

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**



**TÊN HỌC PHẦN: CÁC CÔNG NGHỆ PHẦN MỀM MỚI**

**ĐỒ ÁN XÂY DỰNG API CHO WEB QUẢN LÝ DỊCH VỤ RỬA XE**

**Giáo viên hướng dẫn: Nguyễn Minh Đạo**

**Sinh viên thực hiện:**

**Nguyễn Nhật Tính- 20110576**

**TP.Hồ Chí Minh, Tháng 05, Năm 2023**

**LỜI CẢM ƠN**

Lời đầu tiên nhóm em xin gửi lời cảm ơn đến giảng viên Nguyễn Minh Đạo – người đã dẫn dắt, giảng dạy chúng em xuyên suốt 15 tuần vừa qua, và cuối cùng cũng đến lúc nói lời tạm biệt bằng một project cuối kì. Để hoàn thành tốt đề tài và bài báo cáo này nhóm chúng em phải được sự trợ giúp của thầy để thấy được những lỗi sai, những cái cần phải chỉnh chu cho bản project cuối cùng, và rồi từ những cái sai đó để đưa nhóm đến những sự hoàn thiện cuối cùng. Nhờ những lời khuyên từ kinh nghiệm giảng dạy của thầy để định hướng cho chúng em đi đúng với yêu cầu đã chọn, một người thầy luôn đưa ra những lời giải đáp một cách kỹ càng và những lời góp ý hết sức tuyệt vời để nhóm có thể chỉnh sửa kịp thời giúp cải thiện được nhược điểm cũng như là hoàn thành tốt bài dự án này và đạt được yêu cầu cũng như thời hạn đã đề ra.

Chúng em cũng xin gửi đôi lời cảm ơn chân thành đến các quý thầy cô trong khoa Đào Tạo Chất Lượng Cao nói chung và ngành Công Nghệ Thông Tin nói riêng đã tận tình truyền đạt những kiến thức bổ ích, cần thiết để giúp chúng em có nền tảng để làm ra được đề tài báo cáo này một cách thành công tốt đẹp. Cùng với đó xin cảm ơn những người bạn đồng hành trong và ngoài lớp đã cung cấp nhiều thông tin và kiến thức hữu hiệu giúp chúng em có thể hoàn thiện hơn

**Mục Lục**

[PHẦN 1: MỞ ĐẦU………………………………………………………………………………….…](#_Toc135573814)1

[PHẦN 2: CƠ SỞ LÝ THUYẾT THỰC HIỆN PROJECT……………………………………………](#_Toc135573815)..3

[PHẦN 3: PHÂN TÍCH - THIẾT KẾ CƠ SỞ DỮ LIỆU……………………………………………….5](#_Toc135573816)

[**3.1.** **Ý tưởng** 5](#_Toc135573817)

[**3.2.** **Thiết kế cơ sở dữ liệu** 6](#_Toc135573818)

[PHẦN 4: USECASE VÀ LUỒNG CHẠY HỆ THỐNG……………………………………………..10](#_Toc135573819)

[4.1. Usecase hệ thống cho admin……………………………………………………………………...10](#_Toc135573820)

[4.2. Usecase luồng chạy của hệ thống cho user……………………………………………………….10](#_Toc135573821)

[PHẦN 5: CODE THỰC THI CÁC API CỦA CHƯƠNG TRÌNH…………………………………...11](#_Toc135573822)

[**5.1. User có sẵn khi chạy chương trình** 11](#_Toc135573823)

[**5.2. Đăng nhập bằng các tài khoản user có sẵn** 12](#_Toc135573824)

[**5.3. Tạo User** 13](#_Toc135573825)

[**5.4. Tạo các tiện nghi** 14](#_Toc135573826)

[**5.5. Tạo khách sạn** 14](#_Toc135573827)

[**5.6. Tạo thanh toán** 15](#_Toc135573828)

[**5.7. Tạo đặt chỗ** 15](#_Toc135573829)

[**5.8.Tạo phòng** 15](#_Toc135573830)

[**5.9. Update tiện nghi** 16](#_Toc135573831)

[**5.10. Update khách sạn** 16](#_Toc135573832)

[**5.11. Update thanh toán** 17](#_Toc135573833)

[**5.12. Update đặt chỗ** 17](#_Toc135573834)

[**5.13. Update phòng** 18](#_Toc135573835)

[**5.14. Update user** 18](#_Toc135573836)

[**5.15. Delete tiện nghi** 19](#_Toc135573837)

[**5.16. Delete khách sạn** 19](#_Toc135573838)

[**5.17. Delete thanh toán** 19](#_Toc135573839)

[**5.18. Delete đặt chỗ** 20](#_Toc135573840)

[**5.19. Delete phòng** 20](#_Toc135573841)

[**5.20. Delete user** 20](#_Toc135573842)

[**5.21. Tìm tất cả tiện nghi** 21](#_Toc135573843)

[**5.22. Tìm tất cả khách sạn** 21](#_Toc135573844)

[**5.23. Tìm tất cả thanh toán** 22](#_Toc135573845)

[**5.24. Tìm tất cả đặt chỗ** 22](#_Toc135573846)

[**5.25. Tìm tất cả phòng** 23](#_Toc135573847)

[**5.26. Tìm tất cả user** 23](#_Toc135573848)

[**5.27. Tìm tiện nghi theo id** 24](#_Toc135573849)

[**5.28. Tìm khách sạn theo id** 24](#_Toc135573850)

[**5.29. Tìm thanh toán theo id** 24](#_Toc135573851)

[**5.30. Tìm đặt chỗ theo id** 25](#_Toc135573852)

