BMU
BAKI MÜHƏNDİSLİK UNİVERSİTETİ

Pet Caring Application

Teachers name: Yusif Yusifov
Student’s name: Sama Baliyeva
Student id: 220106036
Specialization: Information technologies
Subject : OOP2
Course: 3

Introduction to the PetPal App

Purpose: PetPal is designed to make pet ownership easier and more enjoyable by providing a comprehensive platform for managing pets. The app helps users register pets, track their health, and securely store important information while fostering a sense of community among pet owners.

Key Features:

- **User Management:** Secure user registration, login, and role-based access (Admin, SuperAdmin, and Regular Users).
- **Pet Management:** Intuitive tools to create, update, and delete pet profiles, including breed, age, and health details.

Authentication and Security

User Authentication:

- Implements secure login using JWT (JSON Web Tokens) for session management.
- Role-based access ensures that users only interact with features appropriate to their roles.

Role-Based Access Control:

- SuperAdmin Role: SuperAdmins have the highest level of access. They can create new Admins and manage their privileges. They can also view all users and their associated pets but cannot modify pet details unless they are the owner.
- Admin Role: Admins can create new Admins, view all users, and fetch user details by their IDs. They can see pets associated with specific users but cannot modify pet-related actions or details. Admins and SuperAdmins cannot access detailed actions performed on pets; only the respective pet owners have this capability.
- Regular Users: Users can view, create, update, and manage their own pet profiles. They have exclusive control over actions and data related to their pets.

How JWT Enforces Role-Based Control:

- Token Generation: Upon login, JWT is generated with embedded claims that include user role (e.g., SuperAdmin, Admin, Regular User).
- Token Validation: Every request is validated against the embedded role in the JWT. Middleware or security filters check the role claims to ensure the user has the necessary permissions.
- Scoped Access: APIs are protected by role-specific endpoints. For example, routes fetching all users or pets are accessible only to Admins and SuperAdmins, while routes for pet updates are restricted to the pet owner.
- Granular Permissions: The application uses a combination of user ID and role claims to ensure that pet-related actions are restricted to the owning user. Even if an Admin fetches a user's pet details, they cannot perform actions on behalf of the owner.

Password Encryption:

- Utilizes bcrypt for secure password hashing, safeguarding user credentials.

Security Best Practices:

- Data validation and sanitization to prevent injection attacks.
- HTTPS enforced for secure data transmission.

Pet Management

Pet Entity:

The system allows users to create and manage pet profiles, which include:

- Name, breed, and age.
- A description of the pet’s characteristics or special needs.

Permissions:

- Only pet owners can perform actions like updating or deleting their pet’s details.
- Admins and SuperAdmins can view pet profiles but cannot make changes.

Database Management

Backend Architecture:

- Spring Boot: Provides the backbone of the application with RESTful APIs.
- Hibernate ORM: Maps Java entities to PostgreSQL tables for seamless database operations.
- PostgreSQL: Efficiently handles user and pet-related data, including medical records and images.

Entity Relationships:

- Users have a one-to-many relationship with pets.

Performance Optimization:

- Indexed queries for fast data retrieval.
- Lazy loading for non-critical data.

No SQL Commands:

- All database interactions are handled through Hibernate, avoiding the need for raw SQL commands

Technical Stack

Backend:

- Spring Boot: Business logic and API development.
- Hibernate ORM: Simplifies database interaction.
- PostgreSQL: Scalable and reliable data storage.

Frontend:

- Kotlin: Ensures a smooth and responsive mobile experience.
- Swagger: Provides detailed API documentation for developers.

Security:

- JWT for authentication and role-based access control.
- Secure password hashing and validation mechanisms.

Challenges & Solutions

Challenge 1: Secure Role-Based Access Control

Solution: Designed a granular role hierarchy, enabling flexible access levels and improved security.

Challenge 2: Proper Error Handling

Solution: Introduced custom exceptions and a centralized exception handler to ensure clear and consistent API error responses, improving debugging and user experience.

Future Enhancements

Social Features:

- Introduce a community section for pet owners to share tips, advice, and milestones.

Gamification:

- Reward users with badges for milestones like regular check-ups or completing health records.

Cross-Platform Compatibility:

- Develop a web-based version to complement the mobile app.

API Expansion:

- Open APIs for integration with third-party pet health devices or services.

Thank you!