МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
(РУТ(МИИТ)

Кафедра «Цифровые технологии управления транспортными процессами»

ПРИЛОЖЕНИЕ К КУРСОВОЙ РАБОТЕ «ИНТЕРНЕТ МАГАЗИН “КАТАЛОГ АВТОМОБИЛЕЙ”»

Направление: 09.03.01 Информатика и вычислительная техника

Профиль: Технологии разработки программного обеспечения

Выполнил:
студент группы УВП-312
Букин Д.П.

Проверил:
старший преподаватель
Заманов Е.А

Москва 2024 г.

СОДЕРЖАНИЕ

ПРИЛОЖЕНИЕ А – MODELS

```
@MappedSuperclass
public abstract class BaseEntity {
    private String id;

    public BaseEntity() {
    }

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }


}


@MappedSuperclass
public abstract class BaseEntityCreatedModified extends BaseEntity{

    private LocalDateTime created; //Дата и время создания
    private LocalDateTime modified; //Дата и время изменения

    public BaseEntityCreatedModified() {
    }
    @Column(name = "created")
    public LocalDateTime getCreated() {
        return created;
    }

    public void setCreated(LocalDateTime created) {
        this.created = created;
    }
    @Column(name = "modified")
    public LocalDateTime getModified() {
        return modified;
    }

    public void setModified(LocalDateTime modified) {
        this.modified = modified;
    }
}

@Entity
@Table(name = "brands")
public class Brand extends BaseEntityCreatedModified {
    private String name;
    private Set<Model> model;

    public Brand() {
    }

    @Column(name = "name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "brand")
    public Set<Model> getModel() {
        return model;
    }

    public void setModel(Set<Model> model) {
        this.model = model;
    }
}


@Entity
@Table(name = "models")
public class Model extends BaseEntityCreatedModified {
    private String name;
    private Category category;
    private String imageUrl;
    private int startYear;
    private int endYear;
    private Set<Offer> offers;
    private Brand brand;
    public Model() {
    }

    @Column(name = "name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    @Column(name = "category")
    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }
    @Column(name = "image_url")
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
    @Column(name = "start_year")
    public int getStartYear() {
        return startYear;
    }

    public void setStartYear(int startYear) {
        this.startYear = startYear;
    }
    @Column(name = "end_year")
    public int getEndYear() {
        return endYear;
    }
```

```
    public void setEndYear(int endYear) {
        this.endYear = endYear;
    }
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "model")
    public Set<Offer> getOffers() {
        return offers;
    }

    public void setOffers(Set<Offer> offers) {
        this.offers = offers;
    }
    @ManyToOne(optional = false)
    @JoinColumn(name = "brand_id", referencedColumnName = "id",
nullable=false)
    public Brand getBrand() {
        return brand;
    }

    public void setBrand(Brand brand) {
        this.brand = brand;
    }
}


@Entity
@Table(name = "offers")
public class Offer extends BaseEntityCreatedModified {
    private String description;
    private Engine engine;
    private String image_url;
    private int mileage;
    private BigDecimal price;
    private Transmission transmission;
    private int years;
    private Model model;
    private User seller;

    public Offer() {
    }
    @Column(name = "description")
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
    @Column(name = "engine")
    public Engine getEngine() {
        return engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
    }
    @Column(name = "image_url")
    public String getImage_url() {
        return image_url;
    }

    public void setImage_url(String image_url) {
        this.image_url = image_url;
    }
    @Column(name = "mileage")
```

```
    public int getMileage() {
        return mileage;
    }

    public void setMileage(int mileage) {
        this.mileage = mileage;
    }

    @Column(name = "price")
    public BigDecimal getPrice() {
        return price;
    }

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

    @Column(name = "transmission")
    public Transmission getTransmission() {
        return transmission;
    }

    public void setTransmission(Transmission transmission) {
        this.transmission = transmission;
    }
    @Column(name = "year")
    public int getYears() {
        return years;
    }

    public void setYears(int years) {
        this.years = years;
    }

    @ManyToOne(optional = false)
    @JoinColumn(name = "model_id", referencedColumnName = "id",
nullable=false)
    public Model getModel() {
        return model;
    }

    public void setModel(Model model) {
        this.model = model;
    }
    @ManyToOne(optional = false)
    @JoinColumn(name = "seller_id", referencedColumnName = "id",
nullable=false)
    public User getSeller() {
        return seller;
    }

    public void setSeller(User seller) {
        this.seller = seller;
    }
}


@Entity
@Table(name = "users")
public class User extends BaseEntityCreatedModified implements Serializable {
    private String username;
    private String password;
    private String firstName;
    private String lastName;
```

```
    private boolean isActive;
    private String imageUrl;
    private Set<Offer> offers;
    private UserRole role;

    public User() {
    }

    @Column(name = "username")
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
    @Column(name = "password")
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @Column(name = "first_name")
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    @Column(name = "last_name")
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String last_name) {
        this.lastName = last_name;
    }
    @Column(name = "image_url")
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    @Column(name = "is_active", nullable = false)
    public boolean getIsActive() {
        return isActive;
    }
    public void setIsActive(boolean active) {
        isActive = active;
    }

    @ManyToOne(optional = false)
    @JoinColumn(name = "role_id", referencedColumnName = "id",
nullable=false)
    public UserRole getRole() {
        return role;
    }

    public void setRole(UserRole role) {
        this.role = role;
```

```
    }

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "seller")
    public Set<Offer> getOffers() {
        return offers;
    }

    public void setOffers(Set<Offer> offers) {
        this.offers = offers;
    }
}


@Entity
@Table(name = "roles")
public class UserRole extends BaseEntity {
    private Role role;
    private Set<User> user;

    public UserRole() {
    }

    @Column(name = "role", nullable = false)
    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "role")
    public Set<User> getUser() {
        return user;
    }

    public void setUser(Set<User> user) {
        this.user = user;
    }
}


public enum Category {
    CAR(0), BUSS(1), TRUCK(2), MOTORCYCLE(3);
    Category(int i) {
    }
}


public enum Engine {
    GASOLINE(0), DIESEL(1), ELECTRIC(2), HYBRID(3);

    Engine(int i) {
    }
}


public enum Role {
   USER, MODERATOR, ADMIN
}
```

```java
public enum Transmission {
    MANUAL(0), AUTOMATIC(1), ROBOT(2), VARIATOR(3);

    Transmission(int i) {
    }
}
```

ПРИЛОЖЕНИЕ Б – DTOs

```
public class BrandDto{
    @UniqueBrandName
    private String name;

    public BrandDto() {
    }
    @NotBlank(message = "The name cannot be empty")
    @Size(min = 2, message = "name must be more than two characters!")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}


public class ModelDto{
    private String name;
    private Category category;
    private String imageUrl;
    private int startYear;
    private int endYear;
    private String brand;
    public ModelDto() {
    }

    @NotEmpty(message = "Model name must not be null or empty!")
    @Size(min = 2, message = "name must be more than two characters!")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    @NotNull(message = "Category must be selected")
    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    @NotEmpty(message = "The imageUrl cannot be empty")
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    @Min(value = 1900, message = "The release start date cannot be earlier
than 1900")
    @Max(value = 2024, message = "The release start date cannot be in the
future")
    public int getStartYear() {
        return startYear;
    }
```

```
    public void setStartYear(int startYear) {
        this.startYear = startYear;
    }

    @Min(value = 1900, message = "The date cannot be earlier than 1900")
    @Max(value = 2024, message = "The date cannot be in the future")
    public int getEndYear() {
        return endYear;
    }

    public void setEndYear(int endYear) {
        this.endYear = endYear;
    }
    @NotBlank(message = "Brand must be selected")
    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }
}


public class OfferDto{
    private String id;
    private String description;
    private Engine engine;
    private String image_url;
    private int mileage;
    private BigDecimal price;
    private Transmission transmission;
    private int years;
    private String model;
    private String seller;

    public OfferDto() {
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @Length(min = 10, message = "Description must be more than ten
characters!")
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
    @NotNull(message = "Engine must be selected")
    public Engine getEngine() {
        return engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
```

```
    }
    @NotEmpty(message = "The imageUrl cannot be empty")
    public String getImage_url() {
        return image_url;
    }

    public void setImage_url(String image_url) {
        this.image_url = image_url;
    }


    @Min(value = 1, message = "The mileage cannot be less than 1")
    public int getMileage() {
        return mileage;
    }

    public void setMileage(int mileage) {
        this.mileage = mileage;
    }

    @NotNull(message = "The Price cannot be empty")
    @DecimalMin(value = "1", message = "The cost cannot be less than 1")
    public BigDecimal getPrice() {
        return price;
    }

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

    @NotNull(message = "Transmission must be selected")
    public Transmission getTransmission() {
        return transmission;
    }

    public void setTransmission(Transmission transmission) {
        this.transmission = transmission;
    }

    @Min(value = 1900, message = "The release date cannot be earlier than
1900")
    @Max(value = 2024, message = "The release date cannot be in the future")
    public int getYears() {
        return years;
    }

    public void setYears(int years) {
        this.years = years;
    }

    @NotBlank(message = "Model must be selected")
    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getSeller() {
        return seller;
    }

    public void setSeller(String seller) {
```

```
        this.seller = seller;
    }
}


public class UserDto{
    @UniqueUsername
    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private boolean isActive;
    private String imageUrl;
    private Role role;

    public UserDto() {
    }

    @NotEmpty(message = "The username cannot be empty")
    @Length(min = 2, message = "Username must be more than two characters and
not use special characters!")
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @NotEmpty(message = "The password cannot be empty")
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
    @NotNull
    @NotEmpty(message = "The firstName cannot be empty")
    @Length(min = 2, message = "The firstName must be more than two
characters and not use special characters!")
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    @NotNull
    @NotEmpty(message = "The lastName cannot be empty")
    @Length(min = 2, message = "The lastName must be more than two characters
and not use special characters!")
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public boolean getIsActive() {
        return isActive;
    }
```

```
    public void setIsActive(boolean active) {
        isActive = active;
    }
    @NotEmpty(message = "The imageUrl cannot be empty")
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
    @NotNull(message = "Role must be selected")
    public Role getRole() {
        return role;
    }
    public void setRole(Role role) {
        this.role = role;
    }
}


public class UserRegistrationDto {
    @UniqueUsername
    private String username;
    private String firstName;
    private String lastName;
    private String imageUrl;
    private String password;
    private String confirmPassword;

    @NotEmpty(message = "The username cannot be empty")
    @Length(min = 2, message = "Username must be more than two characters and
not use special characters!")
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @NotEmpty(message = "The password cannot be empty")
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
    @NotNull
    @NotEmpty(message = "The firstName cannot be empty")
    @Length(min = 2, message = "The firstName must be more than two
characters and not use special characters!")
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    @NotNull
    @NotEmpty(message = "The lastName cannot be empty")
    @Length(min = 2, message = "The lastName must be more than two characters
and not use special characters!")
```

```java
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    @NotEmpty(message = "Confirm Password cannot be null or empty!")
    @Size(min = 5, max = 20)
    public String getConfirmPassword() {
        return confirmPassword;
    }

    public void setConfirmPassword(String confirmPassword) {
        this.confirmPassword = confirmPassword;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
}


public class UserRoleDto{
    private Role role;

    public UserRoleDto() {
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
}
```

