Architecture Backend Java - UniversPiscine

Structure du Projet

```
src/main/java/com/universpiscine/
├── UniversPiscineApplication.java          # Point d'entrée Spring Boot
│
├── config/                                  # Configuration
│   ├── SecurityConfig.java                 # Configuration Spring Security + JWT
│   ├── CorsConfig.java                     # Configuration CORS pour React
Native
│   ├── FirebaseConfig.java                 # Configuration Firebase (optionnel)
│   └── WebSocketConfig.java                # Configuration WebSocket pour le
chat
│
├── model/                                   # Entités JPA
│   ├── User.java                           # Utilisateur (Client/Admin/Employee)
│   ├── Pool.java                           # Piscine
│   ├── Task.java                           # Tâche de maintenance
│   ├── Employee.java                       # Employé/Technicien
│   ├── Client.java                         # Client
│   ├── Reservation.java                    # Réservation
│   ├── Message.java                        # Message (chat)
│   ├── Conversation.java                   # Conversation (chat)
│   ├── Notification.java                   # Notification
│   ├── Review.java                         # Avis client
│   └── enums/                              # Énumérations
│       ├── UserRole.java                   # ADMIN, CLIENT, EMPLOYEE
│       ├── PoolStatus.java                 # ACTIVE, MAINTENANCE, INACTIVE
│       ├── PoolType.java                   # PUBLIC, PRIVEE, CLUB
│       ├── TaskStatus.java                 # A_FAIRE, EN_COURS, TERMINE
│       ├── ReservationStatus.java          # EN_ATTENTE, CONFIRME, TERMINE,
ANNULE
│       ├── EmployeeStatus.java             # DISPONIBLE, OCCUPE, ABSENT
│       └── NotificationType.java           # APPOINTMENT, MESSAGE, ALERT, POOL,
SUCCESS, SYSTEM
│
├── dto/                                     # Data Transfer Objects
│   ├── request/                            # DTOs de requête
│   │   ├── LoginRequest.java
│   │   ├── RegisterRequest.java
│   │   ├── PoolRequest.java
│   │   ├── TaskRequest.java
│   │   ├── ReservationRequest.java
│   │   ├── MessageRequest.java
│   │   ├── ClientRequest.java
│   │   ├── EmployeeRequest.java
│   │   └── ReviewRequest.java
│   │
│   └── response/                           # DTOs de réponse
```

```
│       ├── JwtResponse.java
│       ├── UserResponse.java
│       ├── PoolResponse.java
│       ├── TaskResponse.java
│       ├── EmployeeResponse.java
│       ├── ClientResponse.java
│       ├── ReservationResponse.java
│       ├── MessageResponse.java
│       ├── ConversationResponse.java
│       ├── NotificationResponse.java
│       ├── DashboardStatsResponse.java
│       └── ApiResponse.java                # Réponse générique
│
├── repository/                              # Repositories JPA
│   ├── UserRepository.java
│   ├── PoolRepository.java
│   ├── TaskRepository.java
│   ├── EmployeeRepository.java
│   ├── ClientRepository.java
│   ├── ReservationRepository.java
│   ├── MessageRepository.java
│   ├── ConversationRepository.java
│   ├── NotificationRepository.java
│   └── ReviewRepository.java
│
├── service/                                 # Services métier
│   ├── AuthService.java                    # Authentification
│   ├── UserService.java                    # Gestion utilisateurs
│   ├── PoolService.java                    # Gestion piscines
│   ├── TaskService.java                    # Gestion tâches
│   ├── EmployeeService.java                # Gestion employés
│   ├── ClientService.java                  # Gestion clients
│   ├── ReservationService.java             # Gestion réservations
│   ├── MessageService.java                 # Gestion messages
│   ├── NotificationService.java            # Gestion notifications
│   ├── DashboardService.java               # Statistiques dashboard
│   ├── ChatbotService.java                 # Intégration Ollama
│   └── EmailService.java                   # Envoi d'emails
│
├── controller/                              # Contrôleurs REST
│   ├── AuthController.java                 # POST /api/auth/login, /register
│   ├── UserController.java                 # GET/PUT /api/users
│   ├── PoolController.java                 # CRUD /api/pools
│   ├── TaskController.java                 # CRUD /api/tasks
│   ├── EmployeeController.java             # CRUD /api/employees
│   ├── ClientController.java               # CRUD /api/clients
│   ├── ReservationController.java          # CRUD /api/reservations
│   ├── MessageController.java              # CRUD /api/messages
│   ├── ConversationController.java         # /api/conversations
│   ├── NotificationController.java         # CRUD /api/notifications
│   ├── DashboardController.java            # GET /api/dashboard
│   └── ChatbotController.java              # POST /api/chatbot
│
├── security/                                # Sécurité
```

```
│   ├── JwtTokenProvider.java                # Génération/validation JWT
│   ├── JwtAuthenticationFilter.java         # Filtre JWT
│   ├── UserDetailsServiceImpl.java          # UserDetailsService
│   └── CustomUserDetails.java               # UserDetails personnalisé
│
├── exception/                                # Gestion des exceptions
│   ├── GlobalExceptionHandler.java          # @ControllerAdvice
│   ├── ResourceNotFoundException.java
│   ├── UnauthorizedException.java
│   ├── BadRequestException.java
│   └── ConflictException.java
│
└── util/                                     # Utilitaires
    ├── DateUtils.java
    └── ValidationUtils.java
```

