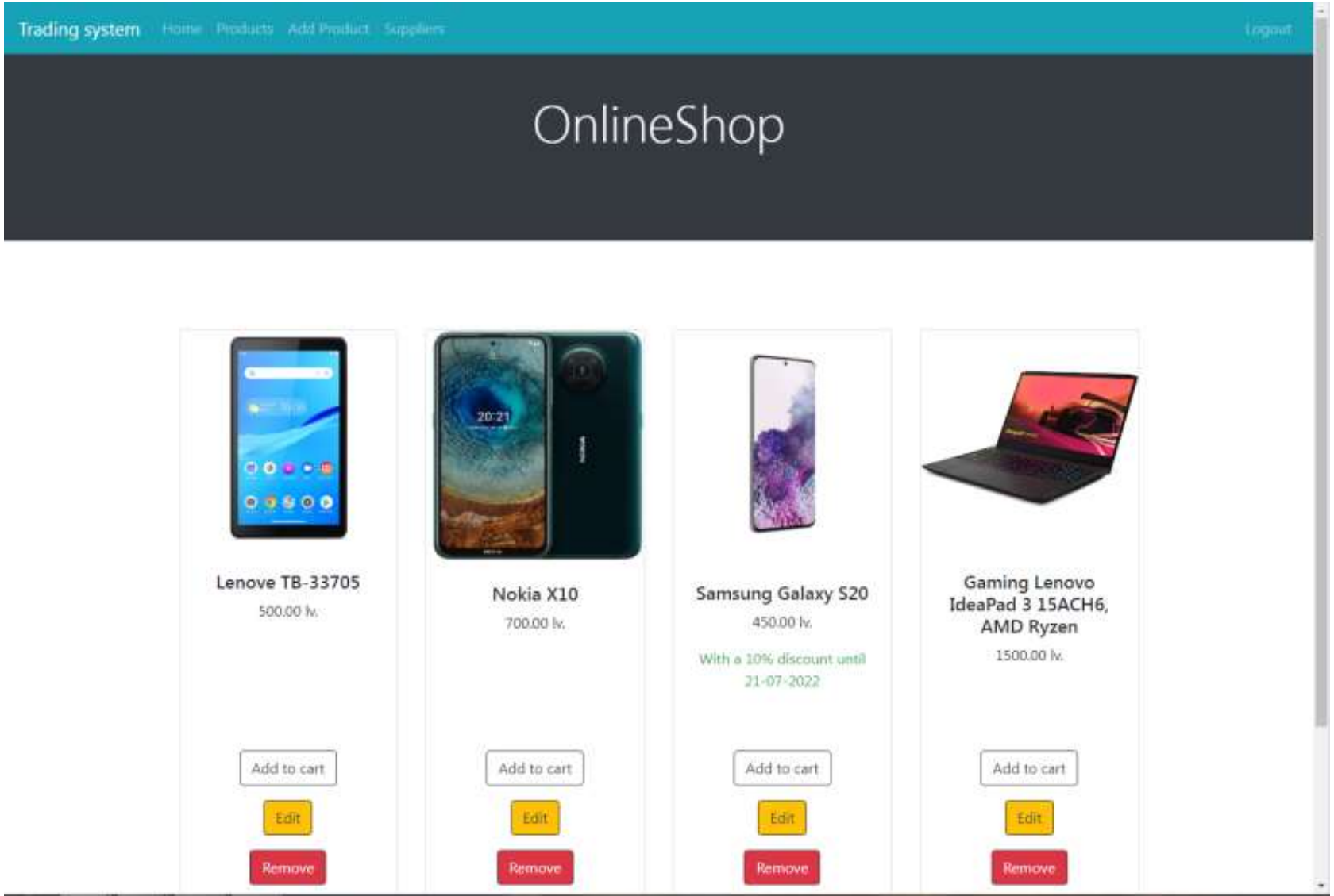
Обикновеният потребител може да разглежда продуктите в магазина, да добавя продукти в количката и да излезе от системата чрез Logout бутона. Може също да отвори количката.