[**5.31. Tìm phòng theo id** 25](#_Toc135573853)

[**5.32. Tìm user theo id** 26](#_Toc135573854)

[**5.33. Liên kết giữa Hotel và Amenities** 26](#_Toc135573855)

[**5.34. Liên kết giữa Amenities và Hotel** 27](#_Toc135573856)

[**5.35. Liên kết giữa Hotel và Room** 28](#_Toc135573857)

[**5.36. Liên kết giữa Payment và Reservation** 28](#_Toc135573858)

[**5.37. Liên kết giữa Room và Reservation** 29](#_Toc135573859)

[**5.38. Liên kết giữa User và Reservation** 30](#_Toc135573860)

[**5.39. Liên kết giữa User và Payment** 31](#_Toc135573861)

[**5.40. Đổi status của chính bản thân user đang đăng nhập** 31](#_Toc135573862)

[**5.41. Đổi status của user theo id** 32](#_Toc135573863)

[**5.42. Các Filter cho các bảng** 32](#_Toc135573864)

[PHẦN 6: KẾT LUẬN………………………………………………………………………………...36](#_Toc135573865)

[TÀI LIỆU THAM KHẢO…………………………………………………………………………….37](#_Toc135573866)

# TÓM TẮT

Xây dựng API trang web quản lý dịch vụ rửa xe Java SpringBoot , MongoDB đảm bảo các chức năng sau :

* Người dùng có thể :
* Đăng nhập, tạo tài khoản, thay đổi thông tin cá nhân
* Thêm phương tiện, sửa và xóa phương tiện của bản thân
* Xem danh sách các dịch vụ
* Tạo giao dịch
* Xem các giao dịch đã thực hiện
* Quản trị viên có thể :
* Đăng nhập
* Quản lý bảng User (liệt kê (thường + phân trang) ,thêm, xóa, tìm kiếm, cập nhật)
* Quản lý bảng Vehicle (liệt kê (thường+phân trang) ,thêm ,xóa, tìm kiếm, cập nhật)
* Quản lý bảng Transaction (liệt kê (thường+phân trang), thêm, xóa, tìm kiếm, cập nhật, xác nhận thanh toán)
* Quản lý bảng Service (liệt kê (thường+phân trang), thêm, xóa, tìm kiếm, cập nhập

# CƠ SỞ DỮ LIỆU

## Tổng quát

1. Bảng **User** :

* id: khóa chính, định danh duy nhất cho người dùng
* name: tên người dùng
* email: tên tài khoản email
* password: mật khẩu
* dienThoai: số điện thoại của người dùng
* roles: quyền của người dùng (Admin or User)
* trangThai: trạng thái hoạt động của tài khoản (True of False)

1. Bảng **Vehicle** :

* id: khóa chính, định danh duy nhất cho phương tiện
* name: tên phương tiện
* user\_id: khóa ngoại tham chiếu đến bảng **User** để xác định khách hàng sở hữu xe

1. Bảng **Service** :

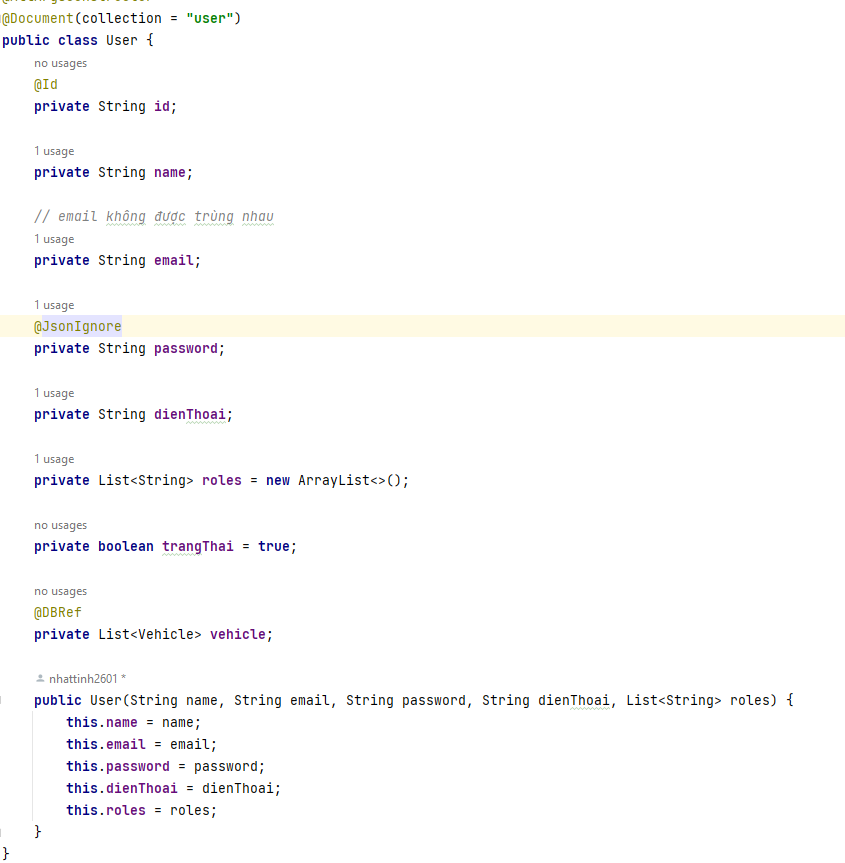
* id: khóa chính, định danh duy nhất cho dịch vụ
* name: tên dịch vụ
* description: miêu tả dịch vụ
* price: giá của dịch vụ