ПРИЛОЖЕНИЕ В – VIEW

```
public class ShowBrandVM{
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}


public class ShowModelVM {
    private String name;
    private Category category;
    private String imageUrl;
    private int startYear;
    private int endYear;
    private ShowBrandVM brand;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    public int getStartYear() {
        return startYear;
    }

    public void setStartYear(int startYear) {
        this.startYear = startYear;
    }

    public int getEndYear() {
        return endYear;
    }

    public void setEndYear(int endYear) {
        this.endYear = endYear;
    }

    public ShowBrandVM getBrand() {
        return brand;
    }
```

```
    public void setBrand(ShowBrandVM brand) {
        this.brand = brand;
    }
}

public class ShowOfferVM {
    private String id;
    private String description;
    private Engine engine;
    private String image_url;
    private int mileage;
    private BigDecimal price;
    private Transmission transmission;
    private int years;
    private ShowModelVM model;
    private ShowUserVM seller;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Engine getEngine() {
        return engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

    public String getImage_url() {
        return image_url;
    }

    public void setImage_url(String image_url) {
        this.image_url = image_url;
    }

    public int getMileage() {
        return mileage;
    }

    public void setMileage(int mileage) {
        this.mileage = mileage;
    }

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
```

```
    }

    public Transmission getTransmission() {
        return transmission;
    }

    public void setTransmission(Transmission transmission) {
        this.transmission = transmission;
    }

    public int getYears() {
        return years;
    }

    public void setYears(int years) {
        this.years = years;
    }

    public ShowModelVM getModel() {
        return model;
    }

    public void setModel(ShowModelVM model) {
        this.model = model;
    }

    public ShowUserVM getSeller() {
        return seller;
    }

    public void setSeller(ShowUserVM seller) {
        this.seller = seller;
    }
}


public class ShowUserRoleVM {
    private Role role;

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
}


public class ShowUserVM {
    private String username;
    private String firstName;
    private String lastName;
    private String imageUrl;
    private boolean isActive;
    private Role role;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
```

```
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    public boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(boolean active) {
        isActive = active;
    }


    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }
}


public class UserProfileView {
    private String username;
    private String firstName;
    private String lastName;
    private String imageUrl;

    public UserProfileView() {
    }

    public UserProfileView(String username, String firstName, String
lastName, String imageUrl) {
        this.username = username;
        this.firstName = firstName;
        this.lastName = lastName;
        this.imageUrl = imageUrl;
    }

    public String getUsername() {
        return username;
    }
```

```
    public void setUsername(String username) {
        this.username = username;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
}
```

ПРИЛОЖЕНИЕ Г – REPOSITORY

```
@Repository
public interface BrandRepository extends JpaRepository<Brand,String> {
    Optional<Brand> findByName(String name);
}


@Repository
public interface ModelRepository extends JpaRepository<Model, String> {
    Optional<Model> findByName(String name);
}


@Repository
public interface OfferRepository extends JpaRepository<Offer, String> {
    @Query("SELECT o FROM Offer o WHERE o.seller.isActive = true")
    List<Offer> findOffersWithActiveClients();
    @Query("SELECT o FROM Offer o JOIN o.model m JOIN m.brand b WHERE b.name
= :brandName")
    List<Offer> findOffersByBrandName(String brandName);
    @Query("SELECT o FROM Offer o JOIN o.seller u WHERE u.username =
:username")
    List<Offer> findOffersBySellerUsername(String username);
    List<Offer> findAllByOrderByPriceAsc();
    List<Offer> findAllByOrderByPriceDesc();
}


@Repository
public interface UserRepository extends JpaRepository<User, String> {
    Optional<User> findByUsername(String username);
    Optional<User> findById(String id);

}

@Repository
public interface UserRoleRepository extends JpaRepository<UserRole, String> {
    Optional<UserRole> findByRole(Role role);

}
```