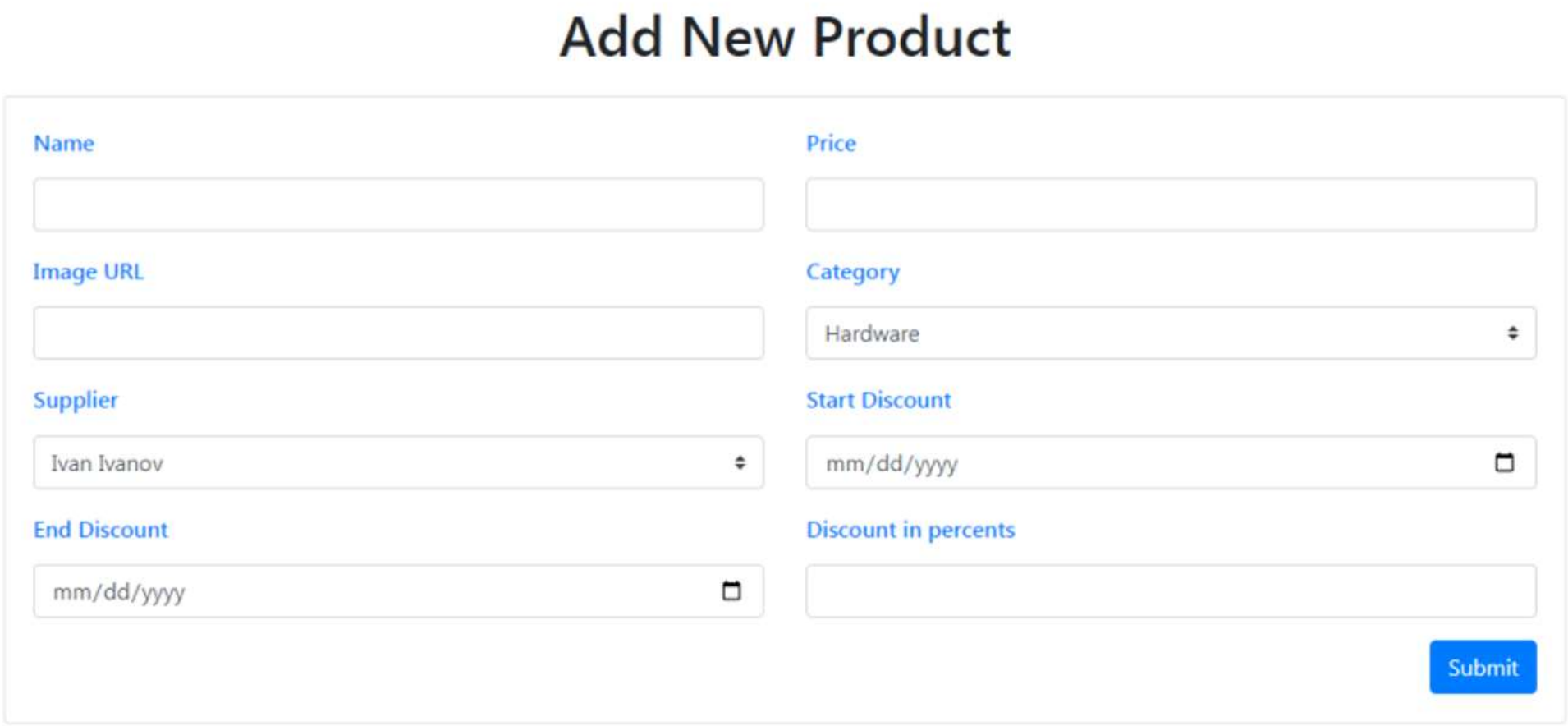
```java
@GetMapping("/cart")
public String showCart(Model model,@RequestParam Optional<String> id)
{
    if (id.isPresent() && id.get().equals("success"))
    model.addAttribute( attributeName: "success", attributeValue: true);
    return "shoppingcart";
}

@PostMapping("/buy")
public String buy(@RequestParam(value="ids[]") Integer[] ids, @RequestParam(value="counts[]") Integer[] counts, Model model)
{
    User user = userService.findByUsername(((UserDetails)SecurityContextHolder.getContext().getAuthentication().getPrincipal()).getUsername());

    for (int i = 0; i < ids.length; i++) {
        BoughtProducts boughtProducts = new BoughtProducts();
        Product product = productService.findById(ids[i]);
        boughtProducts.setUser(user);
        boughtProducts.setProduct(product);
        boughtProducts.setCount(counts[i]);
        boughtProductsService.addBoughtProduct(boughtProducts);
    }

    return "shoppingcart";
}
```

Модератор

Модераторът може да добавя, редактира и изтрива продукти и доставчици.