📋 Liste des Classes Java

1. MODÈLES (Entités JPA)

User.java

```java
@Entity
@Table(name = "users")
public class User {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String email;

    @Column(nullable = false)
    private String password;

    private String name;
    private String phone;
    private String avatar;

    @Enumerated(EnumType.STRING)
    private UserRole role; // ADMIN, CLIENT, EMPLOYEE

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
}
```

Pool.java

```java
@Entity
@Table(name = "pools")
public class Pool {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String address;

    @Enumerated(EnumType.STRING)
    private PoolType type; // PUBLIC, PRIVEE, CLUB

    private LocalDate lastMaintenance;

    @Enumerated(EnumType.STRING)
    private PoolStatus status; // ACTIVE, MAINTENANCE, INACTIVE

    private String image;

    @ManyToOne
    @JoinColumn(name = "client_id")
    private Client owner;

    @OneToMany(mappedBy = "pool")
    private List<Task> tasks;

    @OneToMany(mappedBy = "pool")
    private List<Reservation> reservations;
}
```

Task.java

```java
@Entity
@Table(name = "tasks")
public class Task {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String description;
    private LocalDate date;
    private LocalTime time;

    @Enumerated(EnumType.STRING)
    private TaskStatus status; // A_FAIRE, EN_COURS, TERMINE

    @ManyToOne
    @JoinColumn(name = "pool_id")
    private Pool pool;
```

```
    @ManyToOne
    @JoinColumn(name = "employee_id")
    private Employee assignedEmployee;
}
```

Employee.java

```
@Entity
@Table(name = "employees")
public class Employee {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;

    private String name;
    private String phone;
    private String avatar;

    @ElementCollection
    private List<String> languages;

    @Enumerated(EnumType.STRING)
    private EmployeeStatus status; // DISPONIBLE, OCCUPE, ABSENT

    @OneToMany(mappedBy = "assignedEmployee")
    private List<Task> assignedTasks;
}
```

Client.java

```
@Entity
@Table(name = "clients")
public class Client {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;

    private String name;
    private String email;
    private String phone;
    private String address;
    private String type; // Particulier, Professionnel
```

```
    private String avatar;

    @OneToMany(mappedBy = "owner")
    private List<Pool> pools;

    @OneToMany(mappedBy = "client")
    private List<Reservation> reservations;
}
```