1. Bảng **Transaction**:

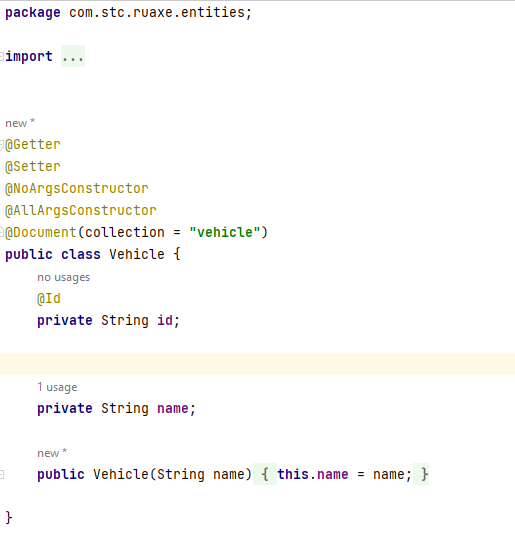
* id: khóa chính, định danh duy nhất cho giao dịch
* user\_id: khóa ngoại tham chiếu đến bảng **User** để xác định khách hàng giao dịch
* vehicle\_id: khóa ngoại tham chiếu đến bảng **Vehicle** để xác định phương tiện nào
* service\_id: khóa ngoại tham chiếu đến bảng **Service** để xác định dịch vụ sử dụng
* amount: chi phí thanh toán
* dateTransaction: ngày thực hiện giao dịch
* trangThai: xác định đã thanh toán hay chưa