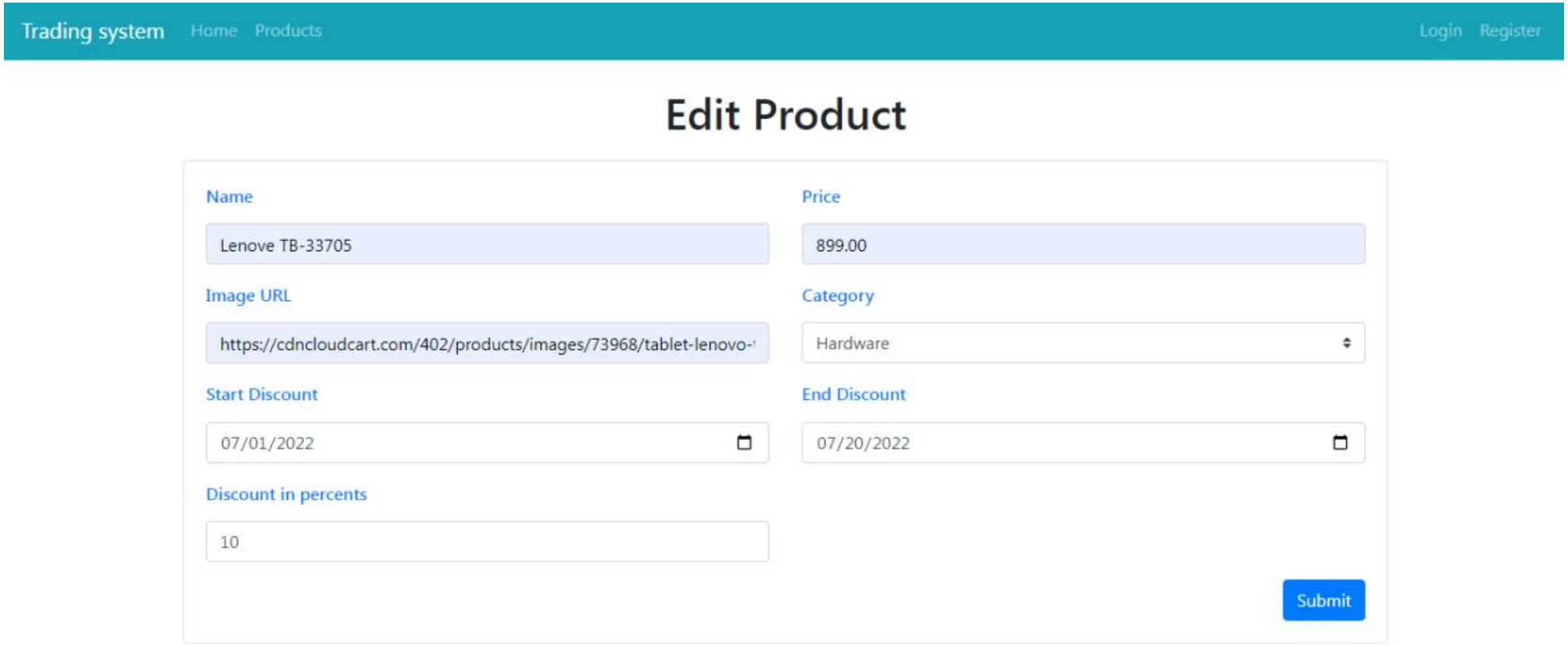
Добавяне на продукт



```java
@GetMapping("/addProduct")
public String getAddProductPage(Model model)
{
    model.addAttribute( attributeName: "suppliers",supplierService.findAll());
    return "addProduct";
}

@PostMapping("/addProduct")
public String addProduct(@ModelAttribute Product product, @RequestParam String supplierId)
{
    product.setSupplier(supplierService.findById(Integer.parseInt(supplierId)));
    productService.addProduct(product);
    return "redirect:/";
}
```

```java
@Transactional
public void addProduct(Product product){

    productRepository.save(product);

}
```

