HealthTrack - 健康管理系统

项目简介

HealthTrack是一个基于Spring Boot + Vue.js的健康管理系统，支持普通用户和医生两种角色，提供用户注册、登录、健康信息管理等功能。

技术栈

后端技术

- **Java**: 8+
- **Spring Boot**: 2.7.18
- **Spring Security**: 5.7.x
- **Spring Data JPA**: 2.7.x
- **MySQL**: 8.0+
- **JWT**: 0.11.5
- **Maven**: 3.6+

前端技术

- **Vue.js**: 3.x
- **Element Plus**: 2.x
- **Vue Router**: 4.x
- **Pinia**: 2.x
- **Vite**: 4.x

项目结构

```
Project/
├── backend/                    # Spring Boot后端
│   ├── src/main/java/com/healthtrack/
│   │   ├── config/             # 配置类
│   │   ├── controller/         # REST控制器
│   │   ├── dto/               # 数据传输对象
│   │   ├── entity/            # JPA实体类
│   │   ├── repository/         # 数据访问层
│   │   ├── security/          # 安全配置
│   │   ├── service/           # 业务逻辑层
│   │   └── HealthTrackApplication.java  # 主启动类
│   ├── src/main/resources/
│   │   ├── application.yml    # 应用配置
│   │   └── sql/init.sql       # 数据库初始化脚本
│   └── pom.xml                # Maven依赖配置
├── frontend/                   # Vue.js前端
│   ├── src/
│   │   ├── views/             # 页面组件
│   │   ├── stores/            # 状态管理
```

```
│   │   ├── router/           # 路由配置
│   │   ├── App.vue           # 根组件
│   │   └── main.js           # 入口文件
│   ├── package.json          # 依赖配置
│   └── vite.config.js        # Vite配置
└── README.md                 # 项目说明文档
```

核心类说明

后端核心类

1. 实体类 (Entity)

User.java - 普通用户实体

- 实现`UserDetails`接口，支持Spring Security认证
- 字段：id, username, password, healthId, name, phone, createdAt, updatedAt
- 角色：ROLE_USER

Doctor.java - 医生实体

- 实现`UserDetails`接口，支持Spring Security认证
- 字段：id, username, password, licenseId, name, phone, specialization, verified, createdAt, updatedAt
- 角色：ROLE_DOCTOR

2. 数据传输对象 (DTO)

RegisterRequest.java - 注册请求对象

- 字段：username, password, name, healthId, phone, userType, licenseId, specialization
- 支持普通用户和医生注册

LoginRequest.java - 登录请求对象

- 字段：username, password, userType

AuthResponse.java - 认证响应对象

- 字段：token, type, id, email, name, role, healthId

3. 控制器 (Controller)

AuthController.java - 认证控制器

- `/api/auth/login` - 用户登录
- `/api/auth/register` - 用户注册
- `/api/auth/roles` - 获取可用角色

UserController.java - 用户控制器

- `/api/user/profile` - 获取用户信息

- `/api/user/check-username` - 检查用户名是否存在
- `/api/user/check-health-id` - 检查健康卡号是否存在
- `/api/user/check-phone` - 检查手机号是否存在

4. 服务层 (Service)

AuthService.java - 认证服务

- `login()` - 处理用户登录，支持User和Doctor类型
- `register()` - 处理用户注册，根据userType创建不同用户
- `registerUser()` - 注册普通用户
- `registerDoctor()` - 注册医生

UserService.java - 用户服务

- 实现`UserDetailsService`接口
- 提供用户CRUD操作和认证功能

DoctorService.java - 医生服务

- 实现`UserDetailsService`接口
- 提供医生CRUD操作和认证功能

UnifiedUserDetailsService.java - 统一用户详情服务

- 实现`UserDetailsService`接口
- 统一处理User和Doctor的认证查找

5. 数据访问层 (Repository)

UserRepository.java - 用户数据访问接口

- 继承`JpaRepository<User, Long>`
- 提供用户名、健康ID、手机号查询方法

DoctorRepository.java - 医生数据访问接口

- 继承`JpaRepository<Doctor, Long>`
- 提供用户名、执照ID、手机号查询方法

6. 安全配置 (Security)

WebSecurityConfig.java - Spring Security配置

- 配置CORS、CSRF、认证提供者
- 设置JWT过滤器
- 配置URL访问权限

JwtUtils.java - JWT工具类

- `generateJwtToken()` - 生成JWT token
- `getUserNameFromJwtToken()` - 从token中获取用户名

- `validateJwtToken()` - 验证token有效性

AuthTokenFilter.java - JWT认证过滤器

- 拦截请求，验证JWT token
- 设置SecurityContext

AuthEntryPointJwt.java - JWT认证入口点

- 处理认证失败的情况

7. 配置类 (Config)

DataInitializer.java - 数据初始化器

- 实现`CommandLineRunner`接口
- 在应用启动时创建测试数据

前端核心组件

1. 页面组件 (Views)