## Đoạn mã tạo Entity trong code

1. Bảng **User:**



1. Bảng **Vehicle:**



1. Bảng **Service:**

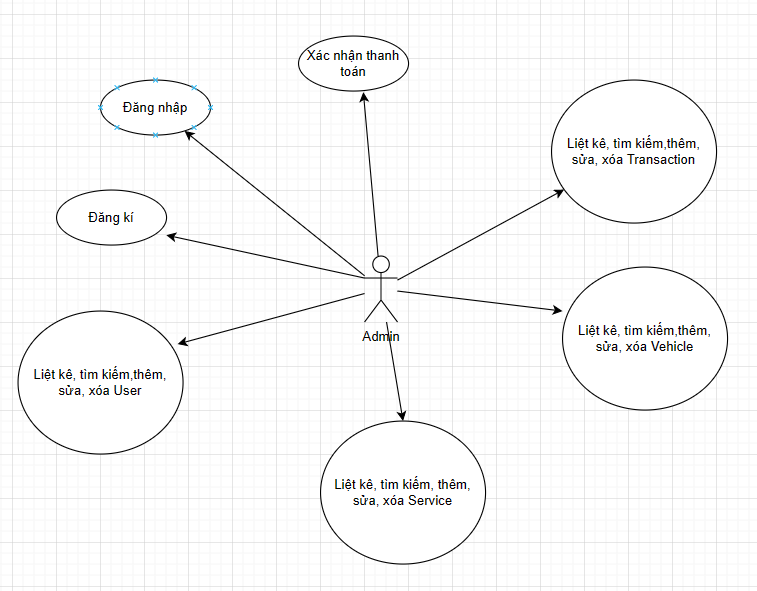


1. Bảng **Transaction:**

