智能菜谱推荐系统 API 接口文档

前端 API 接口 (Vue)

以下是前端 Vue 项目中使用的 API 接口，基于 Axios 封装。

```
import axios from 'axios'
import { ElMessage } from 'element-plus'

// 创建 axios 实例
const request = axios.create({
  baseURL: '/api',
  timeout: 5000
})

// 请求拦截器
request.interceptors.request.use(
    config => {
      const token = localStorage.getItem('token');
      if (token) {
        config.headers['Authorization'] = `Bearer ${token}`;
      }
      // 在请求发送前添加 userId
      config.params = {
        ...config.params,
        userId: '2' // 替换为你的默认 userId
      };
      return config;
    },
    error => {
      return Promise.reject(error);
    }
);

// 响应拦截器
request.interceptors.response.use(
  response => {
    return response.data
  },
  error => {
    ElMessage.error(error.response?.data?.message || '请求失败')
    return Promise.reject(error)
  }
)

// 用户相关接口
export const userApi = {
  login: (data) => request.post('/users/login', data),
  logout: () => request.post('/users/logout'),
  register: (data) => request.post('/users/register', data),
  getUserInfo: () => request.get('/users/info'),
  updateUserInfo: (data) => request.put('/users/info', data),
  getUsers(params) {
        return request.get('/users', { params })
    }
}

// 菜品相关接口
export const recipeApi = {
  getCategories: () => request.get('/categories'),
  getRecipeList: (params) => request.get('/recipes', { params }),
  getRecipeById: (id) => request.get(`/recipes/${id}`),
  getRecipesByCategory: (categoryId) =>
request.get(`/recipes/category/${categoryId}`),
  searchRecipes: (keyword) => request.get('/recipes/search', {
    params: { keyword }  // 使用 params 传递查询参数
  })
}

// 购物车相关接口
export const cartApi = {
  getCartList: () => request.get('/cart/user'),
  addToCart: (data) => request.post('/cart', data),
  updateCartItem: (id, data) => request.put(`/cart/${id}`, data),
  removeFromCart: (id) => request.delete(`/cart/${id}`),
  clearCart: () => request.delete('/cart/user')
}

// 订单相关接口
export const orderApi = {
  getOrderList: () => request.get('/orders/user'),
  getOrderDetail: (id) => request.get(`/orders/${id}`),
  createOrder: (data) => request.post('/orders', data),
  updateOrderStatus: (id, status) => request.put(`/orders/${id}/status`, { status
})
}

// 评价相关接口
export const reviewApi = {
  getRecipeReviews: (recipeId) => request.get(`/reviews/recipe/${recipeId}`),
  addReview: (data) => request.post('/reviews', data),
  updateReview: (id, data) => request.put(`/reviews/${id}`, data),
  deleteReview: (id) => request.delete(`/reviews/${id}`),
    getReviewList(params) {
        return request.get('/reviews', { params })
    }
}

// 统计相关接口
export const statsApi = {
  getDashboardStats: () => request.get('/stats/dashboard')
}

// 收藏相关接口
export const favoriteApi = {
  getFavorites: () => request.get('/favorites/user'),
  addFavorite: (recipeId) => request.post(`/favorites/${recipeId}`),
  removeFavorite: (recipeId) => request.delete(`/favorites/${recipeId}`),
  checkFavorite: (recipeId) => request.get(`/favorites/check/${recipeId}`)
```

```
}

// 推荐相关接口
export const recommendApi = {
  getRecommendedRecipes: () => request.get('/recommend')
}

// 评论相关接口
export const commentApi = {
  // 获取菜谱评论列表
  getCommentsByRecipe: (recipeId) => {
    return request({
      url: `/api/comments/recipe/${recipeId}`,
      method: 'get'
    })
  },

  // 添加评论
  addComment: (data) => {
    return request({
      url: '/api/comments',
      method: 'post',
      data: {
        recipeId: data.recipeId,
        content: data.content,
        rating: data.rating
      }
    })
  },

  // 删除评论
  deleteComment: (commentId) => {
    return request({
      url: `/api/comments/${commentId}`,
      method: 'delete'
    })
  },

  // 点赞评论
  likeComment: (commentId) => {
    return request({
      url: `/api/comments/${commentId}/like`,
      method: 'post'
    })
  },

  // 取消点赞评论
  unlikeComment: (commentId) => {
    return request({
      url: `/api/comments/${commentId}/like`,
      method: 'delete'
    })
  },

  // 检查是否已点赞评论
  checkCommentLike: (commentId) => {
