```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
```

```
        data: null
      }
    }

    const token = `mock-token-${user.id}-${Date.now()}`

    return {
      code: 200,
      message: '登录成功',
      data: {
        token,
        userInfo: {
          id: user.id,
          username: user.username,
          email: user.email,
          avatar: user.avatar,
          roles: user.roles
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
```

```
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},
```

```
// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
```

```
          code: 401,
          message: 'Token 不存在',
          data: null
        }
      }

      // 从旧 token 中解析用户 ID
      const userId = parseInt(token.split('-')[2]) || 1

      // 生成新的 token
      const newToken = `mock-token-${userId}-${Date.now()}`

      return {
        code: 200,
        message: 'Token 刷新成功',
        data: {
          token: newToken
        }
      }
    }
  },

  // 获取用户列表
  {
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
```

```
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
```

```
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
            email: user.email,
            avatar: user.avatar,
            roles: user.roles
```

```
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
```

```
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }
```

```
    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
```

```
      }
    }
  }
},

// 获取用户列表
{
  url: '/api/user/list',
  method: 'get',
  timeout: 500,
  response: ({ query }) => {
    const { page = 1, pageSize = 10, keyword = '' } = query

    // 生成模拟用户列表
    const mockUsers = Array.from({ length: 50 }, (_, index) => ({
      id: index + 1,
      username: Random.name(),
      email: Random.email(),
      avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
      phone: /^1[3-9]\d{9}$/,
      status: Random.boolean() ? 'active' : 'inactive',
      createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
      roles: Random.pick([['admin'], ['user'], ['guest']])
    }))

    // 关键词过滤
    let filteredUsers = mockUsers
    if (keyword) {
      filteredUsers = mockUsers.filter(u =>
        u.username.toLowerCase().includes(keyword.toLowerCase()) ||
        u.email.toLowerCase().includes(keyword.toLowerCase())
      )
    }

    // 分页
    const start = (page - 1) * pageSize
    const end = start + parseInt(pageSize)
    const list = filteredUsers.slice(start, end)

    return {
      code: 200,
      message: '获取成功',
      data: {
```

```
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
```

```
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
            email: user.email,
            avatar: user.avatar,
            roles: user.roles
          }
        }
      }
    }
  },

  // 获取用户信息
  {
    url: '/api/user/info',
    method: 'get',
    timeout: 300,
    response: ({ headers }) => {
      const token = headers.authorization?.replace('Bearer ', '')

      if (!token) {
        return {
          code: 401,
```

```
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID（简化处理）
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
```

```
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},
```

```
// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
      }
    }
  }
},

// 获取用户列表
{
  url: '/api/user/list',
  method: 'get',
  timeout: 500,
  response: ({ query }) => {
    const { page = 1, pageSize = 10, keyword = '' } = query

    // 生成模拟用户列表
    const mockUsers = Array.from({ length: 50 }, (_, index) => ({
      id: index + 1,
      username: Random.name(),
```

```
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
```

```
      code: 200,
      message: '登录成功',
      data: {
        token,
        userInfo: {
          id: user.id,
          username: user.username,
          email: user.email,
          avatar: user.avatar,
          roles: user.roles
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
```

```
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
```

```
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
```

```
      const userId = parseInt(token.split('-')[2]) || 1

      // 生成新的 token
      const newToken = `mock-token-${userId}-${Date.now()}`

      return {
        code: 200,
        message: 'Token 刷新成功',
        data: {
          token: newToken
        }
      }
    }
  },

  // 获取用户列表
  {
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }
```

```
      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
```

```
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