ПРИЛОЖЕНИЕ Д – SERVICES

```
public interface BrandService {
    List<ShowBrandVM> getAllBrands();
    void addNewBrand(BrandDto brandDto);
    ShowBrandVM brandDetails(String brandName);
    ShowBrandVM getBrandById(String id);
}


public interface ModelService {
    List<ShowModelVM> getAllModels();
    void addNewModel(ModelDto modelDto);
    ModelDto modelDetails(String modelName);
    List<BrandDto> showBrand();
}


public interface OfferService<ID> {
    List<ShowOfferVM> getAllOffers();
    void addNewOffer(OfferDto OfferDto);
    ShowOfferVM offerDetails(String offerID);
    List<ModelDto> showModel();
    List<OfferDto> showUser();
    List<ShowOfferVM> findOffersByBrandName(String brandName);
    List<ShowOfferVM> findOffersWithActiveClients();
    List<Offer> findOffersBySellerUsername(String username);
    List<Offer> findAllByOrderByPriceAsc();
    List<Offer> findAllByOrderByPriceDesc();
    void deleteOfferById(String id);
    OfferDto getOfferById(String id);
}


public interface UserRoleService{
    List<UserRoleDto> getAllUserRoles();
    void addUserRole(UserRoleDto userRoleDto);
    ShowUserRoleVM userRoleDetails(String userRole);
}


public interface UserService {
    List<ShowUserVM> getAllUsers();
    void addUser(UserDto addUserVM);
    ShowUserVM userDetails(String userName);
    List<UserRoleDto> showRole();
    void deactivateUser(String username);


@Service
public class AppUserDetailsService implements UserDetailsService {
    private UserRepository userRepository;

    public AppUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        return userRepository.findByUsername(username)
                .map(u -> new
org.springframework.security.core.userdetails.User(
                        u.getUsername(),
                        u.getPassword(),
```

```
                        Collections.singletonList(new
SimpleGrantedAuthority("ROLE_" + u.getRole().getRole().name()))
                )).orElseThrow(() -> new UsernameNotFoundException(username +
" was not found!"));
    }
}


@Service
public class AuthService {
    private UserRepository userRepository;
    private UserRoleRepository userRoleRepository;
    private ModelMapper modelMapper;
    private PasswordEncoder passwordEncoder;

    public AuthService(UserRepository userRepository, UserRoleRepository
userRoleRepository, ModelMapper modelMapper, PasswordEncoder passwordEncoder)
{
        this.userRepository = userRepository;
        this.userRoleRepository = userRoleRepository;
        this.modelMapper = modelMapper;
        this.passwordEncoder = passwordEncoder;
    }

    public void register(UserRegistrationDto registrationDTO) {
        if
(!registrationDTO.getPassword().equals(registrationDTO.getConfirmPassword()))
{
            throw new RuntimeException("passwords.match");
        }

        Optional<User> byUserName =
this.userRepository.findByUsername(registrationDTO.getUsername());

        if (byUserName.isPresent()) {
            throw new RuntimeException("username.used");
        }

        var userRole =
userRoleRepository.findByRole(Role.USER).orElseThrow();

        User user = modelMapper.map(registrationDTO, User.class);
        user.setRole(userRole);
        user.setIsActive(true);
        user.setCreated(LocalDateTime.now());
        user.setModified(LocalDateTime.now());
        userRepository.saveAndFlush(user);

        String encodedPassword = passwordEncoder.encode(user.getPassword());
        user.setPassword(encodedPassword);

        this.userRepository.saveAndFlush(user);
    }

    public User getUser(String username) {
        return userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException(username + "
was not found!"));
    }
}


@Service
@EnableCaching
```

```
public class BrandServiceImpl implements BrandService {

    private BrandRepository brandRepository;
    private final ValidationUtil validationUtil;
    private ModelMapper modelMapper;

    @Autowired
    public BrandServiceImpl(BrandRepository brandRepository, ValidationUtil
validationUtil, ModelMapper modelMapper) {
        this.brandRepository = brandRepository;
        this.validationUtil = validationUtil;
        this.modelMapper = modelMapper;
    }

    @Override
    @Cacheable("brands")
    public List<ShowBrandVM> getAllBrands() {
        return this.brandRepository.findAll().stream().map(brand ->
modelMapper.map(brand, ShowBrandVM.class)).collect(Collectors.toList());
    }

    @Override
    @CacheEvict(cacheNames = "brands", allEntries = true)
    public void addNewBrand(BrandDto brandDto) {
        if (!this.validationUtil.isValid(brandDto)) {

            this.validationUtil
                    .violations(brandDto)
                    .stream()
                    .map(ConstraintViolation::getMessage)
                    .forEach(System.out::println);

            throw new IllegalArgumentException("Illegal arguments!");
        }

        Brand brand = this.modelMapper.map(brandDto, Brand.class);
        brand.setCreated(LocalDateTime.now());
        brand.setModified(LocalDateTime.now());

        this.brandRepository.saveAndFlush(brand);

    }
    @Override
    public ShowBrandVM brandDetails(String brandName) {
        return
modelMapper.map(brandRepository.findByName(brandName).orElse(null),
ShowBrandVM.class);
    }

    @Override
    public ShowBrandVM getBrandById(String id) {
        return modelMapper.map(brandRepository.findById(id),
ShowBrandVM.class);
    }
}


@Service
@EnableCaching
public class ModelServiceImpl implements ModelService {
    private ModelRepository modelRepository;
    private BrandRepository brandRepository;
    private final ValidationUtil validationUtil;
    private ModelMapper modelMapper;
```

```
    @Autowired
    public ModelServiceImpl(ModelRepository modelRepository, BrandRepository
brandRepository, ValidationUtil validationUtil, ModelMapper modelMapper) {
        this.modelRepository = modelRepository;
        this.brandRepository = brandRepository;
        this.validationUtil = validationUtil;
        this.modelMapper = modelMapper;
    }

    @Override
    @Cacheable("models")
    public List<ShowModelVM> getAllModels() {
        return modelRepository.findAll().stream().map(models ->
modelMapper.map(models, ShowModelVM.class)).collect(Collectors.toList());
    }

    @Override
    @CacheEvict(cacheNames = "models", allEntries = true)
    public void addNewModel(ModelDto modelDto) {
        Model model = modelMapper.map(modelDto, Model.class);

model.setBrand(brandRepository.findByName(modelDto.getBrand()).orElseThrow())
;
        model.setCreated(LocalDateTime.now());
        model.setModified(LocalDateTime.now());

        modelRepository.saveAndFlush(model);
    }

    @Override
    public ModelDto modelDetails(String modelName) {
        return
modelMapper.map(modelRepository.findByName(modelName).orElseThrow(),
ModelDto.class);
    }

    @Override
    public List<BrandDto> showBrand() {
        return brandRepository.findAll().stream().map(brand ->
modelMapper.map(brand, BrandDto.class)).collect(Collectors.toList());
    }

}


@Service
@EnableCaching
public class OfferServiceImpl implements OfferService {
    private OfferRepository offerRepository;
    private ModelRepository modelRepository;
    private UserRepository userRepository;
    private final ValidationUtil validationUtil;
    private ModelMapper modelMapper;
    @Autowired
    public OfferServiceImpl(OfferRepository offerRepository, ModelRepository
modelRepository, UserRepository userRepository, ValidationUtil
validationUtil, ModelMapper modelMapper) {
        this.offerRepository = offerRepository;
        this.modelRepository = modelRepository;
        this.userRepository = userRepository;
        this.validationUtil = validationUtil;
        this.modelMapper = modelMapper;
    }
```

```
    @Override
    @Cacheable("offers")
    public List<ShowOfferVM> getAllOffers() {
        return offerRepository.findAll().stream().map(offer ->
modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }

    @Override
    @CacheEvict(cacheNames = "offers", allEntries = true)
    public void addNewOffer(OfferDto offerDto) {
        Offer offer = modelMapper.map(offerDto, Offer.class);

offer.setModel(modelRepository.findByName(offerDto.getModel()).orElseThrow())
;

offer.setSeller(userRepository.findByUsername(offerDto.getSeller()).orElseThr
ow());

        offer.setCreated(LocalDateTime.now());
        offer.setModified(LocalDateTime.now());
        this.offerRepository.saveAndFlush(offer);
    }

    @Override
    public List<ModelDto> showModel() {
        return modelRepository.findAll().stream().map(model ->
modelMapper.map(model, ModelDto.class)).collect(Collectors.toList());
    }
    @Override
    public List<UserDto> showUser() {
        return userRepository.findAll().stream().map(user ->
modelMapper.map(user, UserDto.class)).collect(Collectors.toList());
    }

    @Override
    public List<ShowOfferVM> findOffersByBrandName(String brandName) {
        return
offerRepository.findOffersByBrandName(brandName).stream().map(offer ->
modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }
    @Override
    public ShowOfferVM offerDetails(String offerID) {
        return modelMapper.map(offerRepository.findById(offerID),
ShowOfferVM.class);
    }

    @Override
    public List<ShowOfferVM> findOffersWithActiveClients() {
        return
offerRepository.findOffersWithActiveClients().stream().map(offer ->
modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }

    @Override
    public List<ShowOfferVM> findOffersBySellerUsername(String username) {
        return
offerRepository.findOffersBySellerUsername(username).stream().map(offer ->
modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }

    @Override
    public List<ShowOfferVM> findAllByOrderByPriceAsc() {
        return offerRepository.findAllByOrderByPriceAsc().stream().map(offer
```

```
-> modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }

    @Override
    public List<ShowOfferVM> findAllByOrderByPriceDesc() {
        return offerRepository.findAllByOrderByPriceDesc().stream().map(offer
-> modelMapper.map(offer, ShowOfferVM.class)).collect(Collectors.toList());
    }

    @Override
    public void deleteOfferById(String id) {
        offerRepository.deleteById(id);
    }

    @Override
    public OfferDto getOfferById(String id) {
        return modelMapper.map(offerRepository.findById(id), OfferDto.class);
    }
}


@Service
public class UserRoleServiceImpl implements UserRoleService {

    private UserRoleRepository userRoleRepository;
    private final ValidationUtil validationUtil;
    private ModelMapper modelMapper;

    @Autowired
    public UserRoleServiceImpl(UserRoleRepository userRoleRepository,
ValidationUtil validationUtil, ModelMapper modelMapper) {
        this.userRoleRepository = userRoleRepository;
        this.validationUtil = validationUtil;
        this.modelMapper = modelMapper;
    }

    @Override
    public void addUserRole(UserRoleDto userRoleDto) {
        UserRole userRole = modelMapper.map(userRoleDto, UserRole.class);
        UserRole savedUserRole = userRoleRepository.save(userRole);
        modelMapper.map(savedUserRole, UserRoleDto.class);
    }

    @Override
    public List<UserRoleDto> getAllUserRoles() {
        return userRoleRepository.findAll().stream().map(userRole ->
modelMapper.map(userRole, UserRoleDto.class)).collect(Collectors.toList());
    }

    @Override
    public ShowUserRoleVM userRoleDetails(String userRole) {
        return
modelMapper.map(userRoleRepository.findByRole(Role.valueOf(userRole)).orElse(
null), ShowUserRoleVM.class);
    }
}


@Service
@EnableCaching
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;
    private UserRoleRepository userRoleRepository;
```

```
    private final ValidationUtil validationUtil;
    private ModelMapper modelMapper;
    @Autowired
    public UserServiceImpl(UserRepository userRepository, UserRoleRepository
userRoleRepository, ValidationUtil validationUtil, ModelMapper modelMapper) {
        this.userRepository = userRepository;
        this.userRoleRepository = userRoleRepository;
        this.validationUtil = validationUtil;
        this.modelMapper = modelMapper;
    }

    @Override
    @Cacheable("users")
    public List<ShowUserVM> getAllUsers() {
        return userRepository.findAll().stream().map(user ->
modelMapper.map(user, ShowUserVM.class)).collect(Collectors.toList());
    }
    @Override
    @CacheEvict(cacheNames = "users", allEntries = true)
    public void addUser(UserDto userDto) {
            User user = modelMapper.map(userDto, User.class);

user.setRole(userRoleRepository.findByRole(userDto.getRole()).orElseThrow());
            user.setIsActive(true);
            user.setCreated(LocalDateTime.now());
            user.setModified(LocalDateTime.now());
            userRepository.saveAndFlush(user);
    }

    @Override
    public ShowUserVM userDetails(String userName) {
        return
modelMapper.map(userRepository.findByUsername(userName).orElse(null),
ShowUserVM.class);
    }

    @Override
    public List<UserRoleDto> showRole() {
        return userRoleRepository.findAll().stream().map(userRole ->
modelMapper.map(userRole, UserRoleDto.class)).collect(Collectors.toList());
    }
    @Override
    @CacheEvict(cacheNames = "users", allEntries = true)
    public void deactivateUser(String username) {
        Optional<User> user = userRepository.findByUsername(username);
        if (user.isPresent()) {
            User deactUser = user.get();
            deactUser.setIsActive(false);
            deactUser.setModified(LocalDateTime.now());
            userRepository.save(deactUser);
        }
    }
}
```