Редактиране на продукт

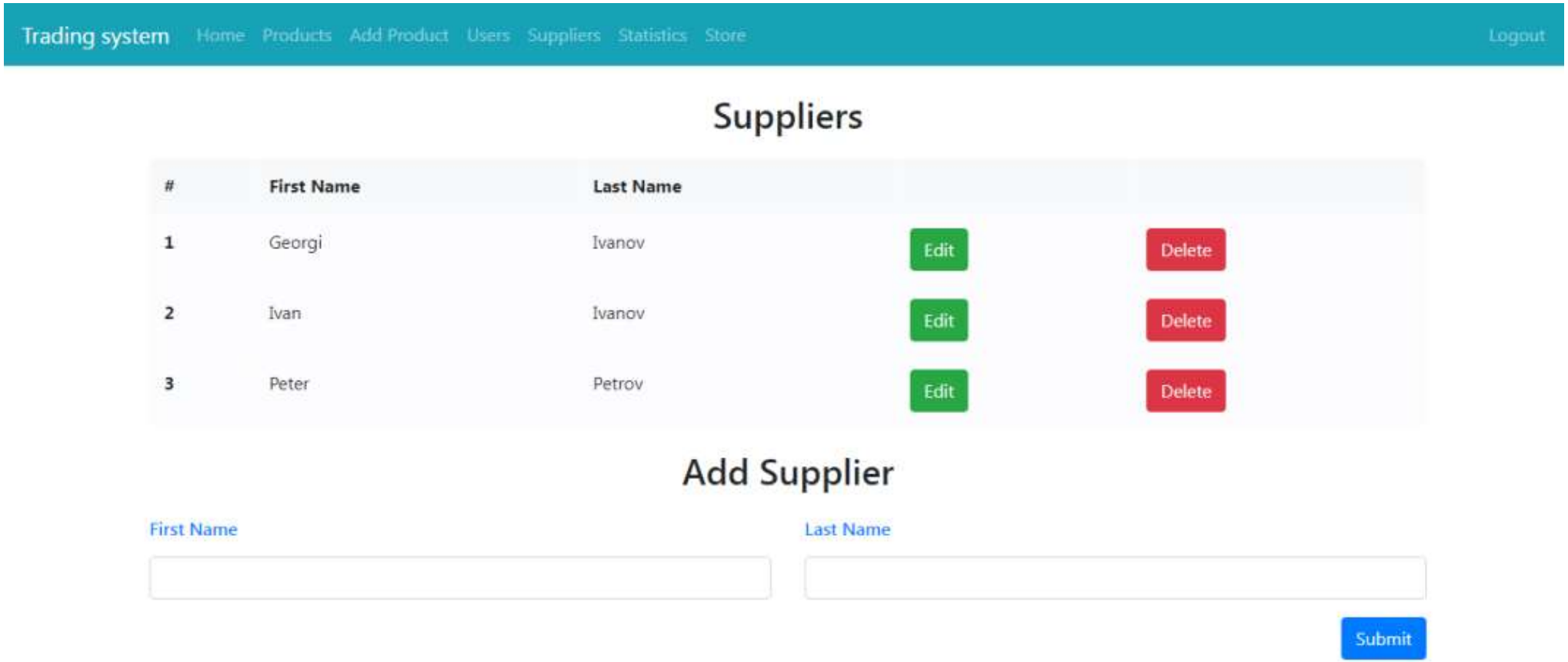


```java
@Transactional
public void updateProduct(int id, Product productDTO) {

    Product product = productRepository.findById(id);

    if (productDTO.getName() != null ) {
        product.setName(productDTO.getName());
    }

    if (productDTO.getPrice() != 0.) {
        product.setPrice(productDTO.getPrice());
    }

    if (productDTO.getImageURL() != null) {
        product.setImageURL(productDTO.getImageURL());
    }

    if (productDTO.getCategory() != null) {
        product.setCategory(productDTO.getCategory());
    }

    if (productDTO.getSupplier() != null) {
        product.setSupplier(productDTO.getSupplier());
    }

    if (productDTO.getStartDiscount() != null) {
        product.setStartDiscount(productDTO.getStartDiscount());
    }

    if (productDTO.getEndDiscount() != null) {
        product.setEndDiscount(productDTO.getEndDiscount());
    }

    if (productDTO.getDiscount() != 0.) {
        product.setDiscount(String.valueOf(productDTO.getDiscount()));
    }

    productRepository.save(product);
}
```