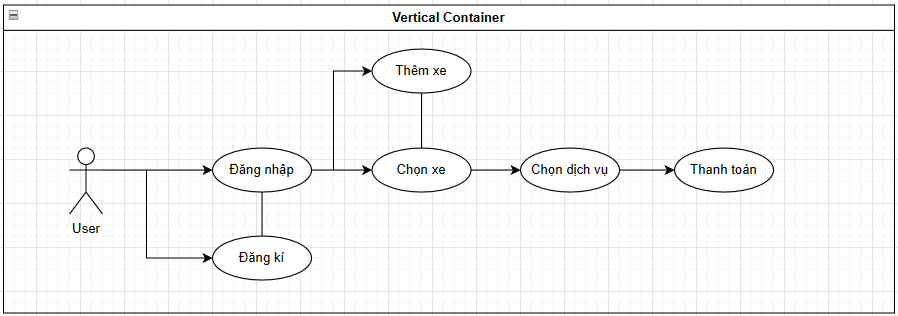


# USECASE VÀ LUỒNG CHẠY HỆ THỐNG

## Usecase hệ thống cho admin

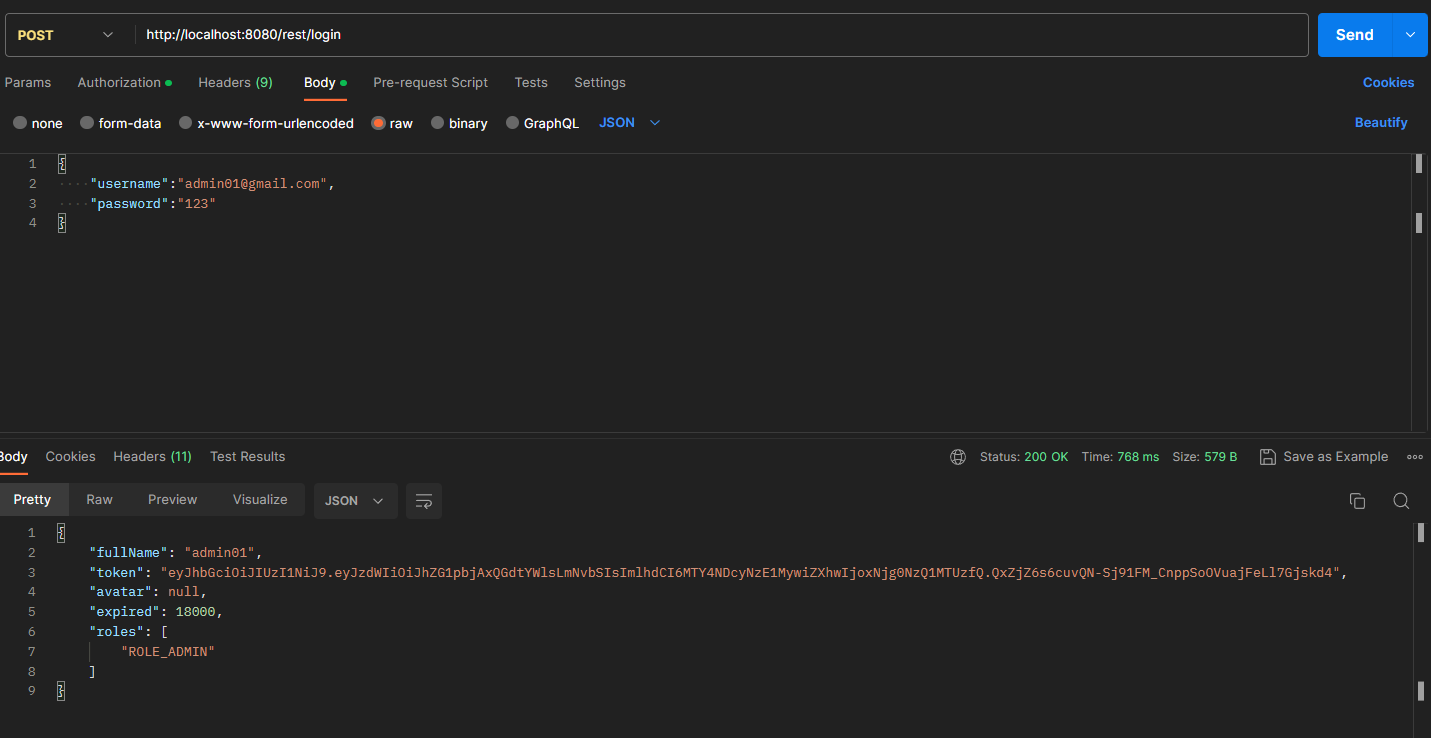


## Usecase luồng chạy của hệ thống cho user