ПРИЛОЖЕНИЕ Е – VALIDATION

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
@Constraint(validatedBy = UniqueBrandNameValidator.class)
public @interface UniqueBrandName {
    String message() default "Brand already exists!";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}


public class UniqueBrandNameValidator implements
ConstraintValidator<UniqueBrandName, String> {
    private final BrandRepository brandRepository;

    public UniqueBrandNameValidator(BrandRepository brandRepository) {
        this.brandRepository = brandRepository;
    }
    @Override
    public boolean isValid(String s, ConstraintValidatorContext
constraintValidatorContext) {
        return brandRepository.findByName(s).isEmpty();
    }
}


@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
@Constraint(validatedBy = UniqueUsernameValidator.class)
public @interface UniqueUsername {
    String message() default "User already exists!";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}


public class UniqueUsernameValidator implements
ConstraintValidator<UniqueUsername, String> {
    private final UserRepository userRepository;
    public UniqueUsernameValidator(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
    @Override
    public boolean isValid(String s, ConstraintValidatorContext
constraintValidatorContext) {
        return userRepository.findByUsername(s).isEmpty();
    }
}


public interface ValidationUtil {
    <E> boolean isValid(E object);
    <E> Set<ConstraintViolation<E>> violations(E object);

}

@Component
public class ValidationUtilImpl implements ValidationUtil{
    private final Validator validator;
    @Autowired
    public ValidationUtilImpl(Validator validator) {
        this.validator = validator;
    }
    @Override
```

```
    public <E> boolean isValid(E object) {
        Set<ConstraintViolation<E>> violations =
this.validator.validate(object);
        return violations.size() == 0;
    }
    @Override
    public <E> Set<ConstraintViolation<E>> violations(E object) {
        return this.validator.validate(object);
    }
}
```

ПРИЛОЖЕНИЕ Ж – CONTROLLERS

```
@Controller
@RequestMapping("/users")
public class AuthController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    private AuthService authService;
    @Autowired
    public AuthController(AuthService authService) {
        this.authService = authService;
    }

    @ModelAttribute("userRegistrationDto")
    public UserRegistrationDto initForm() {
        return new UserRegistrationDto();
    }
    @GetMapping("/register")
    public String register() {
        return "register";
    }
    @PostMapping("/register")
    public String doRegister(@Valid UserRegistrationDto userRegistrationDto,
                             BindingResult bindingResult,
                             RedirectAttributes redirectAttributes, Principal
principal) {

        if (bindingResult.hasErrors()) {
            redirectAttributes.addFlashAttribute("userRegistrationDto",
userRegistrationDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.userRegistrationDto", bindingResult);

            return "redirect:/users/register";
        }
        this.authService.register(userRegistrationDto);

        LOG.log(Level.INFO, "Registered " + principal.getName());

        return "redirect:/users/login";
    }
    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @PostMapping("/login-error")
    public String onFailedLogin(

@ModelAttribute(UsernamePasswordAuthenticationFilter.SPRING_SECURITY_FORM_USE
```

```
RNAME_KEY) String username,
            RedirectAttributes redirectAttributes) {


redirectAttributes.addFlashAttribute(UsernamePasswordAuthenticationFilter.SPR
ING_SECURITY_FORM_USERNAME_KEY, username);
        redirectAttributes.addFlashAttribute("badCredentials", true);

        return "redirect:/users/login";
    }
    @GetMapping("/profile")
    public String profile(Principal principal, Model model) {

        LOG.log(Level.INFO, "Logged into the profile " +
principal.getName());

        String username = principal.getName();
        User user = authService.getUser(username);

        UserProfileView userProfileView = new UserProfileView(
                username,
                user.getFirstName(),
                user.getLastName(),
                user.getImageUrl()
        );

        model.addAttribute("user", userProfileView);

        return "profile";
    }
}


@Controller
@RequestMapping("/brands")
public class BrandController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @Autowired
    private BrandService brandService;
    public BrandController(BrandService brandService) {
        this.brandService = brandService;
    }
    @GetMapping("/all")
    public String showAllBrands(Principal principal, Model model){
        LOG.log(Level.INFO, "Show all brands for " + principal.getName());
        model.addAttribute("brandInfos", brandService.getAllBrands());

        return "brands-all";
    }

    @GetMapping("/add")
    public String addBrand() {
        return "brand-add";
    }

    @ModelAttribute("brandModel")
    public BrandDto initBrand() {
        return new BrandDto();
    }

    @PostMapping("/add")
    public String addBrand(@Valid BrandDto brandDto, BindingResult
bindingResult, RedirectAttributes redirectAttributes, Principal principal){
        if (bindingResult.hasErrors()) {
```

```
            redirectAttributes.addFlashAttribute("brandModel", brandDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.brandModel",
                    bindingResult);
            return "redirect:/brands/add";
        }

        brandService.addNewBrand(brandDto);

        LOG.log(Level.INFO, "Add new brand for " + principal.getName());

        return "redirect:/";
    }
}


@Controller
public class HomeController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @GetMapping("/")
    public String homePage() {
        return "index";
    }
}


@Controller
@RequestMapping("/models")
public class ModelController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @Autowired
    private ModelService modelService;
    public ModelController(ModelService modelService) {
        this.modelService = modelService;
    }

    @GetMapping("/all")
    public String showAllModels(Principal principal, Model model){
        LOG.log(Level.INFO, "Show all models for " + principal.getName());
        model.addAttribute("modelInfos", modelService.getAllModels());
        return "models-all";
    }

    @GetMapping("/add")
    public String addModel(Model model) {
        model.addAttribute("availableBrands", modelService.showBrand());
        return "model-add";
    }

    @ModelAttribute("modelModel")
    public ModelDto initModel() {
        return new ModelDto();
    }
    @PostMapping("/add")
    public String addModel(@Valid ModelDto modelDto, BindingResult
bindingResult, RedirectAttributes redirectAttributes, Principal principal){

        if (bindingResult.hasErrors()) {
            redirectAttributes.addFlashAttribute("modelModel", modelDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.modelModel",
                    bindingResult);
```

```
            return "redirect:/models/add";
        }
        modelService.addNewModel(modelDto);

        LOG.log(Level.INFO, "Add new model for " + principal.getName());

        return "redirect:/";
    }
}


@Controller
@RequestMapping("/offers")
public class OfferController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @Autowired
    private OfferService offerService;
    @Autowired
    private AuthService authService;

    public OfferController(OfferService offerService) {
        this.offerService = offerService;
    }

    @GetMapping("/all")
    public String showAllOffers(Principal principal, Model model){
        LOG.log(Level.INFO, "Show all offers for " + principal.getName());
        model.addAttribute("offerInfos", offerService.getAllOffers());
        return "offers-all";
    }

    @GetMapping("/add")
    public String addOffer(Model model) {
        model.addAttribute("availableModel", offerService.showModel());
        return "offer-add";
    }

    @ModelAttribute("offerModel")
    public OfferDto initOffer() {
        return new OfferDto();
    }
    @PostMapping("/add")
    public String addOffer(@Valid OfferDto offerDto, BindingResult
bindingResult, RedirectAttributes redirectAttributes, Principal principal){

        if (bindingResult.hasErrors()) {
            redirectAttributes.addFlashAttribute("offerModel", offerDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.offerModel",
                    bindingResult);
            return "redirect:/offers/add";
        }
        offerDto.setSeller(principal.getName());
        offerService.addNewOffer(offerDto);

        LOG.log(Level.INFO, "Add new offer for " + principal.getName());

        return "redirect:/";
    }

    @GetMapping("/active_offers")
    public String OffersWithActiveClients(Principal principal, Model model) {
        LOG.log(Level.INFO, "Show all active offers for " +
```

```
principal.getName());
        model.addAttribute("offerInfosActive",
offerService.findOffersWithActiveClients());
        return "active_clients";
    }

    @GetMapping("/my_offers")
    public String showMyOffers(Principal principal, Model model){
        LOG.log(Level.INFO, "Show my all offers for " + principal.getName());
        String username = principal.getName();
        model.addAttribute("offersUserInfos",
offerService.findOffersBySellerUsername(username));
        return "user_offers";
    }

    @DeleteMapping("/delete/{id}")
    void deleteOffer(@PathVariable String id){
        offerService.deleteOfferById(id);
    }

    @GetMapping("/sorted/asc")
    public String getSortedOffersAsc(Principal principal, Model model) {
        LOG.log(Level.INFO, "Looked through the sorted offers for " +
principal.getName());
        model.addAttribute("offerInfosActive",
offerService.findAllByOrderByPriceAsc());
        return "active_clients";
    }

    @GetMapping("/sorted/desc")
    public String getSortedOffersDesc(Principal principal, Model model) {
        LOG.log(Level.INFO, "Looked through the sorted offers for " +
principal.getName());
        model.addAttribute("offerInfosActive",
offerService.findAllByOrderByPriceDesc());
        return "active_clients";
    }