            email: user.email,
            avatar: user.avatar,
            roles: user.roles
          }
        }
      }
    }
  },

  // 获取用户信息
```

```
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID（简化处理）
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
```

```
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
```

```
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
      }
    }
  }
},

// 获取用户列表
{
```

```
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
```

```
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
```

```
      return {
        code: 400,
        message: '用户名或密码错误',
        data: null
      }
    }

    const token = `mock-token-${user.id}-${Date.now()}`

    return {
      code: 200,
      message: '登录成功',
      data: {
        token,
        userInfo: {
          id: user.id,
          username: user.username,
          email: user.email,
          avatar: user.avatar,
          roles: user.roles
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID（简化处理）
    const userId = parseInt(token.split('-')[2]) || 1
```

```
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
```

```
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')
```

```
    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
      }
    }
  }
},

// 获取用户列表
{
  url: '/api/user/list',
  method: 'get',
  timeout: 500,
  response: ({ query }) => {
    const { page = 1, pageSize = 10, keyword = '' } = query

    // 生成模拟用户列表
    const mockUsers = Array.from({ length: 50 }, (_, index) => ({
      id: index + 1,
      username: Random.name(),
      email: Random.email(),
      avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
      phone: /^1[3-9]\d{9}$/,
      status: Random.boolean() ? 'active' : 'inactive',
      createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
      roles: Random.pick([['admin'], ['user'], ['guest']])
```

```
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
```

```
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
```

```
          email: user.email,
          avatar: user.avatar,
          roles: user.roles
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},
```

```
// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
```

```
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
```

```
        message: 'Token 刷新成功',
        data: {
          token: newToken
        }
      }
    }
  },

  // 获取用户列表
  {
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
```

```
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
```

```
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
            email: user.email,
            avatar: user.avatar,
            roles: user.roles
          }
        }
      }
    }
  },

  // 获取用户信息
  {
    url: '/api/user/info',
    method: 'get',
    timeout: 300,
    response: ({ headers }) => {
      const token = headers.authorization?.replace('Bearer ', '')
```

```
    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID（简化处理）
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
```

```
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
```

```
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
      }
    }
  }
},

// 获取用户列表
{
  url: '/api/user/list',
  method: 'get',
  timeout: 500,
  response: ({ query }) => {
    const { page = 1, pageSize = 10, keyword = '' } = query

    // 生成模拟用户列表
```

```
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */
```

```
import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }
```

```
    const token = `mock-token-${user.id}-${Date.now()}`

    return {
      code: 200,
      message: '登录成功',
      data: {
        token,
        userInfo: {
          id: user.id,
          username: user.username,
          email: user.email,
          avatar: user.avatar,
          roles: user.roles
        }
      }
    }
  }
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
```

```
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
```

```
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
```

```
      }

      // 从旧 token 中解析用户 ID
      const userId = parseInt(token.split('-')[2]) || 1

      // 生成新的 token
      const newToken = `mock-token-${userId}-${Date.now()}`

      return {
        code: 200,
        message: 'Token 刷新成功',
        data: {
          token: newToken
        }
      }
    }
  },

  // 获取用户列表
  {
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
```

```
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
        }
      }
    }
  }
]
```

```
/**
 * 用户相关 Mock 数据
 */

import Mock from 'mockjs'
const { Random } = Mock
```