Reservation.java

```
@Entity
@Table(name = "reservations")
public class Reservation {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String service; // Nettoyage complet, Traitement eau, Inspection,
Réparation
    private LocalDate date;
    private LocalTime startTime;
    private LocalTime endTime;

    @Enumerated(EnumType.STRING)
    private ReservationStatus status; // EN_ATTENTE, CONFIRME, TERMINE, ANNULE

    @ManyToOne
    @JoinColumn(name = "pool_id")
    private Pool pool;

    @ManyToOne
    @JoinColumn(name = "client_id")
    private Client client;

    @ManyToOne
    @JoinColumn(name = "technician_id")
    private Employee technician;

    @OneToOne(mappedBy = "reservation")
    private Review review;
}
```

Message.java

```
@Entity
@Table(name = "messages")
public class Message {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;

    @Column(columnDefinition = "TEXT")
    private String text;

    private LocalDateTime timestamp;
    private boolean read;

    @ManyToOne
    @JoinColumn(name = "sender_id")
    private User sender;

    @ManyToOne
    @JoinColumn(name = "conversation_id")
    private Conversation conversation;
}
```

Conversation.java

```
@Entity
@Table(name = "conversations")
public class Conversation {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToMany
    @JoinTable(name = "conversation_participants")
    private List<User> participants;

    @OneToMany(mappedBy = "conversation")
    @OrderBy("timestamp DESC")
    private List<Message> messages;

    private LocalDateTime lastMessageAt;
}
```

Notification.java

```
@Entity
@Table(name = "notifications")
public class Notification {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Enumerated(EnumType.STRING)
    private NotificationType type;

    private String title;
```

```
    private String message;
    private LocalDateTime createdAt;
    private boolean read;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
}
```

Review.java

```
@Entity
@Table(name = "reviews")
public class Review {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Integer rating; // 1-5
    private String comment;
    private LocalDateTime createdAt;

    @OneToOne
    @JoinColumn(name = "reservation_id")
    private Reservation reservation;

    @ManyToOne
    @JoinColumn(name = "client_id")
    private Client client;
}
```

---

2. ÉNUMÉRATIONS

```
// enums/UserRole.java
public enum UserRole {
    ADMIN, CLIENT, EMPLOYEE
}

// enums/PoolStatus.java
public enum PoolStatus {
    ACTIVE, MAINTENANCE, INACTIVE
}

// enums/PoolType.java
public enum PoolType {
    PUBLIC, PRIVEE, CLUB
}

// enums/TaskStatus.java
```

```
public enum TaskStatus {
    A_FAIRE, EN_COURS, TERMINE
}

// enums/ReservationStatus.java
public enum ReservationStatus {
    EN_ATTENTE, CONFIRME, TERMINE, ANNULE
}

// enums/EmployeeStatus.java
public enum EmployeeStatus {
    DISPONIBLE, OCCUPE, ABSENT
}

// enums/NotificationType.java
public enum NotificationType {
    APPOINTMENT, MESSAGE, ALERT, POOL, SUCCESS, SYSTEM
}
```

---

3. DTOs

Request DTOs

```
// LoginRequest.java
public class LoginRequest {
    @Email @NotBlank
    private String email;
    @NotBlank
    private String password;
    private UserRole role;
}

// RegisterRequest.java
public class RegisterRequest {
    @NotBlank private String name;
    @Email @NotBlank private String email;
    @NotBlank @Size(min = 6) private String password;
    private String phone;
    @NotNull private UserRole role;
}

// ReservationRequest.java
public class ReservationRequest {
    @NotBlank private String service;
    @NotNull private LocalDate date;
    @NotNull private LocalTime startTime;
    @NotNull private LocalTime endTime;
    @NotNull private Long poolId;
    private Long technicianId;
}
```

```
// MessageRequest.java
public class MessageRequest {
    @NotNull private Long conversationId;
    @NotBlank private String text;
}
```