}


@Controller
@RequestMapping("/users")
public class UserController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @Autowired
    private UserService userService;

    public UserController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping("/all")
    public String showAllUser(Principal principal, Model model){
        LOG.log(Level.INFO, "Show all users for " + principal.getName());
        model.addAttribute("userInfos", userService.getAllUsers());
        return "users-all";
    }

    @GetMapping("/add")
    public String addUser(Model model) {
        model.addAttribute("availableRole", userService.showRole());
        return "user-add";
    }
```

```
    @ModelAttribute("userModel")
    public UserDto initUser() {
        return new UserDto();
    }

    @PostMapping("/add")
    public String addUser(@Valid UserDto userDto, BindingResult
bindingResult, RedirectAttributes redirectAttributes, Principal principal){

        if (bindingResult.hasErrors()) {
            redirectAttributes.addFlashAttribute("userModel", userDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.userModel",
                    bindingResult);
            return "redirect:/users/add";
        }

        userService.addUser(userDto);

        LOG.log(Level.INFO, "Add new user for " + principal.getName());

        return "redirect:/";
    }

    @GetMapping("/deactivate_user/{username}")
    public String deactivateUser(@PathVariable String username, Principal
principal) {
        LOG.log(Level.INFO, "Deactivated the user for " +
principal.getName());
        userService.deactivateUser(username);
        return "redirect:/users/all";
    }
}


@Controller
@RequestMapping("/userRoles")
public class UserRoleController {
    private static final Logger LOG = LogManager.getLogger(Controller.class);
    @Autowired
    private UserRoleService userRoleService;

    public UserRoleController(UserRoleService userRoleService) {
        this.userRoleService = userRoleService;
    }

    @GetMapping("/all")
    public String showAllUserRole(Principal principal, Model model){
        LOG.log(Level.INFO, "Show all users roles for " +
principal.getName());
        model.addAttribute("userRoleInfos",
userRoleService.getAllUserRoles());
        return "userRoles-all";
    }
    @GetMapping("/add")
    public String addUserRole() {
        return "userRole-add";
    }
    @ModelAttribute("userRoleModel")
    public UserRoleDto initUserRole() {
        return new UserRoleDto();
    }
    @PostMapping("/add")
```

```
    public String addUserRole(@Valid UserRoleDto userRoleDto, BindingResult
bindingResult, RedirectAttributes redirectAttributes, Principal principal){
        if (bindingResult.hasErrors()) {
            redirectAttributes.addFlashAttribute("userRoleModel",
userRoleDto);

redirectAttributes.addFlashAttribute("org.springframework.validation.BindingR
esult.userRoleModel",
                    bindingResult);
            return "redirect:/userRoles/add";
        }

        userRoleService.addUserRole(userRoleDto);

        LOG.log(Level.INFO, "Add new user role for" + principal.getName());

        return "redirect:/";
    }
}
```

ПРИЛОЖЕНИЕ 3 – THYMELEAF

```
<footer id="footer" class="footer mt-auto py-3 bg-light" th:fragment="footer"
xmlns:th="http://www.thymeleaf.org">
  <div class="container">
    <div class="text-muted p-2 text-center">&copy; Web-project Bukin 2023.
All rights reserved.</div>
  </div>
</footer>
```

```
<head th:fragment="head" xmlns:th="http://www.thymeleaf.org">
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Spring Web Project</title>
  <link rel="stylesheet" href="/css/bootstrap.min.css"/>
</head>
```

```
<nav class="navbar navbar-expand-lg bg-light"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
    <div class="navbar" id="navbarNavAltMarkup">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
                <a class="nav-item nav-link text-dark" href="/">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-item nav-link text-dark" href="/brands/all">All
Brands</a>
            </li>

            <li class="nav-item">

                <a class="nav-item nav-link text-dark" href="/models/all">All
Models</a>
            </li>

            <li class="nav-item" sec:authorize="hasRole('ADMIN')">

                <a class="nav-item nav-link text-dark" href="/offers/all">All
Offers</a>
```

```
            </li>

            <li class="nav-item" >

                <a class="nav-item nav-link text-dark"
href="/offers/active_offers">Active offers</a>
            </li>

            <li class="nav-item" sec:authorize="hasAnyRole('ADMIN', 'USER')">

                <a class="nav-item nav-link text-dark"
href="/offers/my_offers">My offers</a>
            </li>

            <li class="nav-item" sec:authorize="hasRole('ADMIN')">
                <a class="nav-item nav-link text-dark" href="/users/all">All
Users</a>
            </li>

<!--            <li class="nav-item" sec:authorize="hasRole('ADMIN')">-->
<!--                <a class="nav-item nav-link text-dark"
href="/userRoles/all">All UserRoles</a>-->
<!--            </li>-->

            <li class="nav-item" sec:authorize="hasRole('ADMIN')">
                <a class="nav-item nav-link text-dark" href="/users/add">Add
User</a>
            </li>

            <li class="nav-item" sec:authorize="hasAnyRole('ADMIN', 'USER')">
                <a class="nav-item nav-link text-dark" href="/offers/add">Add
Offer</a>
            </li>

            <li class="nav-item" sec:authorize="hasRole('ADMIN')">
                <a class="nav-item nav-link text-dark" href="/models/add">Add
Model</a>
            </li>

            <li class="nav-item" sec:authorize="hasRole('ADMIN')">
                <a class="nav-item nav-link text-dark" href="/brands/add">Add
Brand</a>
            </li>

<!--            <li class="nav-item" sec:authorize="hasRole('ADMIN')">-->
<!--                <a class="nav-item nav-link text-dark"
href="/userRoles/add">Add UserRole</a>-->
<!--            </li>-->

            <li class="nav-item" sec:authorize="isAuthenticated()">
                <a class="nav-item nav-link text-dark text-info"
href="/users/profile"><th:block sec:authentication="name"></th:block></a>
            </li>

            <li class="nav-item" sec:authorize="isAuthenticated()">
                <form
                        th:method="post"
                        th:action="@{/users/logout}">
                    <input class="btn btn-link nav-link" type="submit"
value="Logout">
                </form>
            </li>

            <li class="nav-item" sec:authorize="!isAuthenticated()">
```

```
                <a class="nav-item nav-link text-dark"
th:href="@{/users/register}">Register</a>
            </li>

            <li class="nav-item" sec:authorize="!isAuthenticated()">
                <a class="nav-item nav-link text-dark"
th:href="@{/users/login}">Login</a>
            </li>

        </ul>
    </div>
</nav>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
  <div class="container">
    <div class="row">
      <div class="col-md-2"></div>
      <div class="jumbotron text-center rounded col-md-8 align-self-center
pt-5">
        <h2 class="text-center text-dark mt-5">Активные предложения</h2>
        <div class="mb-3">
          <a th:href="@{/offers/sorted/asc}" class="btn btn-primary btn-sm
mx-1">Сортировать по возрастанию цены</a>
          <a th:href="@{/offers/sorted/desc}" class="btn btn-primary btn-sm
mx-1">Сортировать по убыванию цены</a>
        </div>
        <div th:each="o : ${offerInfosActive}" class="d-flex flex-row bg-text
mb-3" style="border: 1px solid #383838; text-align: left;">
          <img th:src="${o.image_url}" alt="Изображение" class="img-fluid mx-
2" style="max-width: 300px; max-height: 200px;"/>
          <div class="d-flex flex-column mx-2">
            <h4 th:text="'Описание: ' + ${o.description}"></h4>
            <h4 th:text="'Двигатель: ' + ${o.engine}"></h4>
            <h4 th:text="'Пробег: ' + ${o.mileage}"></h4>
            <h4 th:text="'Стоимость: ' + ${o.price}"></h4>
            <h4 th:text="'Трансмиссия: ' + ${o.transmission}"></h4>
            <h4 th:text="'Год: ' + ${o.years}"></h4>
            <h4 th:text="'Модель: ' + ${o.model.name}"></h4>
            <h4 th:text="'Продавец: ' + ${o.seller.username}"></h4>
          </div>
        </div>
      </div>
    </div>
  </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">

<head th:replace="fragments/head"/>

<body class="d-flex flex-column h-100">

<div th:replace="fragments/navbar">Navbar</div>
```

```
<div class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>

            <div class="jumbotron text-center rounded col-md-12 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Добавить бренд</h1>
                </div>

                <form th:action="@{/brands/add}"
                      th:method="post"
                      th:object="${brandModel}"
                      class="m-md-auto w-50 pb-3">

                    <div class="mb-3">
                        <label class="form-label" for="name">Имя
бренда</label>
                        <input th:field="*{name}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
                               id="name" aria-describedby="name"
name="name"/>
                        <small th:if="${#fields.hasErrors('name')}"
th:errors="*{name}" class="text-danger">Error
                            message</small>
                    </div>

                    <button type="submit" class="btn btn-secondary">Добавить
бренд</button>
                </form>
            </div>
        </div>
    </div>
</div>

<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">Список брендов</h2>
                <div class='row mb-4 d-flex justify-content-around'>
                    <div th:each="b : ${brandInfos}" class="col-md-4 d-flex
flex-column bg-text mb-3 rounded" style="border: 1px solid #383838; text-
align: center;">
                        <h4 th:text="'Имя бренда: ' + *{b.name}"></h4>
                    </div>
                </div>
            </div>
        </div>
```