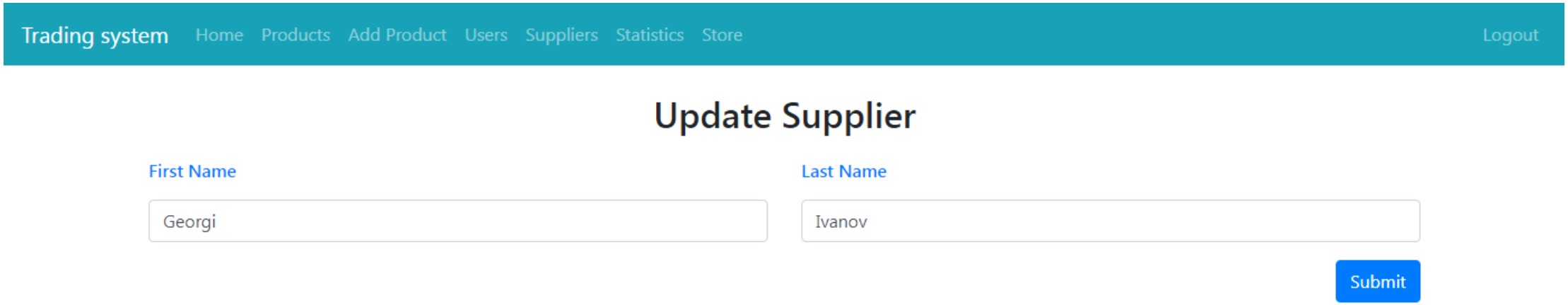
```java
@Transactional
public void deleteProduct(Product product) {
    product.setDeleted(true);
    productRepository.save(product);
      productRepository.delete(product);
}
```

```java
@GetMapping("/editProduct")
public String getEditProductPage(@RequestParam String id,Model model)
{
    model.addAttribute( attributeName: "product",productService.findById(Integer.parseInt(id)));
    model.addAttribute( attributeName: "suppliers",supplierService.findAll());
    return "updateProduct";
}

@PostMapping("/editProduct")
public String editProduct(@RequestParam String id, @ModelAttribute Product product, @RequestParam String supplierId)
{
    product.setSupplier(supplierService.findById(Integer.parseInt(supplierId)));
    productService.updateProduct(product.getId(),product);
    return "redirect:/";
}

@GetMapping("/deleteProduct")
public String deleteProduct(@RequestParam String id) {
    productService.deleteProduct(productService.findById(Integer.parseInt(id)));
    return "redirect:/";
}
```

Доставчици


Trading system
Home
Products
Add Product
Users
Suppliers
Statistics
Store
Logout
Suppliers
First Name Last Name
1 Georgi Ivanov Edit Delete
2 Ivan Ivanov Edit Delete
3 Peter Petrov Edit Delete
Add Supplier
First Name
Last Name
Submit