Response DTOs

```
// JwtResponse.java
public class JwtResponse {
    private String token;
    private String type = "Bearer";
    private UserResponse user;
}

// DashboardStatsResponse.java
public class DashboardStatsResponse {
    private Integer totalPools;
    private Integer activeTasks;
    private Integer availableEmployees;
    private Integer pendingReservations;
    private Integer totalClients;
}

// ApiResponse.java
public class ApiResponse<T> {
    private boolean success;
    private String message;
    private T data;
    private LocalDateTime timestamp;
}
```

---

4. REPOSITORIES

```
// UserRepository.java
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByEmail(String email);
    boolean existsByEmail(String email);
    List<User> findByRole(UserRole role);
}

// PoolRepository.java
@Repository
public interface PoolRepository extends JpaRepository<Pool, Long> {
    List<Pool> findByStatus(PoolStatus status);
    List<Pool> findByOwnerId(Long clientId);
```

```
    @Query("SELECT COUNT(p) FROM Pool p WHERE p.status = :status")
    Long countByStatus(@Param("status") PoolStatus status);
}

// TaskRepository.java
@Repository
public interface TaskRepository extends JpaRepository<Task, Long> {
    List<Task> findByStatus(TaskStatus status);
    List<Task> findByAssignedEmployeeId(Long employeeId);
    List<Task> findByPoolId(Long poolId);
    List<Task> findByDateOrderByTimeAsc(LocalDate date);
}

// ReservationRepository.java
@Repository
public interface ReservationRepository extends JpaRepository<Reservation, Long> {
    List<Reservation> findByClientId(Long clientId);
    List<Reservation> findByStatus(ReservationStatus status);
    List<Reservation> findByTechnicianId(Long technicianId);
    List<Reservation> findByPoolIdAndDate(Long poolId, LocalDate date);
}

// NotificationRepository.java
@Repository
public interface NotificationRepository extends JpaRepository<Notification, Long>
{
    List<Notification> findByUserIdOrderByCreatedAtDesc(Long userId);
    List<Notification> findByUserIdAndReadFalse(Long userId);
    @Modifying @Query("UPDATE Notification n SET n.read = true WHERE n.user.id =
:userId")
    void markAllAsRead(@Param("userId") Long userId);
}

// MessageRepository.java
@Repository
public interface MessageRepository extends JpaRepository<Message, Long> {
    List<Message> findByConversationIdOrderByTimestampAsc(Long conversationId);
}

// ConversationRepository.java
@Repository
public interface ConversationRepository extends JpaRepository<Conversation, Long>
{
    @Query("SELECT c FROM Conversation c JOIN c.participants p WHERE p.id =
:userId ORDER BY c.lastMessageAt DESC")
    List<Conversation> findByParticipantId(@Param("userId") Long userId);
}
```

---

5. SERVICES

```
// AuthService.java
@Service
public class AuthService {
    public JwtResponse login(LoginRequest request);
    public JwtResponse register(RegisterRequest request);
    public void logout(String token);
}

// DashboardService.java
@Service
public class DashboardService {
    public DashboardStatsResponse getStats();
    public List<TaskResponse> getTodayTasks();
    public List<ReservationResponse> getPendingReservations();
}

// ChatbotService.java
@Service
public class ChatbotService {
    private static final String OLLAMA_URL =
"http://localhost:11434/api/generate";

    public String generateResponse(String userMessage) {
        // Appel à l'API Ollama (llama3.2)
    }
}

// NotificationService.java
@Service
public class NotificationService {
    public void sendNotification(Long userId, NotificationType type, String title,
String message);
    public void markAsRead(Long notificationId);
    public void markAllAsRead(Long userId);
}
```