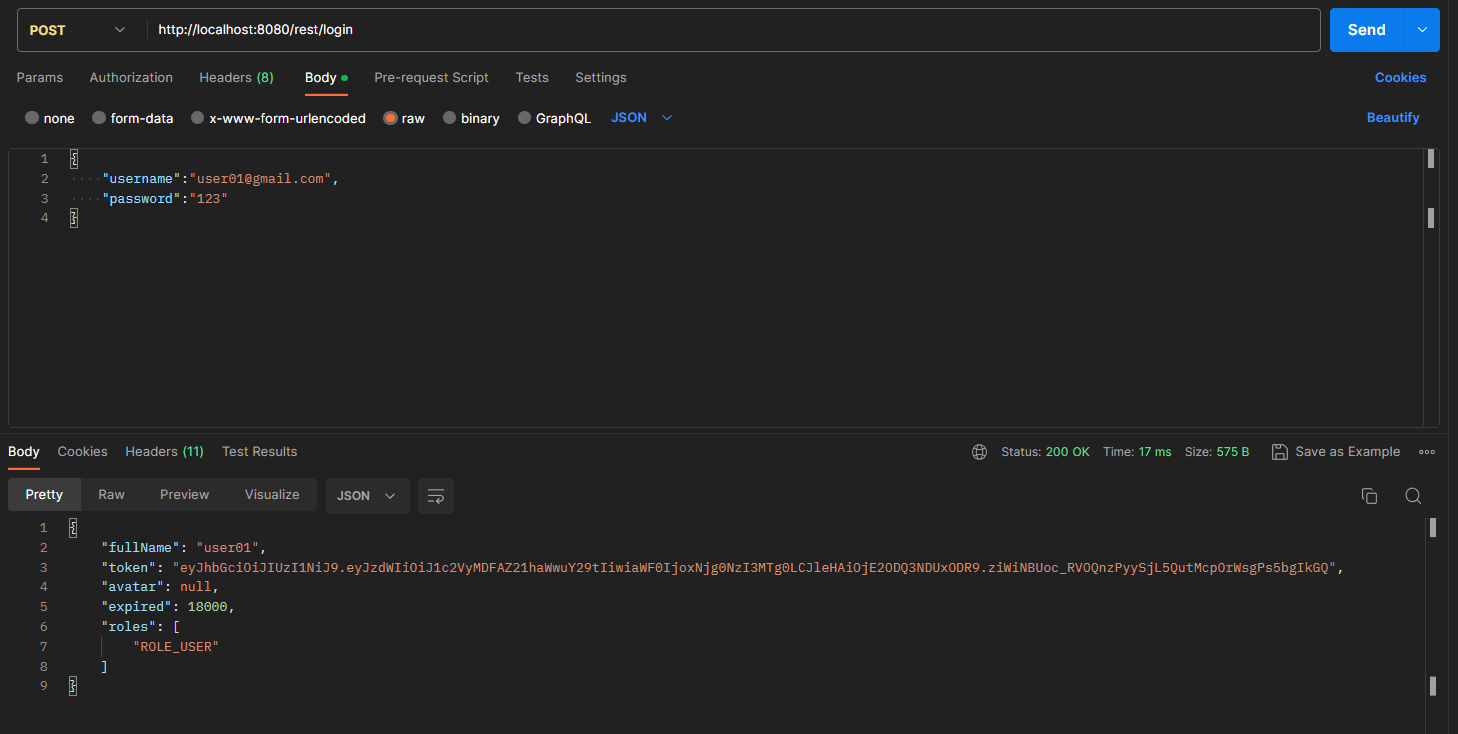


# CODE THỰC THI CÁC API CỦA CHƯƠNG TRÌNH

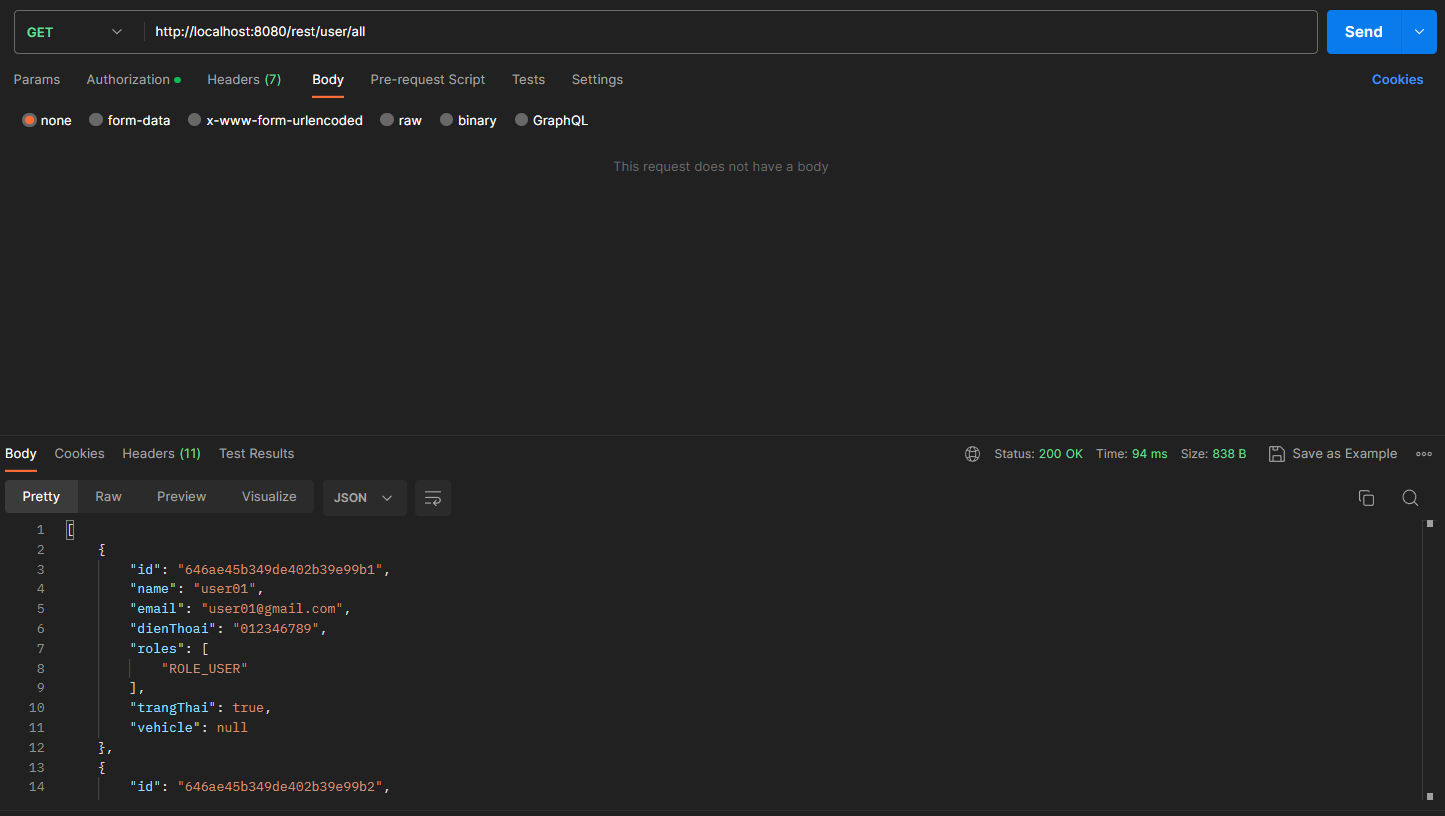
## Đăng nhập dưới quyền admin



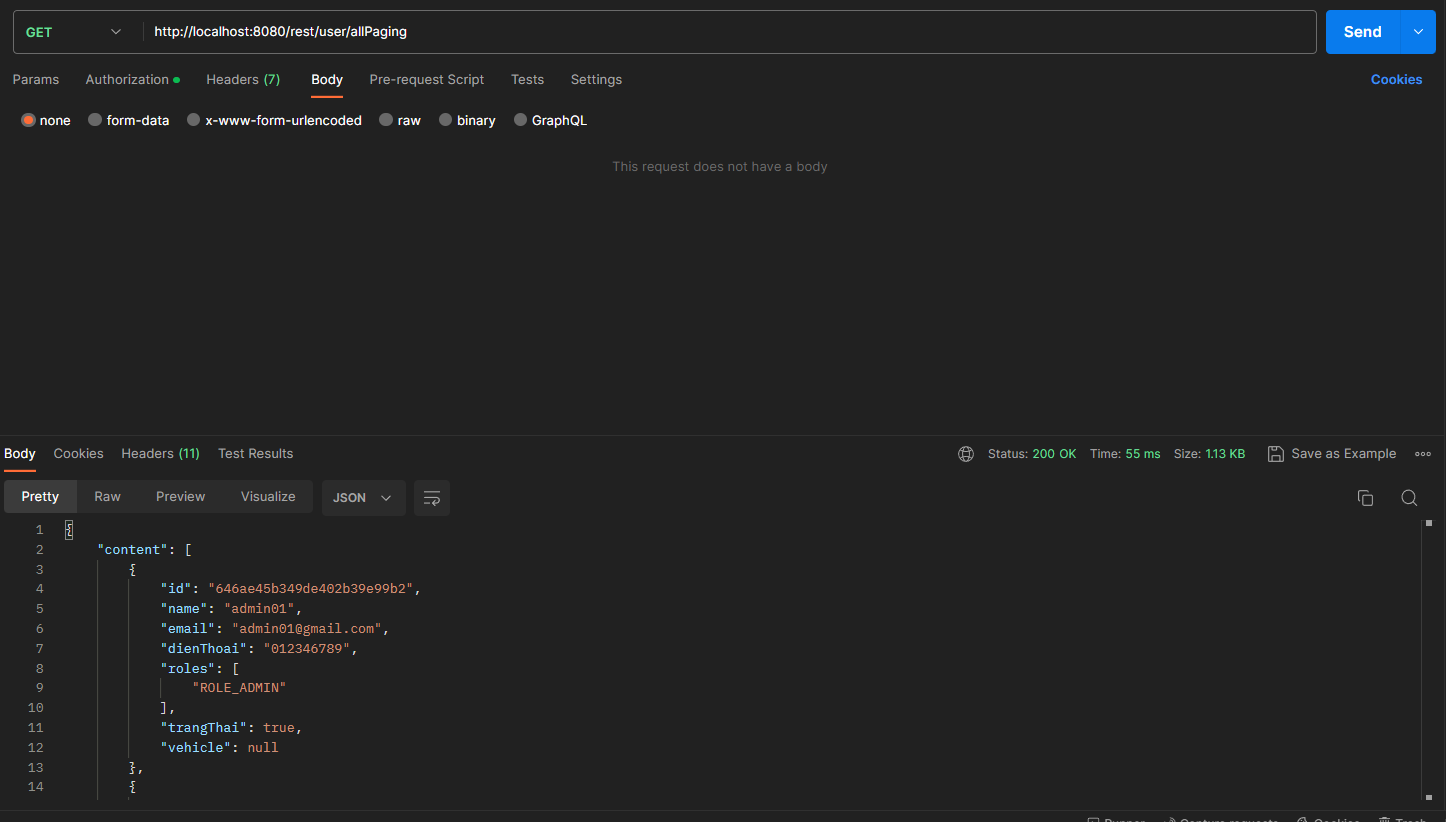