```
// 模拟用户数据
const users = [
  {
    id: 1,
    username: 'admin',
    password: '123456',
    email: 'admin@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/1?v=4',
    roles: ['admin'],
    permissions: ['*:*:*']
  },
  {
    id: 2,
    username: 'user',
```

```
    password: '123456',
    email: 'user@example.com',
    avatar: 'https://avatars.githubusercontent.com/u/2?v=4',
    roles: ['user'],
    permissions: ['user:read']
  }
]
```

```
export default [
  // 用户登录
  {
    url: '/api/user/login',
    method: 'post',
    timeout: 500,
    response: ({ body }) => {
      const { username, password } = body
      const user = users.find(u => u.username === username && u.password
=== password)

      if (!user) {
        return {
          code: 400,
          message: '用户名或密码错误',
          data: null
        }
      }

      const token = `mock-token-${user.id}-${Date.now()}`

      return {
        code: 200,
        message: '登录成功',
        data: {
          token,
          userInfo: {
            id: user.id,
            username: user.username,
            email: user.email,
            avatar: user.avatar,
            roles: user.roles
          }
        }
      }
    }
```

```
},

// 获取用户信息
{
  url: '/api/user/info',
  method: 'get',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: '未登录',
        data: null
      }
    }

    // 从 token 中解析用户 ID(简化处理)
    const userId = parseInt(token.split('-')[2]) || 1
    const user = users.find(u => u.id === userId) || users[0]

    return {
      code: 200,
      message: '获取成功',
      data: {
        id: user.id,
        username: user.username,
        email: user.email,
        avatar: user.avatar,
        roles: user.roles,
        permissions: user.permissions
      }
    }
  }
},

// 用户登出
{
  url: '/api/user/logout',
  method: 'post',
  timeout: 200,
  response: () => {
    return {
```

```
      code: 200,
      message: '登出成功',
      data: null
    }
  }
},

// 更新用户信息
{
  url: '/api/user/info',
  method: 'put',
  timeout: 500,
  response: ({ body }) => {
    return {
      code: 200,
      message: '更新成功',
      data: {
        ...body,
        updateTime: new Date().toISOString()
      }
    }
  }
},

// 修改密码
{
  url: '/api/user/password',
  method: 'post',
  timeout: 500,
  response: ({ body }) => {
    const { oldPassword, newPassword } = body

    if (!oldPassword || !newPassword) {
      return {
        code: 400,
        message: '参数错误',
        data: null
      }
    }

    // 模拟旧密码验证
    if (oldPassword !== '123456') {
      return {
        code: 400,
```

```
        message: '旧密码错误',
        data: null
      }
    }

    return {
      code: 200,
      message: '密码修改成功',
      data: null
    }
  }
},

// 刷新 Token
{
  url: '/api/user/refresh-token',
  method: 'post',
  timeout: 300,
  response: ({ headers }) => {
    const token = headers.authorization?.replace('Bearer ', '')

    if (!token) {
      return {
        code: 401,
        message: 'Token 不存在',
        data: null
      }
    }

    // 从旧 token 中解析用户 ID
    const userId = parseInt(token.split('-')[2]) || 1

    // 生成新的 token
    const newToken = `mock-token-${userId}-${Date.now()}`

    return {
      code: 200,
      message: 'Token 刷新成功',
      data: {
        token: newToken
      }
    }
  }
},
```

```
  // 获取用户列表
  {
    url: '/api/user/list',
    method: 'get',
    timeout: 500,
    response: ({ query }) => {
      const { page = 1, pageSize = 10, keyword = '' } = query

      // 生成模拟用户列表
      const mockUsers = Array.from({ length: 50 }, (_, index) => ({
        id: index + 1,
        username: Random.name(),
        email: Random.email(),
        avatar: `https://avatars.githubusercontent.com/u/${index +
1}?v=4`,
        phone: /^1[3-9]\d{9}$/,
        status: Random.boolean() ? 'active' : 'inactive',
        createTime: Random.datetime('yyyy-MM-dd HH:mm:ss'),
        roles: Random.pick([['admin'], ['user'], ['guest']])
      }))

      // 关键词过滤
      let filteredUsers = mockUsers
      if (keyword) {
        filteredUsers = mockUsers.filter(u =>
          u.username.toLowerCase().includes(keyword.toLowerCase()) ||
          u.email.toLowerCase().includes(keyword.toLowerCase())
        )
      }

      // 分页
      const start = (page - 1) * pageSize
      const end = start + parseInt(pageSize)
      const list = filteredUsers.slice(start, end)

      return {
        code: 200,
        message: '获取成功',
        data: {
          list,
          total: filteredUsers.length,
          page: parseInt(page),
          pageSize: parseInt(pageSize)
```

```
        }
      }
    }
  }
]
```