```
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Добро пожаловать на мой учебный web проект на
spring!</h1>
                </div>
                <div><span th:text="'Время последнего обновления
базы:'"></span> 
                    <span th:text="${#dates.createNow()}"></span>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Login</h1>
                </div>
                <form th:action="@{/users/login}"
                      th:method="post"
                      class="m-md-auto w-50 pb-3">
                    <div class="mb-3">
                        <label class="form-label" for="username">Имя
пользователя</label>
                        <input id="username"
                               name="username"
                               th:value="${username}"
                               type="text" class="form-control text-
center"aria-describedby="userNameHelp"/>
                    </div>

                    <div class="mb-3">
                        <label class="form-label"
```

```
for="password">Пароль</label>
                        <input id="password"
                               name="password"
                               th:value="${password}"
                               type="password" class="form-control text-
center" aria-describedby="passwordHelp"/>
                    </div>

                    <p class="text-danger" th:if="${badCredentials}">
                        Invalid username or password.
                    </p>

                    <button type="submit" class="btn btn-
secondary">Login</button>
                </form>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">

<head th:replace="fragments/head"/>

<body class="d-flex flex-column h-100">

<div th:replace="fragments/navbar">Navbar</div>

<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Add Model</h1>
                </div>
                <form th:action="@{/models/add}"
                      th:method="post"
                      th:object="${modelModel}"
                      class="m-md-auto w-50 pb-3">

                    <div class="mb-3">
                        <label class="form-label" for="name">Название</label>
                        <input th:field="*{name}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
id="name" aria-describedby="nameHelp"
                               name="name"/>
                        <small th:if="${#fields.hasErrors('name')}"
th:errors="*{name}" class="text-danger">Ошибка</small>
                    </div>

                    <fieldset class="mb-3">
                        <legend>Выберите категорию модели</legend>
                        <!-- Переключатели для выбора категории модели. -->
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{category}"
```

```
                                        th:value="CAR"
                                        th:errorclass="is-invalid"
                                        type="radio" class="form-check-input"
name="category"/>
                                    <span>Автомобиль</span>
                                </label>
                            </div>
                            <div class="form-check">
                                <label class="form-check-label">
                                    <input th:field="*{category}"
                                        th:value="BUSS"
                                        th:errorclass="is-invalid"
                                        type="radio" class="form-check-input"
name="category"/>
                                    <span>Автобус</span>
                                </label>
                            </div>
                            <div class="form-check">
                                <label class="form-check-label">
                                    <input th:field="*{category}"
                                        th:value="TRUCK"
                                        th:errorclass="is-invalid"
                                        type="radio" class="form-check-input"
name="category"/>
                                    <span>Грузовик</span>
                                </label>
                            </div>
                            <div class="form-check">
                                <label class="form-check-label">
                                    <input th:field="*{category}"
                                        th:value="MOTORCYCLE"
                                        th:errorclass="is-invalid"
                                        type="radio" class="form-check-input"
name="category"/>
                                    <span>Мотоцикл</span>
                                </label>
                            </div>
                            <small th:if="${#fields.hasErrors('category')}"
th:errors="*{category}"
                                   class="text-danger">Сообщение об
ошибке</small>
                        </fieldset>

                        <div class="mb-3">
                            <label class="form-label" for="imageUrl">URL
изображения</label>
                            <input th:field="*{imageUrl}"
                                   th:errorclass="is-invalid"
                                   type="text" class="form-control text-center"
                                   id="imageUrl" aria-describedby="imageUrl"
name="imageUrl"/>
                            <small th:if="${#fields.hasErrors('imageUrl')}"
th:errors="*{imageUrl}"
                                   class="text-danger">Ошибка</small>
                        </div>

                        <div class="mb-3">
                            <label class="form-label" for="startYear">Год
начала</label>
                            <input th:field="*{startYear}"
                                   th:errorclass="is-invalid"
                                   type="number" class="form-control text-center"
                                   id="startYear" aria-describedby="startYear"
name="startYear"/>
```

```
                        <small th:if="${#fields.hasErrors('startYear')}"
th:errors="*{startYear}"
                               class="text-danger">Ошибка</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label" for="endYear">Год
окончания</label>
                        <input th:field="*{endYear}"
                               th:errorclass="is-invalid"
                               type="number" class="form-control text-center"
                               id="endYear" aria-describedby="endYear"
name="endYear"/>
                        <small th:if="${#fields.hasErrors('endYear')}"
th:errors="*{endYear}"
                               class="text-danger">Ошибка</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label" for="brand"><h5>All
available brands</h5></label>
                        <select multiple="false" class="form-control text-
center"
                                id="brand" name="name" th:field="*{brand}">
                            <option th:each="m : ${availableBrands}"
                                    th:value="${m.name}"
                                    th:text="${m.name}">Brand name
                            </option>
                        </select>
                        <small th:if="${#fields.hasErrors('brand')}"
th:errors="*{brand}"
                               class="text-danger">Error
                            message</small>
                    </div>

                    <button type="submit" class="btn btn-secondary">Добавить
модель</button>
                </form>
            </div>
        </div>
    </div>
</main>

<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">Список моделей</h2>
                <div class='row mb-4 d-flex justify-content-around'>
                    <div th:each="m : ${modelInfos}" class="col-md-4 d-flex
flex-column bg-text mb-3 rounded" style="border: 1px solid #383838; text-
align: center;">
```

```
                        <img th:src="${m.imageUrl}" alt="Image" class="img-
fluid mx-auto" style="max-width: 250px; max-height: 250px;"/>
                        <h4 th:text="'Название: ' + *{m.name}"></h4>
                        <h4 th:text="'Категория: ' + *{m.category}"></h4>
                        <h4 th:text="'Год начала выпуска: ' +
*{m.startYear}"></h4>
                        <h4 th:text="'Год окончания выпуска: ' +
*{m.endYear}"></h4>
                        <h4 th:text="'Бренд: ' + *{m.brand.name}"></h4>

                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Add Offer</h1>
                </div>
                <form th:action="@{/offers/add}"
                      th:method="post"
                      th:object="${offerModel}"
                      class="m-md-auto w-50 pb-3">

                    <div class="mb-3">
                        <label class="form-label"
for="description">Описание</label>
                        <input th:field="*{description}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
id="description"
                               name="description"/>
                        <small th:if="${#fields.hasErrors('description')}"
th:errors="*{description}"
                               class="text-danger">Ошибка</small>
                    </div>

                    <fieldset class="mb-3">
                        <legend>Выберите тип двигателя</legend>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{engine}"
                                       th:value="GASOLINE"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="engine"/>
                                <span>Бензин</span>
                            </label>
```

```
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{engine}"
                                       th:value="DIESEL"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="engine"/>
                                <span>Дизель</span>
                            </label>
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{engine}"
                                       th:value="ELECTRIC"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="engine"/>
                                <span>Электричество</span>
                            </label>
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{engine}"
                                       th:value="HYBRID"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="engine"/>
                                <span>Гибрид</span>
                            </label>
                        </div>
                        <small th:if="${#fields.hasErrors('engine')}"
th:errors="*{engine}"
                               class="text-danger">Сообщение об
ошибке</small>
                    </fieldset>

                    <div class="mb-3">
                        <label class="form-label" for="imageUrl">URL
изображения</label>
                        <input th:field="*{image_url}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
                               id="imageUrl" name="imageUrl"/>
                        <small th:if="${#fields.hasErrors('image_url')}"
th:errors="*{image_url}"
                               class="text-danger">Ошибка</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label"
for="mileage">Пробег</label>
                        <input th:field="*{mileage}"
                               th:errorclass="is-invalid"
                               type="number" class="form-control text-center"
                               id="mileage" name="mileage"/>
                        <small th:if="${#fields.hasErrors('mileage')}"
th:errors="*{mileage}"
                               class="text-danger">Ошибка</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label" for="price">Цена</label>
                        <input th:field="*{price}"
```

```
                                th:errorclass="is-invalid"
                                type="number" class="form-control text-center"
                                id="price" name="price"/>
                        <small th:if="${#fields.hasErrors('price')}"
th:errors="*{price}"
                                class="text-danger">Ошибка</small>
                    </div>

                    <fieldset class="mb-3">
                        <legend>Выберите тип трансмиссии</legend>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{transmission}"
                                       th:value="MANUAL"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="transmission"/>
                                <span>Механическая</span>
                            </label>
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{transmission}"
                                       th:value="AUTOMATIC"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="transmission"/>
                                <span>Автоматическая</span>
                            </label>
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{transmission}"
                                       th:value="ROBOT"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="transmission"/>
                                <span>Роботизированная</span>
                            </label>
                        </div>
                        <div class="form-check">
                            <label class="form-check-label">
                                <input th:field="*{transmission}"
                                       th:value="VARIATOR"
                                       th:errorclass="is-invalid"
                                       type="radio" class="form-check-input"
name="transmission"/>
                                <span>Вариатор</span>
                            </label>
                        </div>
                        <small th:if="${#fields.hasErrors('transmission')}"
th:errors="*{transmission}"
                               class="text-danger">Сообщение об
ошибке</small>
                    </fieldset>