Login.vue - 登录页面

- 用户名/密码登录
- 表单验证
- 自动跳转到对应Dashboard

Register.vue - 注册页面

- 支持普通用户和医生注册
- 表单验证
- 用户名、密码、用户类型选择

Dashboard.vue - 普通用户仪表板

- 用户信息展示
- 健康数据管理

DoctorDashboard.vue - 医生仪表板

- 医生信息展示
- 患者管理功能

Profile.vue - 用户资料页面

- 个人信息编辑
- 密码修改

2. 状态管理 (Stores)

auth.js - 认证状态管理

- 用户登录状态
- JWT token管理
- 登录/注册/登出方法

3. 路由配置 (Router)

index.js - 路由配置

- 页面路由定义
- 路由守卫配置
- 根据用户角色跳转

数据库设计

用户表 (system_user)

```
CREATE TABLE system_user (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(512) NOT NULL,
    health_id BIGINT UNIQUE,
    name VARCHAR(128),
    phone VARCHAR(20) UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

医生表 (system_provider)

```
CREATE TABLE system_provider (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(512) NOT NULL,
    license_id BIGINT UNIQUE,
    name VARCHAR(128),
    phone VARCHAR(20) UNIQUE,
    role TINYINT,
    is_verified BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

环境要求

开发环境

- **JDK**: 8 或更高版本
- **Maven**: 3.6 或更高版本
- **Node.js**: 16 或更高版本
- **MySQL**: 8.0 或更高版本

系统要求

- **操作系统**: Windows, macOS, Linux
- **内存**: 至少 4GB RAM
- **磁盘空间**: 至少 2GB 可用空间

安装和启动

1. 克隆项目

```
git clone <repository-url>
cd Project
```

2. 数据库配置

安装MySQL

- 下载并安装MySQL 8.0+
- 创建数据库用户和密码

配置数据库连接

编辑 `backend/src/main/resources/application.yml`:

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/healthtrack?
useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: root
    password: 12345678  # 修改为你的MySQL密码
```

初始化数据库

```
# 登录MySQL
mysql -u root -p

# 执行初始化脚本
source backend/src/main/resources/sql/init.sql
```

3. 启动后端服务

方式一：使用Maven命令

```
cd backend
mvn clean package
mvn spring-boot:run
```

方式二：使用IDE

- 导入Maven项目到IDE
- 运行 `HealthTrackApplication.java`

方式三：使用JAR文件

```
cd backend
mvn clean package -DskipTests
java -jar target/healthtrack-backend-1.0.0.jar
```

后端服务将在 `http://localhost:8001` 启动

4. 启动前端服务

```
cd frontend
npm install
npm run dev
```

前端服务将在 `http://localhost:5173` 启动

使用说明

1. 访问应用

打开浏览器访问 `http://localhost:5173`

2. 注册用户

- 点击"注册"按钮
- 填写用户名、密码
- 选择用户类型（普通用户/医生）
- 点击"注册"

3. 登录系统

- 输入用户名和密码

- 点击"登录"
- 系统会根据用户类型跳转到对应Dashboard

4. 测试账户

系统会自动创建以下测试账户：

- **普通用户**: testuser / password123
- **医生用户**: testdoctor / password123

API接口文档

认证接口

用户登录

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "testuser",
  "password": "password123"
}
```

用户注册

```
POST /api/auth/register
Content-Type: application/json

{
  "username": "newuser",
  "password": "password123",
  "userType": "USER"
}
```

用户接口

获取用户信息

```
GET /api/user/profile
Authorization: Bearer <jwt-token>
```

检查用户名是否存在

```
GET /api/user/check-username?username=testuser
```

配置说明

JWT配置

```
jwt:
  secret:
mySuperSecureKey_ThisKeyMustBeAtLeast64BytesLong_UseItForHS512Algorithm!!!
  expiration: 86400000  # 24小时
```

日志配置

```
logging:
  level:
    com.healthtrack: DEBUG
    org.springframework.security: DEBUG
    org.hibernate.SQL: DEBUG
```

常见问题

1. 数据库连接失败

- 检查MySQL服务是否启动
- 验证数据库连接配置
- 确认数据库用户权限

2. JWT密钥错误

- 确保JWT密钥长度至少64字符
- 检查密钥配置是否正确

3. 前端跨域问题

- 后端已配置CORS，允许所有来源
- 检查前端请求URL是否正确

4. 用户认证失败

- 检查用户名和密码是否正确
- 确认用户类型选择正确
- 查看后端日志获取详细错误信息

开发指南

添加新功能

1. 在`entity`包中创建实体类
2. 在`repository`包中创建数据访问接口
3. 在`service`包中实现业务逻辑
4. 在`controller`包中创建REST接口
5. 在前端`views`包中创建页面组件

数据库迁移

- 修改实体类后，Hibernate会自动更新表结构
- 生产环境建议使用Flyway或Liquibase进行数据库版本管理

安全注意事项

- JWT密钥应使用环境变量配置
- 生产环境应使用HTTPS
- 定期更新依赖版本

许可证

本项目采用 MIT 许可证 - 查看 LICENSE 文件了解详情

贡献指南

1. Fork 项目
2. 创建功能分支 (`git checkout -b feature/AmazingFeature`)
3. 提交更改 (`git commit -m 'Add some AmazingFeature'`)
4. 推送到分支 (`git push origin feature/AmazingFeature`)
5. 打开 Pull Request

联系方式

如有问题或建议，请通过以下方式联系：

- 邮箱: [your-email@example.com]
- 项目Issues: [GitHub Issues链接]

注意: 这是一个教学项目，生产环境使用前请进行充分的安全评估和测试。