    return request({
      url: `/api/comments/${commentId}/like`,
      method: 'get'
    })
  }
}



## 后端 Controller 和 Mapper 接口建议 (Spring Boot)

以下是根据前端 API 接口，建议的后端 Spring Boot Controller 和 Mapper 接口:

### 1. UserController

*   `@PostMapping("/users/login")`: 用户登录
    *   Controller 方法: `login(User user)`
    *   Mapper 接口: `UserMapper.getUserByUsername(String username)`
*   `@PostMapping("/users/logout")`: 用户登出 (通常前端处理，后端可以记录日志)
    *   Controller 方法: `logout()`
*   `@PostMapping("/users/register")`: 用户注册
    *   Controller 方法: `register(User user)`
    *   Mapper 接口: `UserMapper.insert(User user)`
*   `@GetMapping("/users/info")`: 获取用户信息
    *   Controller 方法: `getUserInfo(@RequestHeader("Authorization") String
token)`
    *   Mapper 接口: `UserMapper.selectById(Integer id)`
*   `@PutMapping("/users/info")`: 更新用户信息
    *   Controller 方法: `updateUserInfo(@RequestHeader("Authorization") String
token, User user)`
    *   Mapper 接口: `UserMapper.updateById(User user)`
*   `@GetMapping("/users")`: 获取用户列表 (带分页)
    *   Controller 方法: `getUsers(Pageable pageable, @RequestParam Map<String,
Object> params)`
    *   Mapper 接口: `UserMapper.selectUserList(Page<User> page, @Param("params")
Map<String, Object> params)`

### 2. RecipeController

*   `@GetMapping("/categories")`: 获取菜谱分类列表
    *   Controller 方法: `getCategories()`
    *   Mapper 接口: `CategoryMapper.selectList(Wrapper queryWrapper)`
*   `@GetMapping("/recipes")`: 获取菜谱列表 (带分页和查询参数)
    *   Controller 方法: `getRecipeList(Pageable pageable, @RequestParam
Map<String, Object> params)`
    *   Mapper 接口: `RecipeMapper.selectRecipeList(Page<Recipe> page,
@Param("params") Map<String, Object> params)`
*   `@GetMapping("/recipes/{id}")`: 获取菜谱详情
    *   Controller 方法: `getRecipeById(@PathVariable Integer id)`
    *   Mapper 接口: `RecipeMapper.selectById(Integer id)`
*   `@GetMapping("/recipes/category/{categoryId}")`: 获取指定分类下的菜谱列表
    *   Controller 方法: `getRecipesByCategory(@PathVariable Integer categoryId)`
    *   Mapper 接口: `RecipeMapper.selectList(Wrapper queryWrapper)`
*   `@GetMapping("/recipes/search")`: 搜索菜谱
    *   Controller 方法: `searchRecipes(@RequestParam String keyword)`
    *   Mapper 接口: `RecipeMapper.searchRecipes(String keyword)`
```

```
### 3. CartController

*   `@GetMapping("/cart/user")`：获取用户购物车列表
    *   Controller 方法：`getCartList(@RequestHeader("Authorization") String token)`
    *   Mapper 接口：`CartMapper.selectCartListByUserId(Integer userId)`
*   `@PostMapping("/cart")`：添加菜品到购物车
    *   Controller 方法：`addToCart(@RequestHeader("Authorization") String token, @RequestBody Cart cart)`
    *   Mapper 接口：`CartMapper.insert(Cart cart)`
*   `@PutMapping("/cart/{id}")`：更新购物车商品数量
    *   Controller 方法：`updateCartItem(@PathVariable Integer id, @RequestBody Cart cart)`
    *   Mapper 接口：`CartMapper.updateById(Cart cart)`
*   `@DeleteMapping("/cart/{id}")`：从购物车移除商品
    *   Controller 方法：`removeFromCart(@PathVariable Integer id)`
    *   Mapper 接口：`CartMapper.deleteById(Integer id)`
*   `@DeleteMapping("/cart/user")`：清空用户购物车
    *   Controller 方法：`clearCart(@RequestHeader("Authorization") String token)`
    *   Mapper 接口：`CartMapper.delete(Wrapper queryWrapper)`

### 4. OrderController

*   `@GetMapping("/orders/user")`：获取用户订单列表
    *   Controller 方法：`getOrderList(@RequestHeader("Authorization") String token)`
    *   Mapper 接口：`OrderMapper.selectOrderListByUserId(Integer userId)`
*   `@GetMapping("/orders/{id}")`：获取订单详情
    *   Controller 方法：`getOrderDetail(@PathVariable Integer id)`
    *   Mapper 接口：`OrderMapper.selectOrderWithDetailById(Integer id)`
*   `@PostMapping("/orders")`：创建订单
    *   Controller 方法：`createOrder(@RequestHeader("Authorization") String token, @RequestBody Order order)`
    *   Mapper 接口：`OrderMapper.insert(Order order)`, `OrderDetailMapper.insertBatch(List<OrderDetail> orderDetails)`
*   `@PutMapping("/orders/{id}/status")`：更新订单状态
    *   Controller 方法：`updateOrderStatus(@PathVariable Integer id, @RequestBody Map<String, Object> params)`
    *   Mapper 接口：`OrderMapper.updateById(Order order)`