## Đăng nhập dưới quyền User



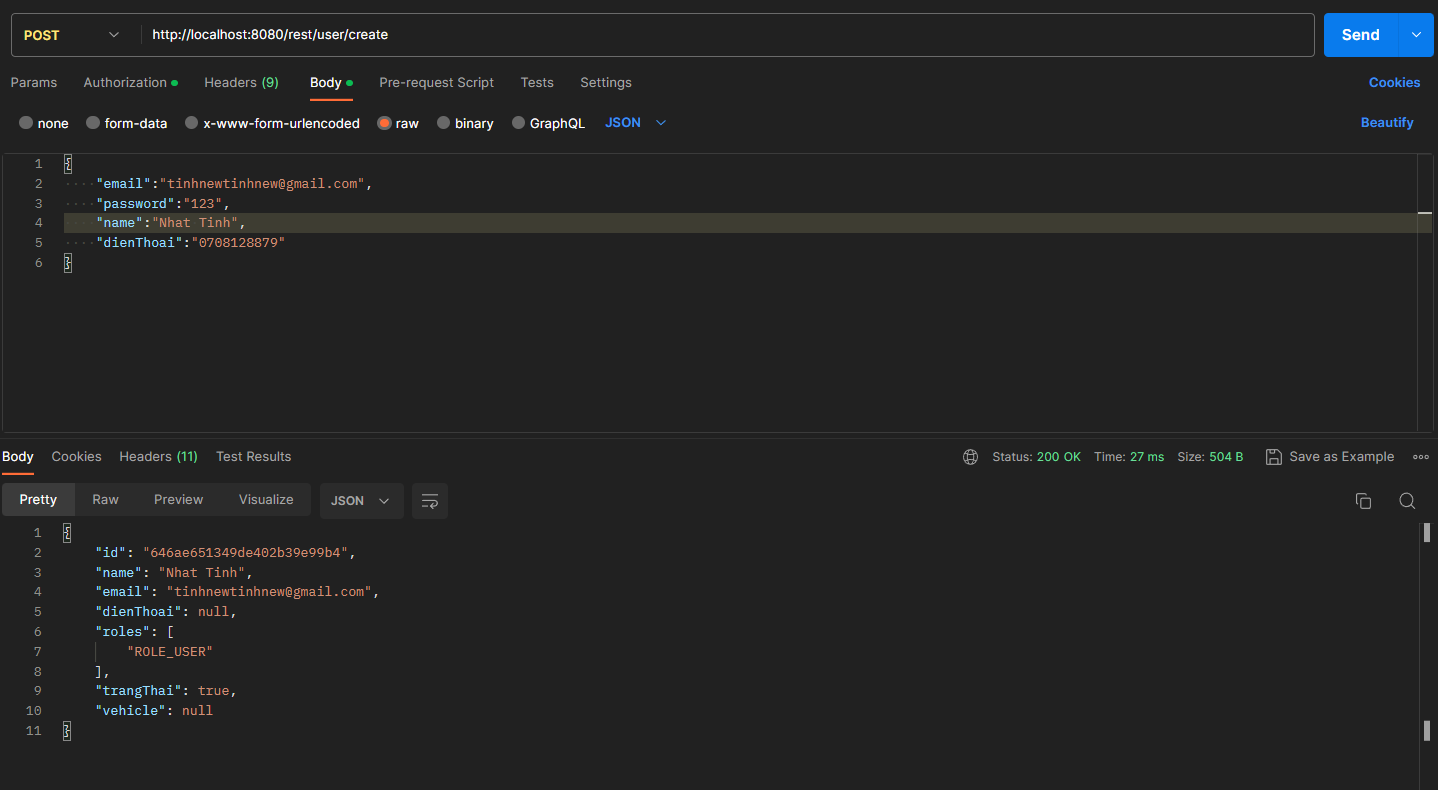
## List All User



## List All User Paging



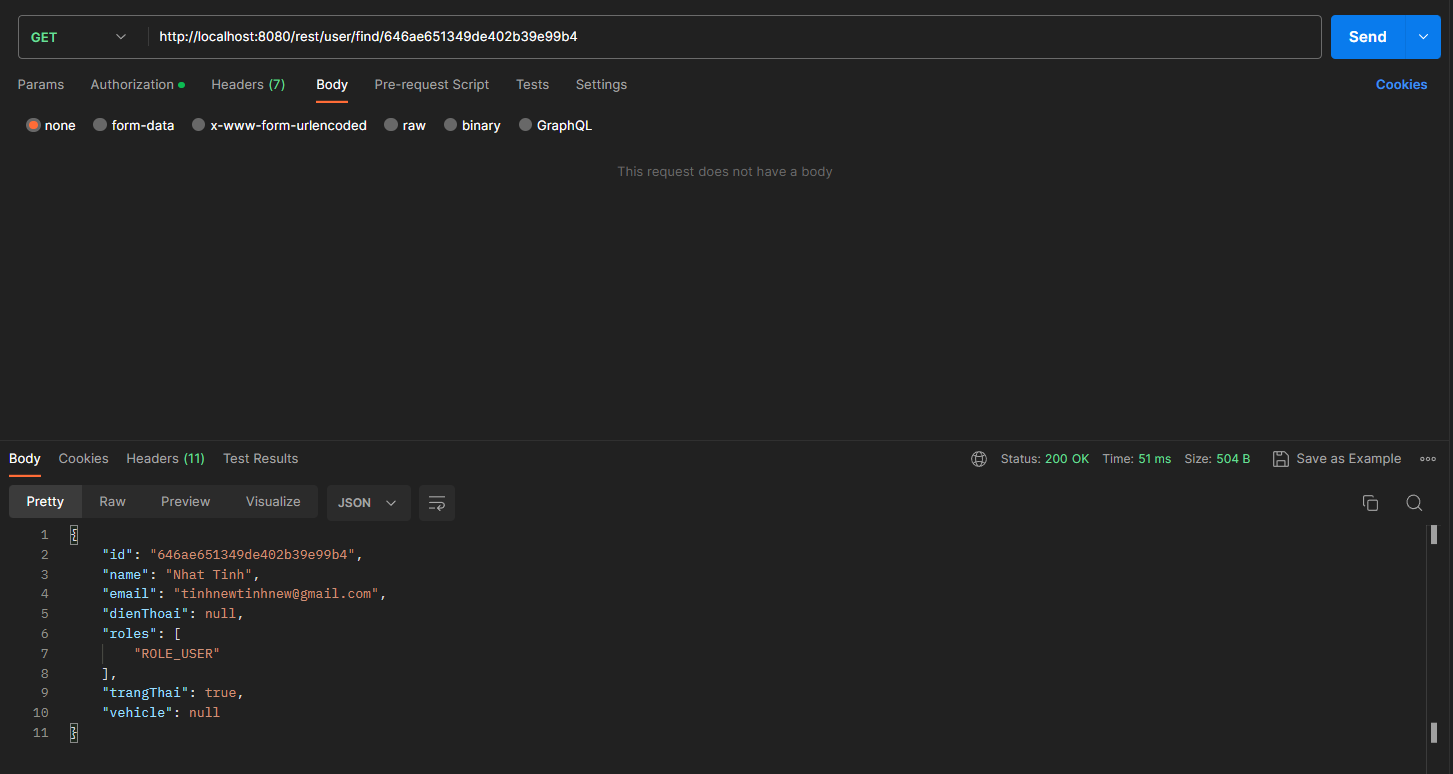