Редактиране на доставчици

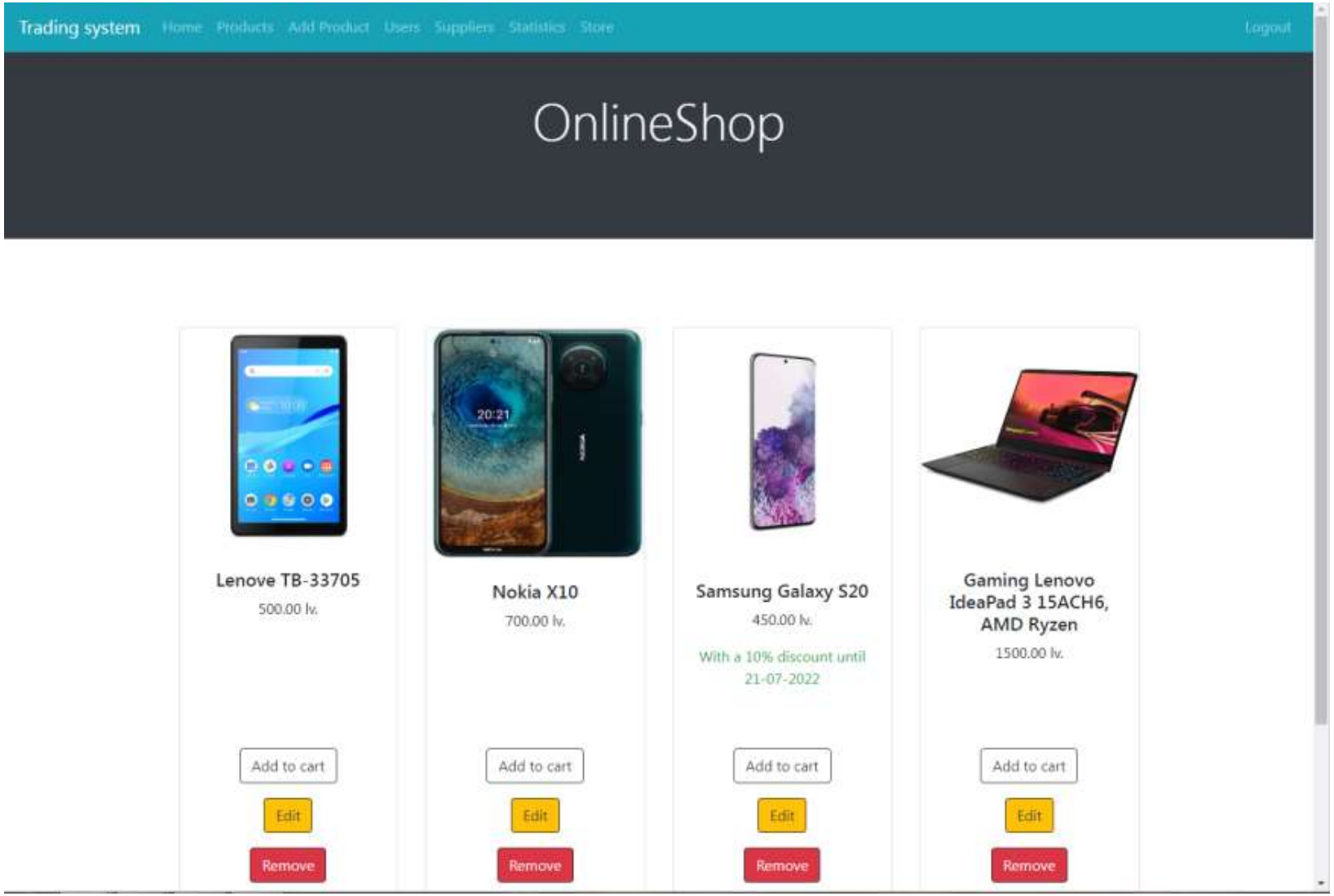


```java
@Transactional
public void updateSupplier(int id, Supplier supplierDTO) {

    Supplier supplier = supplierRepository.findById(id);

    if (supplierDTO.getFirstName() != null ) {
        supplier.setFirstName(supplierDTO.getFirstName());
    }

    if (supplierDTO.getLastName() != null ) {
        supplier.setLastName(supplierDTO.getLastName());
    }

    supplierRepository.save(supplier);
}

1 usage
@Transactional
public void deleteSupplier(Supplier supplier) { supplierRepository.delete(supplier); }
```

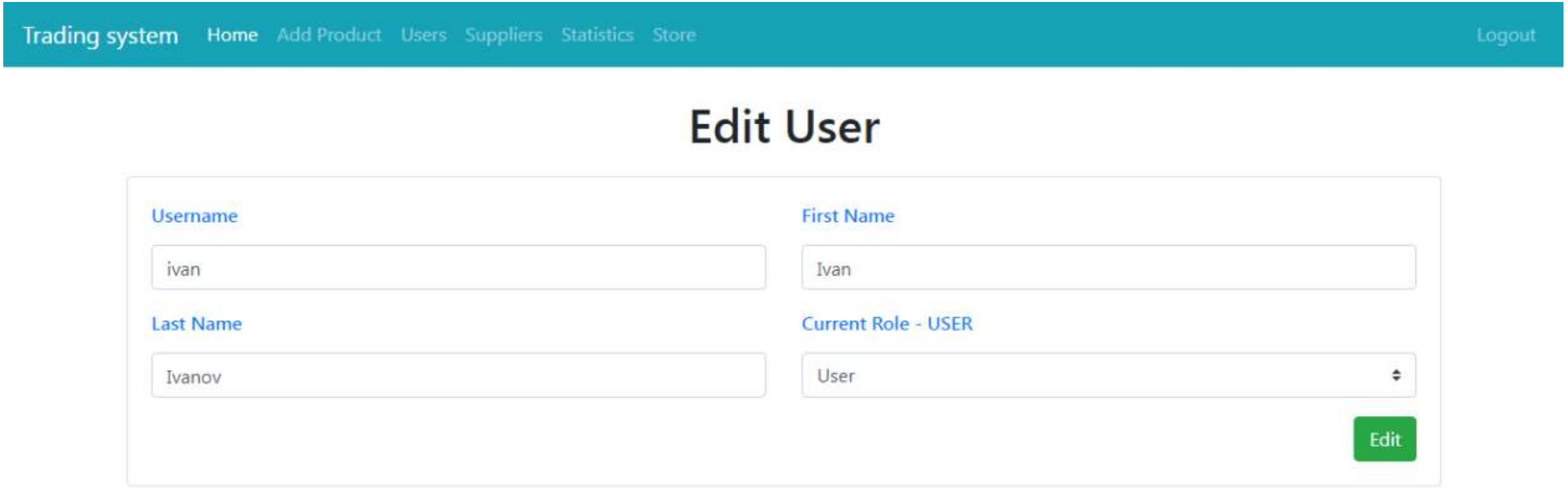
```java
@PostMapping("/addSupplier")
public String addSupplier(@ModelAttribute Supplier supplier)
{
    supplierService.addSupplier(supplier);
    return "redirect:/suppliers";
}

@GetMapping("/editSupplier")
public String editSupplier(@RequestParam String id, Model model) {
    model.addAttribute( attributeName: "supplier",supplierService.findById(Integer.parseInt(id)));
    return "updateSupplier";
}

@PostMapping("/editSupplier")
public String editSupplier(@ModelAttribute Supplier supplier, @RequestParam String id) {
    supplierService.updateSupplier(Integer.parseInt(id), supplier);
    return "redirect:/suppliers";
}

@GetMapping("/deleteSupplier")
public String deleteSupplier(@RequestParam String id) {
    supplierService.deleteSupplier(supplierService.findById(Integer.parseInt(id)));
    return "redirect:/suppliers";
}
```