                    <div class="mb-3">
                        <label class="form-label" for="years">Годы
выпуска</label>
                        <input th:field="*{years}"
                               th:errorclass="is-invalid"
                               type="number" class="form-control text-center"
                               id="years" name="years"/>
                        <small th:if="${#fields.hasErrors('years')}"
```

```
th:errors="*{years}"
                                    class="text-danger">Ошибка</small>
                        </div>

                        <div class="mb-3">
                            <label class="form-label" for="model">Название
модели</label>
                            <select class="form-control text-center"
                                    id="model" name="model" th:field="*{model}">
                                <option th:each="model : ${availableModel}"
                                        th:value="${model.name}"
                                        th:text="${model.name}">Model name
                                </option>
                            </select>
                            <small th:if="${#fields.hasErrors('model')}"
th:errors="*{model}" class="text-danger">Ошибка</small>
                        </div>

                        <button type="submit" class="btn btn-secondary">Добавить
предложение</button>
                    </form>
                </div>
            </div>
        </div>
    </main>
    <footer th:replace="fragments/footer"/>
    </body>
    </html>


    <!DOCTYPE html>
    <html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments/head"/>
    <body class="d-flex flex-column h-100">
    <div th:replace="fragments/navbar">Navbar</div>
    <main class="flex-shrink-0">
        <div class="container">
            <div class="row">
                <div class="col-md-2"></div>
                <main class="jumbotron text-center rounded col-md-12 align-self-
center pt-5">
                    <div class="justify-content-center">
                        <h1>Offer Details</h1>
                    </div>
                    <div class="row bg-light rounded">
                        <div class="col-md-6">
                            <h2>Offer</h2>
                            <h4 th:text="'Модель: ' + ${offerDetail.model}"></h4>
                            <h4 th:text="'Клиент: ' +
${offerDetail.seller}"></h4>
                            <h4 th:text="'Год: ' + ${offerDetail.years}"></h4>
                            <h4 th:text="'Пробег: ' +
${offerDetail.mileage}"></h4>
                            <h4 th:text="'Описание: ' +
${#strings.substring(offerDetail.description,0,5)}"></h4>
                            <h4 th:text="'Цена: ' + ${offerDetail.price}"></h4>
                            <h4 th:text="'Тип трансмиссии: ' +
${offerDetail.transmission}"></h4>
                            <h4 th:text="'Тип двигателя: ' +
${offerDetail.engine}"></h4>

                        </div>
                    </div>
                </main>
```

```
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">Список всех
предложений</h2>
                <div th:each="o : ${offerInfos}" class="d-flex flex-column
bg-text mb-3" style="border: 1px solid #383838; text-align: center;">
                    <img th:src="${o.image_url}" alt="Image" class="img-fluid
mx-auto" style="max-width: 300px; max-height: 200px;"/>
                    <h4 th:text="'Описание: ' + ${o.description}"></h4>
                    <h4 th:text="'Двигатель: ' + ${o.engine}"></h4>
                    <h4 th:text="'Пробег: ' + ${o.mileage}"></h4>
                    <h4 th:text="'Стоимость: ' + ${o.price}"></h4>
                    <h4 th:text="'Трансмиссия: ' + ${o.transmission}"></h4>
                    <h4 th:text="'Год: ' + ${o.years}"></h4>
                    <h4 th:text="'Модель: ' + ${o.model.name}"></h4>
                    <h4 th:text="'Продавец: ' + ${o.seller.username}"></h4>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.w3.org/1999/xhtml">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
  <div class="container">
    <div class="row">
      <div class="col-md-2"></div>
      <main class="jumbotron text-center rounded col-md-12 align-self-center
pt-5">
        <div class="justify-content-center">
          <h1>Profile</h1>
        </div>
        <div class="row bg-light rounded">
          <div class="col-md-6" th:object="${user}">
            <h2>User</h2>
            <img th:src="${imageUrl}" alt="Product Image" class="img-fluid
mx-auto" style="max-width: 250px; max-height: 250px;"/>
            <h4 th:text="'Имя пользователя: ' + *{username}"></h4>
```

```
            <h4 th:text="'Имя: ' + *{firstName}"></h4>
            <h4 th:text="'Фамилия: ' + *{lastName}"></h4>
<!--            <h4 th:text="'Изображение: ' + *{imageUrl}"></h4>-->
            <b>User Roles: </b>
            <div sec:authentication="principal.authorities"></div>
          </div>
<!--          <div class="col-md-6"><img src="/pic/employee.png"
height="350px"></div>-->
        </div>
      </main>
    </div>
  </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
  <div class="container">
    <div class="row">
      <div class="col-md-2"></div>
      <div class="jumbotron text-center rounded col-md-8 align-self-center
pt-5">
        <div class="justify-content-center">
          <h1>Register</h1>
        </div>
        <form th:action="@{/users/register}"
              th:method="post"
              th:object="${userRegistrationDto}"
              class="m-md-auto w-50 pb-3">
          <div class="mb-3">
            <label class="form-label" for="username">Имя пользователя</label>
            <input th:field="*{username}"
                   th:errorclass="is-invalid"
                   type="text" class="form-control text-center" id="username"
aria-describedby="usernameHelp"
                   name="username"/>
            <small th:if="${#fields.hasErrors('username')}"
th:errors="*{username}" class="text-danger">Error
              message</small>
          </div>

          <div class="mb-3">
            <label class="form-label" for="firstName">Имя</label>
            <input th:field="*{firstName}"
                   th:errorclass="is-invalid"
                   type="text" class="form-control text-center"
id="firstName" aria-describedby="firstName"
                   name="firstName"/>
            <small th:if="${#fields.hasErrors('firstName')}"
th:errors="*{firstName}" class="text-danger">Error
              message</small>
          </div>

          <div class="mb-3">
            <label class="form-label" for="lastName">Фамилия</label>
            <input th:field="*{lastName}"
                   th:errorclass="is-invalid"
```

```
                    type="lastName" class="form-control text-center"
                    id="lastName" aria-describedby="lastName"
name="lastName"/>
            <small th:if="${#fields.hasErrors('lastName')}"
th:errors="*{lastName}"
                    class="text-danger">Error
                message</small>
          </div>
          <div class="mb-3">
            <label class="form-label" for="password">Пароль</label>
            <input th:field="*{password}"
                    required minlength="5" maxlength="20"
                    th:errorclass="is-invalid"
                    type="password" min="0" class="form-control text-center"
                    id="password" aria-describedby="password"
name="password"/>
            <small th:if="${#fields.hasErrors('password')}"
th:errors="*{password}"
                    class="text-danger">Error
                message</small>
          </div>

          <div class="mb-3">
            <label class="form-label" for="confirmPassword">Продублируйте
пароль</label>
            <input th:field="*{confirmPassword}"
                    required minlength="5" maxlength="20"
                    th:errorclass="is-invalid"
                    type="password" min="0" class="form-control text-center"
                    id="confirmPassword" aria-describedby="confirmPassword"
name="confirmPassword"/>
            <small th:if="${#fields.hasErrors('confirmPassword')}"
th:errors="*{confirmPassword}"
                    class="text-danger">Error
                message</small>
          </div>

          <button type="submit" class="btn btn-secondary">Register</button>
        </form>
      </div>
    </div>
  </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<div class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-12 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Add User</h1>
                </div>

                <form th:action="@{/users/add}"
```

```
                        th:method="post"
                        th:object="${userModel}"
                        class="m-md-auto w-50 pb-3">

                    <div class="mb-3">
                        <label class="form-label" for="username">Имя
пользователя</label>
                        <input th:field="*{username}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
                               id="userName" aria-describedby="userName"
name="userName"/>
                        <small th:if="${#fields.hasErrors('username')}"
th:errors="*{username}" class="text-danger">Error
                            message</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label"
for="password">Пароль</label>
                        <input th:field="*{password}"
                               th:errorclass="is-invalid"
                               type="password" class="form-control text-
center"
                               id="password" aria-describedby="password"
name="password"/>
                        <small th:if="${#fields.hasErrors('password')}"
th:errors="*{password}" class="text-danger">Error
                            message</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label" for="firstName">Имя</label>
                        <input th:field="*{firstName}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
                               id="firstName" aria-describedby="firstName"
name="firstName"/>
                        <small th:if="${#fields.hasErrors('firstName')}"
th:errors="*{firstName}" class="text-danger">Error
                            message</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label"
for="lastName">Фамилия</label>
                        <input th:field="*{lastName}"
                               th:errorclass="is-invalid"
                               type="text" class="form-control text-center"
                               id="lastName" aria-describedby="lastName"
name="lastName"/>
                        <small th:if="${#fields.hasErrors('lastName')}"
th:errors="*{lastName}" class="text-danger">Error
                            message</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label"
for="imageUrl">Изображение</label>
                        <input th:field="*{imageUrl}"
                               th:errorclass="is-invalid" type="text"
class="form-control text-center"
                               id="imageUrl" aria-describedby="imageUrl"
name="imageUrl"/>
```

```
                        <small th:if="${#fields.hasErrors('imageUrl')}"
th:errors="*{imageUrl}" class="text-danger">Error
                            message</small>
                    </div>

                    <div class="mb-3">
                        <label class="form-label" for="role"><h5>Все
доступные роли</h5></label>
                        <select class="form-control text-center" id="role"
name="role" th:field="*{role}">
                            <option th:each="r : ${availableRole}"
                                    th:value="${r.role}"
                                    th:text="${r.role}">Role name
                            </option>
                        </select>
                        <small th:if="${#fields.hasErrors('role')}"
th:errors="*{role}" class="text-danger">Error message</small>
                    </div>