---

6. CONTROLLERS

```
// AuthController.java
@RestController
@RequestMapping("/api/auth")
public class AuthController {
    @PostMapping("/login")
    public ResponseEntity<JwtResponse> login(@Valid @RequestBody LoginRequest
request);

    @PostMapping("/register")
    public ResponseEntity<JwtResponse> register(@Valid @RequestBody
RegisterRequest request);
```

```
    @PostMapping("/logout")
    public ResponseEntity<ApiResponse> logout();
}

// PoolController.java
@RestController
@RequestMapping("/api/pools")
public class PoolController {
    @GetMapping
    public ResponseEntity<List<PoolResponse>> getAll();

    @GetMapping("/{id}")
    public ResponseEntity<PoolResponse> getById(@PathVariable Long id);

    @PostMapping
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<PoolResponse> create(@Valid @RequestBody PoolRequest
request);

    @PutMapping("/{id}")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<PoolResponse> update(@PathVariable Long id, @Valid
@RequestBody PoolRequest request);

    @DeleteMapping("/{id}")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<ApiResponse> delete(@PathVariable Long id);
}

// ReservationController.java
@RestController
@RequestMapping("/api/reservations")
public class ReservationController {
    @GetMapping
    public ResponseEntity<List<ReservationResponse>> getMyReservations();

    @PostMapping
    public ResponseEntity<ReservationResponse> create(@Valid @RequestBody
ReservationRequest request);

    @PutMapping("/{id}/cancel")
    public ResponseEntity<ApiResponse> cancel(@PathVariable Long id);

    @PutMapping("/{id}/confirm")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<ApiResponse> confirm(@PathVariable Long id);
}

// ChatbotController.java
@RestController
@RequestMapping("/api/chatbot")
public class ChatbotController {
    @PostMapping
```

```
    public ResponseEntity<ApiResponse<String>> chat(@RequestBody Map<String,
String> request);
}

// DashboardController.java
@RestController
@RequestMapping("/api/dashboard")
@PreAuthorize("hasRole('ADMIN')")
public class DashboardController {
    @GetMapping("/stats")
    public ResponseEntity<DashboardStatsResponse> getStats();
}
```

🌐 API Endpoints Récapitulatif

Méthode	Endpoint	Description	Rôle
POST	/api/auth/login	Connexion	Public
POST	/api/auth/register	Inscription	Public
GET	/api/pools	Liste des piscines	Tous
POST	/api/pools	Créer une piscine	Admin
PUT	/api/pools/{id}	Modifier une piscine	Admin
DELETE	/api/pools/{id}	Supprimer une piscine	Admin
GET	/api/tasks	Liste des tâches	Admin/Employee
POST	/api/tasks	Créer une tâche	Admin
PUT	/api/tasks/{id}/status	Changer status tâche	Admin/Employee
GET	/api/employees	Liste des employés	Admin
GET	/api/clients	Liste des clients	Admin
POST	/api/clients	Créer un client	Admin
GET	/api/reservations	Mes réservations	Client
POST	/api/reservations	Créer réservation	Client
PUT	/api/reservations/{id}/cancel	Annuler	Client
PUT	/api/reservations/{id}/confirm	Confirmer	Admin
GET	/api/conversations	Mes conversations	Tous
POST	/api/messages	Envoyer message	Tous
GET	/api/notifications	Mes notifications	Tous
PUT	/api/notifications/read-all	Tout marquer lu	Tous

Méthode	Endpoint	Description	Rôle
GET	/api/dashboard/stats	Stats dashboard	Admin
POST	/api/chatbot	Chat avec IA	Tous

🔧 Dépendances pom.xml

```
<dependencies>
    <!-- Spring Boot -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-websocket</artifactId>
    </dependency>

    <!-- JWT -->
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.11.5</version>
    </dependency>

    <!-- Database -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <!-- Lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

    <!-- MapStruct (DTO mapping) -->
```

```
    <dependency>
        <groupId>org.mapstruct</groupId>
        <artifactId>mapstruct</artifactId>
        <version>1.5.5.Final</version>
    </dependency>
</dependencies>
```

Schéma Base de Données