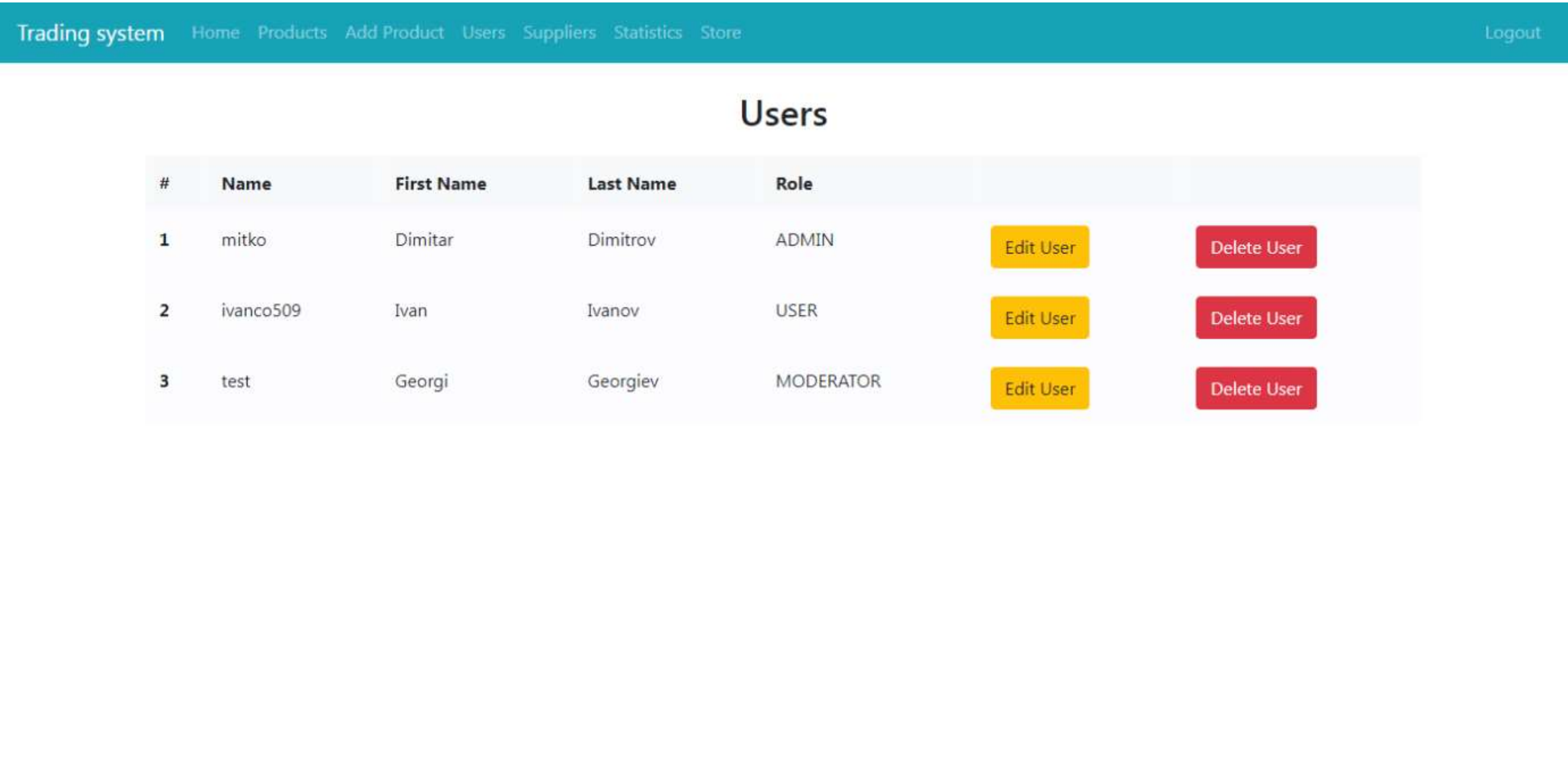
Администратор

Администраторът има възможност да добавя, редактира и изтрива цялата информация в системата. Част от функциите се препокриват с тези на модератора.



Редактиране на потребители




Trading system
Home
Products
Add Product
Users
Suppliers
Statistics
Store
Logout
Users
Name First Name Last Name Role
1 mitko Dimitar Dimitrov ADMIN Edit User Delete User
2 ivanco509 Ivan Ivanov USER Edit User Delete User
3 test Georgi Georgiev MODERATOR Edit User Delete User

```java
@Transactional
public void updateUser(int id, User userDTO) {

    User user = userRepository.findById(id);

    if (userDTO.getFirstName() != null) {
        user.setFirstName(userDTO.getFirstName());
    }

    if (userDTO.getLastName() != null) {
        user.setLastName(userDTO.getLastName());
    }

    if (userDTO.getUsername() != null) {
        user.setUsername(userDTO.getUsername());
    }

    if (userDTO.getRole() != null) {
        user.setRole(userDTO.getRole());
    }

    userRepository.save(user);
}
```

```java
@GetMapping("/users")
public String showUsers(Model model) {

    List<User> users = userService.findAll();
    model.addAttribute( attributeName: "users", users);
    return "users";
}

@GetMapping("/editUser")
public String editUser(@RequestParam String id, Model model) {

    User user = userService.findById(Integer.parseInt(id));
    model.addAttribute( attributeName: "user", user);
    return "updateUser";
}

@PostMapping("/editUser")
public String editContact(@ModelAttribute User user, @RequestParam("roleName") String roleName, @RequestParam String id) {
    user.setRole(roleService.findByName(roleName));
    userService.updateUser(Integer.parseInt(id),user);
    return "redirect:/users";
}

@GetMapping("/deleteUser")
public String deleteUser(@RequestParam String id) {
    userService.removeUser(Integer.parseInt(id));
    return "redirect:/users";
}

@GetMapping("/users/new")
public String CreateUser(Model model) {
    User user = new User();
    model.addAttribute( attributeName: "user", user);
    return "users";
}
```

Статистика


Trading system
Sold Products
Statistics
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
Lenove TB-33705
Nokia X10
Samsung Galaxy S20
test
Sold Products By Category
Statistics
3.0
2.5
2.0
1.5
1.0
0.5
0
HARDWARE
FURNITURE
OTHER

```java
public  List<BoughtProduct> getBoughtProductsByCount(){

    List<Object[]> list = boughtProductsRepository.getBoughtProductsByCount();
    List<BoughtProduct> boughtProducts = new ArrayList<>();
    for (Object[] obj : list) {
        BoughtProduct boughtProduct = new BoughtProduct();
        boughtProduct.setName(productService.findById(Integer.parseInt(obj[0].toString())).getName());
        boughtProduct.setCount(Integer.parseInt(obj[1].toString()));
        boughtProducts.add(boughtProduct);
    }
    return boughtProducts;
}

1 usage
public  List<BoughtProduct> getBoughtProductsByCategory(){

    List<Object[]> list = boughtProductsRepository.getBoughtProductsByCategory();
    List<BoughtProduct> boughtProducts = new ArrayList<>();
    for (Object[] obj : list) {
        BoughtProduct boughtProduct = new BoughtProduct();
        boughtProduct.setName(obj[0].toString());
        boughtProduct.setCount(Integer.parseInt(obj[1].toString()));
        boughtProducts.add(boughtProduct);
    }
    return boughtProducts;
}
```