### 5. ReviewController

*   `@GetMapping("/reviews/recipe/{recipeId}")`：获取菜谱评价列表
    *   Controller 方法：`getRecipeReviews(@PathVariable Integer recipeId)`
    *   Mapper 接口：`ReviewMapper.selectReviewListByRecipeId(Integer recipeId)`
*   `@PostMapping("/reviews")`：添加评价
    *   Controller 方法：`addReview(@RequestHeader("Authorization") String token, @RequestBody Review review)`
    *   Mapper 接口：`ReviewMapper.insert(Review review)`
*   `@PutMapping("/reviews/{id}")`：更新评价
    *   Controller 方法：`updateReview(@PathVariable Integer id, @RequestHeader("Authorization") String token, @RequestBody Review review)`
    *   Mapper 接口：`ReviewMapper.updateById(Review review)`
*   `@DeleteMapping("/reviews/{id}")`：删除评价
```

```
    *   Controller 方法：`deleteReview(@PathVariable Integer id, @RequestHeader("Authorization") String token)`
    *   Mapper 接口：`ReviewMapper.deleteById(Integer id)`
*   `@GetMapping("/reviews")`：获取评价列表（带分页）
    *   Controller 方法：`getReviewList(Pageable pageable, @RequestParam Map<String, Object> params)`
    *   Mapper 接口：`ReviewMapper.selectReviewList(Page<Review> page, @Param("params") Map<String, Object> params)`

### 6. FavoriteController

*   `@GetMapping("/favorites/user")`：获取用户收藏列表
    *   Controller 方法：`getFavorites(@RequestHeader("Authorization") String token)`
    *   Mapper 接口：`FavoriteMapper.selectFavoriteListByUserId(Integer userId)`
*   `@PostMapping("/favorites/{recipeId}")`：添加菜谱到收藏
    *   Controller 方法：`addFavorite(@RequestHeader("Authorization") String token, @PathVariable Integer recipeId)`
    *   Mapper 接口：`FavoriteMapper.insert(Favorite favorite)`
*   `@DeleteMapping("/favorites/{recipeId}")`：从收藏移除菜谱
    *   Controller 方法：`removeFavorite(@RequestHeader("Authorization") String token, @PathVariable Integer recipeId)`
    *   Mapper 接口：`FavoriteMapper.delete(Wrapper queryWrapper)`
*   `@GetMapping("/favorites/check/{recipeId}")`：检查菜谱是否已收藏
    *   Controller 方法：`checkFavorite(@RequestHeader("Authorization") String token, @PathVariable Integer recipeId)`
    *   Mapper 接口：`FavoriteMapper.selectCount(Wrapper queryWrapper)`

### 7. RecommendController

*   `@GetMapping("/recommend")`：获取推荐菜谱列表
    *   Controller 方法：`getRecommendedRecipes(@RequestHeader("Authorization") String token)`
    *   Mapper 接口：(根据推荐算法实现，可能需要多个 Mapper 接口)

### 8. CommentController

*   `@GetMapping("/api/comments/recipe/{recipeId}")`：获取菜谱评论列表
    *   Controller 方法：`getCommentsByRecipe(@PathVariable Integer recipeId)`
    *   Mapper 接口：`CommentMapper.selectCommentListByRecipeId(Integer recipeId)`
*   `@PostMapping("/api/comments")`：添加评论
    *   Controller 方法：`addComment(@RequestHeader("Authorization") String token, @RequestBody Comment comment)`
    *   Mapper 接口：`CommentMapper.insert(Comment comment)`
*   `@DeleteMapping("/api/comments/{commentId}")`：删除评论
    *   Controller 方法：`deleteComment(@RequestHeader("Authorization") String token, @PathVariable Integer commentId)`
    *   Mapper 接口：`CommentMapper.deleteById(Integer commentId)`
*   `@PostMapping("/api/comments/{commentId}/like")`：点赞评论
    *   Controller 方法：`likeComment(@RequestHeader("Authorization") String token, @PathVariable Integer commentId)`
    *   Mapper 接口：`CommentLikeMapper.insert(CommentLike commentLike)`
*   `@DeleteMapping("/api/comments/{commentId}/like")`：取消点赞评论
    *   Controller 方法：`unlikeComment(@RequestHeader("Authorization") String token, @PathVariable Integer commentId)`
    *   Mapper 接口：`CommentLikeMapper.delete(Wrapper queryWrapper)`
*   `@GetMapping("/api/comments/{commentId}/like")`：检查是否已点赞评论
    *   Controller 方法：`checkCommentLike(@RequestHeader("Authorization") String token, @PathVariable Integer commentId)`
    *   Mapper 接口：`CommentLikeMapper.selectCount(Wrapper queryWrapper)`

**说明：**

*   以上 Controller 和 Mapper 接口仅为建议，您可以根据实际业务需求进行调整。
*   `Wrapper` 是 Mybatis-Plus 提供的条件构造器，用于构建 SQL 查询条件。
*   `Pageable` 是 Spring Data 提供的分页接口，用于实现分页查询。
*   `@Param` 注解用于在 Mapper 接口中传递参数。
*   `@RequestHeader("Authorization") String token` 用于从请求头中获取 JWT Token。
*   需要根据实际情况添加 Service 层，用于处理业务逻辑。
*   需要添加异常处理、日志记录等功能。
```