                    <button type="submit" class="btn btn-secondary">Add
User</button>
                </form>
            </div>
        </div>
    </div>
</div>
</body>
<footer th:replace="fragments/footer"/>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">Мои предложения</h2>
                <div th:each="o : ${offersUserInfos}" class="d-flex flex-row
bg-text mb-3" style="border: 1px solid #383838; text-align: left;">
                    <img th:src="${o.image_url}" alt="Image" class="img-fluid
mx-2" style="max-width: 300px; max-height: 200px;"/>
                    <div class="d-flex flex-column mx-2">
                        <h4 th:text="'Описание: ' + ${o.description}"></h4>
                        <h4 th:text="'Двигатель: ' + ${o.engine}"></h4>
                        <h4 th:text="'Пробег: ' + ${o.mileage}"></h4>
                        <h4 th:text="'Стоимость: ' + ${o.price}"></h4>
                        <h4 th:text="'Трансмиссия: ' +
${o.transmission}"></h4>
                        <h4 th:text="'Год: ' + ${o.years}"></h4>
                        <h4 th:text="'Модель: ' + ${o.model.name}"></h4>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
```

```
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">

<head th:replace="fragments/head"/>

<body class="d-flex flex-column h-100">

<div th:replace="fragments/navbar">Navbar</div>

<div class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>

            <div class="jumbotron text-center rounded col-md-12 align-self-
center pt-5">
                <div class="justify-content-center">
                    <h1>Add User Role</h1>
                </div>

                <form th:action="@{/userRoles/add}"
                      th:method="post"
                      th:object="${userRoleModel}"
                      class="m-md-auto w-50 pb-3">

                    <div class="mb-3">
                        <label class="form-label" for="role">Роль
юзера</label>
                        <select th:field="*{role}"
                                th:errorclass="is-invalid"
                                class="form-control text-center"
                                id="role" name="role">
                            <option value="USER">User</option>
                            <option value="CLIENT">Seller</option>
                            <option value="ADMIN">Admin</option>
                        </select>
                        <small th:if="${#fields.hasErrors('role')}"
th:errors="*{role}" class="text-danger">Error
                            message</small>
                    </div>

                    <button type="submit" class="btn btn-secondary">Add User
Role</button>
                </form>
            </div>
        </div>
    </div>
</div>

<footer th:replace="fragments/footer"/>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
```

```
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">All Roles</h2>
                <div class='row mb-4 d-flex justify-content-around'>
                    <div th:each="u : ${userRoleInfos}" class="col-md-4 d-
flex flex-column bg-text mb-3 rounded">
                        <h4 th:text="'Role Name: ' + *{u.role}"></h4>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>


<!DOCTYPE html>
<html lang="en" class="h-100" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/head"/>
<body class="d-flex flex-column h-100">
<div th:replace="fragments/navbar">Navbar</div>
<main class="flex-shrink-0">
    <div class="container">
        <div class="row">
            <div class="col-md-2"></div>
            <div class="jumbotron text-center rounded col-md-8 align-self-
center pt-5">
                <h2 class="text-center text-dark mt-5">Список всех
пользователей</h2>
                <div class='row mb-4 d-flex justify-content-around'>
                    <div th:each="u : ${userInfos}" class="col-md-4 d-flex
flex-column bg-text mb-3 rounded" style="border: 1px solid #383838; text-
align: center;">
                        <img th:src="${u.imageUrl}" alt="Product Image"
class="img-fluid mx-auto" style="max-width: 250px; max-height: 250px;"/>
                        <h4 th:text="'Имя пользователя: ' +
*{u.username}"></h4>
                        <h4 th:text="'Полное имя: ' + *{u.firstName} + ' ' +
*{u.lastName}"></h4>
                        <h4 th:text="'Статус: ' + *{u.isActive}"></h4>
                        <h4 th:text="'Роль: ' + *{u.role}"></h4>

                        <div class="align-bottom">
                            <a class="btn btn-danger align-bottom"
th:href="@{/users/deactivate_user/{username}(username=${u.username})}">
                                Деактивировать
                            </a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</main>
<footer th:replace="fragments/footer"/>
</body>
</html>
```

ПРИЛОЖЕНИЕ И – CONFIGS

```
@Configuration
public class ApplicationBeanConfiguration {
    @Bean
    public ModelMapper modelMapper() {
        ModelMapper modelMapper = new ModelMapper();

        modelMapper.getConfiguration()
                .setFieldMatchingEnabled(true)

.setFieldAccessLevel(org.modelmapper.config.Configuration.AccessLevel.PRIVATE
);
        return modelMapper;
    }
}


@Configuration
public class AppSecurityConfiguration {
    private UserRepository userRepository;
    public AppSecurityConfiguration(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http,
SecurityContextRepository securityContextRepository) throws Exception {
        http
                .authorizeHttpRequests(
                        authorizeHttpRequests ->
                                authorizeHttpRequests.

requestMatchers(PathRequest.toStaticResources().atCommonLocations())
                                        .permitAll().
                                        requestMatchers("/", "/users/login",
"/users/register", "/users/login-error",
                                                        "offers/active_offers",
"/brands/all", "/models/all", "/sorted/desc", "/sorted/asc")
                                        .permitAll().
                                        requestMatchers("/users/profile",
"/offers/add", "/offers/my_offers").authenticated().
                                        requestMatchers("/users/add",
"/models/add", "/brands/add","/deactivate_user/").hasRole(Role.ADMIN.name()).
                                        requestMatchers("/offers/all",
"/users/all").hasRole(Role.ADMIN.name()).
                                        anyRequest().authenticated()
                )
                .formLogin(
                        (formLogin) ->
                                formLogin.
                                        loginPage("/users/login").

usernameParameter(UsernamePasswordAuthenticationFilter.SPRING_SECURITY_FORM_U
SERNAME_KEY).

passwordParameter(UsernamePasswordAuthenticationFilter.SPRING_SECURITY_FORM_P
ASSWORD_KEY).
                                        defaultSuccessUrl("/").
                                        failureForwardUrl("/users/login-
error")
                )
                .logout((logout) ->
                        logout.logoutUrl("/users/logout").
                                logoutSuccessUrl("/").
```

```
                                    invalidateHttpSession(true)
                ).securityContext(
                        securityContext -> securityContext.

securityContextRepository(securityContextRepository)
                );

        return http.build();
    }
    @Bean
    public SecurityContextRepository securityContextRepository() {
        return new DelegatingSecurityContextRepository(
                new RequestAttributeSecurityContextRepository(),
                new HttpSessionSecurityContextRepository()
        );
    }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
    @Bean
    public UserDetailsService userDetailsService() {
        return new AppUserDetailsService(userRepository);
    }
}

@Configuration
public class InterceptorConfigurator implements WebMvcConfigurer {
    @Override
    public void addInterceptors(InterceptorRegistry registry) {

    }
}


@Configuration
public class RedisConfig {
    @Value("${redis.host}")
    private String redisHost;
    @Value("${redis.port}")
    private int redisPort;
    @Bean
    public LettuceConnectionFactory redisConnectionFactory() {
        RedisStandaloneConfiguration configuration = new
RedisStandaloneConfiguration(redisHost, redisPort);

        return new LettuceConnectionFactory(configuration);
    }
    @Bean
    public RedisCacheManager cacheManager() {
        RedisCacheConfiguration cacheConfig =
myDefaultCacheConfig(Duration.ofMinutes(10)).disableCachingNullValues();

        return RedisCacheManager.builder(redisConnectionFactory())
                .cacheDefaults(cacheConfig)
                .withCacheConfiguration("brands",
myDefaultCacheConfig(Duration.ofMinutes(10)))
                .withCacheConfiguration("models",
myDefaultCacheConfig(Duration.ofMinutes(10)))
                .withCacheConfiguration("offers",
myDefaultCacheConfig(Duration.ofMinutes(10)))
                .withCacheConfiguration("users",
myDefaultCacheConfig(Duration.ofMinutes(10)))
                .build();
```

```
    }
    private RedisCacheConfiguration myDefaultCacheConfig(Duration duration) {
        return RedisCacheConfiguration
                .defaultCacheConfig()
                .entryTtl(duration)
                .serializeValuesWith(SerializationPair.fromSerializer(new
GenericJackson2JsonRedisSerializer()));
    }
}
```

```
#Data Source Properties
spring.datasource.driverClassName = org.postgresql.Driver
spring.datasource.url = jdbc:postgresql://localhost:5432/autocenter
spring.datasource.username = postgres
spring.datasource.password = rutppoptop

#JPA Properties
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql = FALSE
spring.jpa.properties.hibernate.show_sql = FALSE
spring.jpa.hibernate.ddl-auto = update

#spring.jpa.open-in-view=false

app.default.password: topsecret

#Disable the default loggers
logging.level.org = WARN
logging.level.blog = WARN

#Show SQL executed with parameter bindings
#logging.level.org.hibernate.SQL = DEBUG
#logging.level.org.hibernate.type.descriptor = TRACE

server.port= 8100

redis.host=localhost
redis.port=6379

logging.file.path=E:\\logs\\
logging.level.root=warn
logging.level.org.springframework=INFO
```