## Tạo mới User



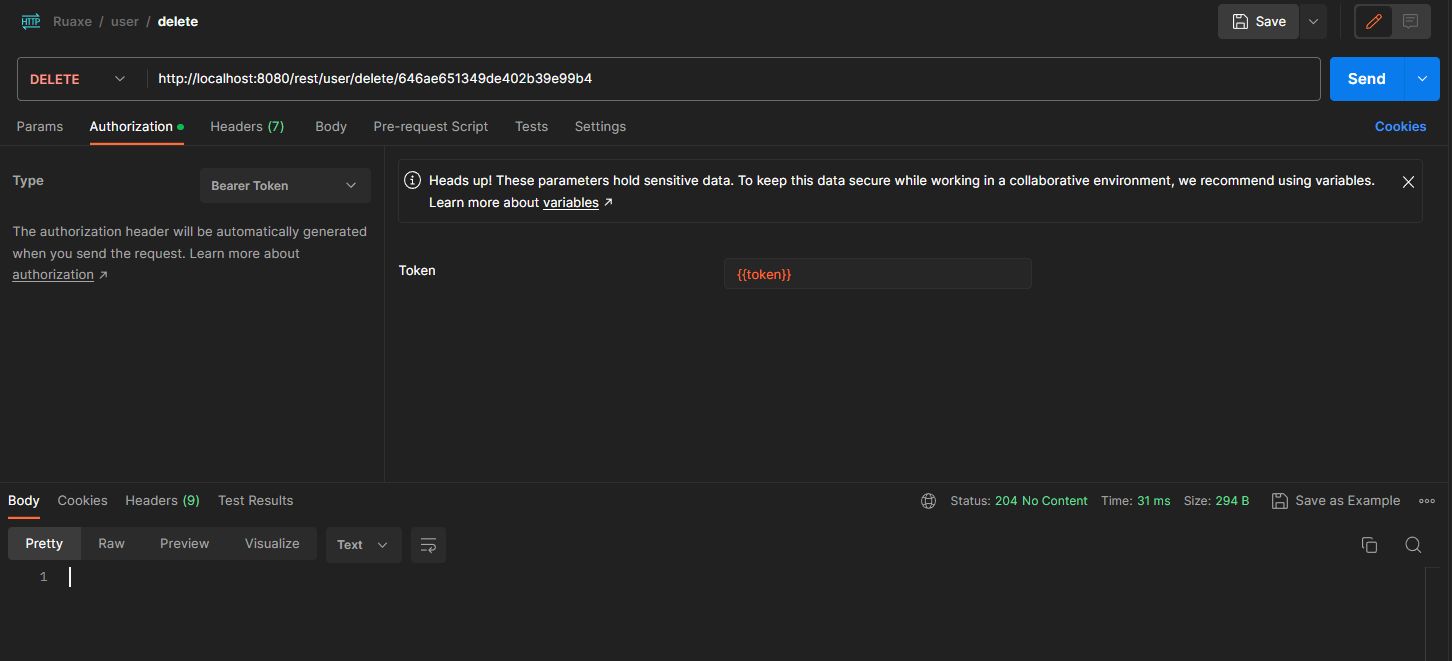
A screen shot of a computer screen

Description automatically generated with low confidence

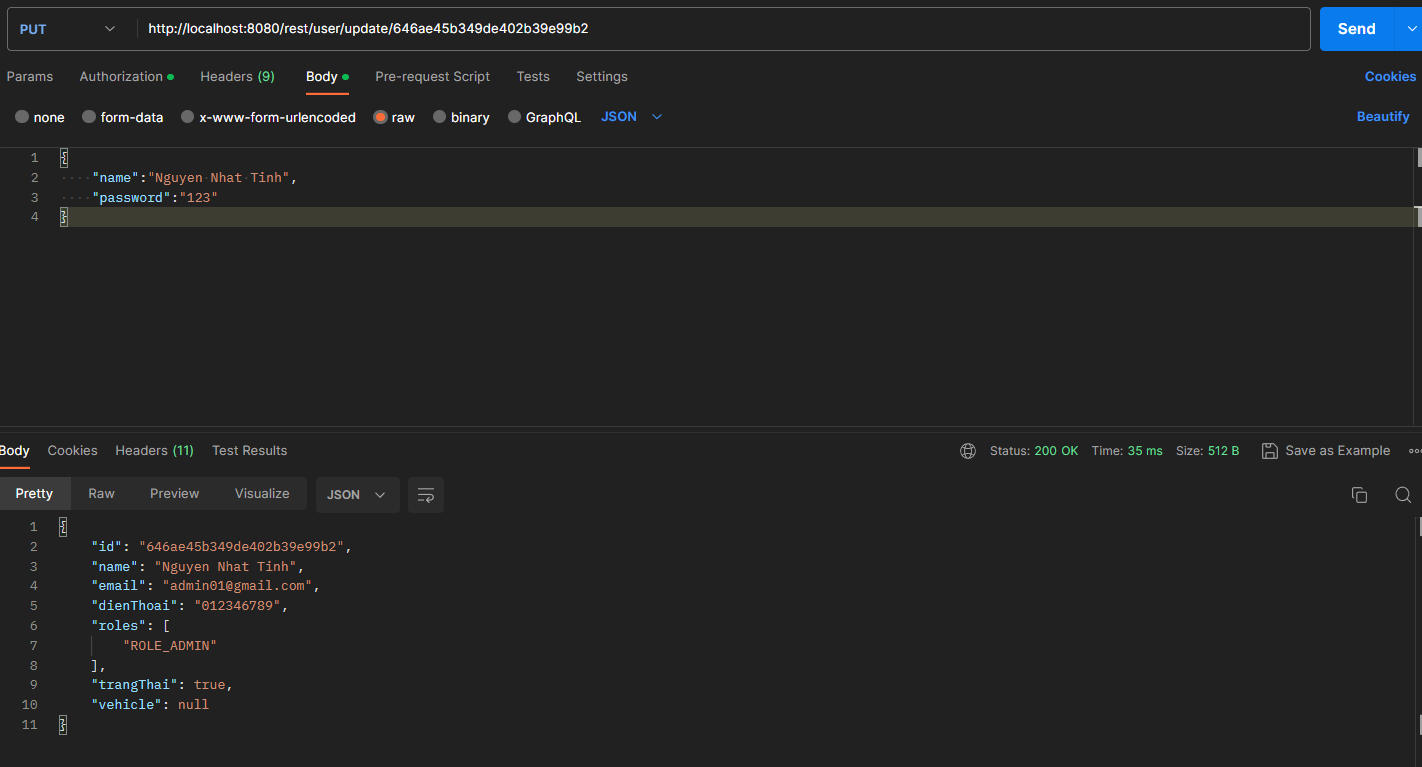
## Tìm kiếm User



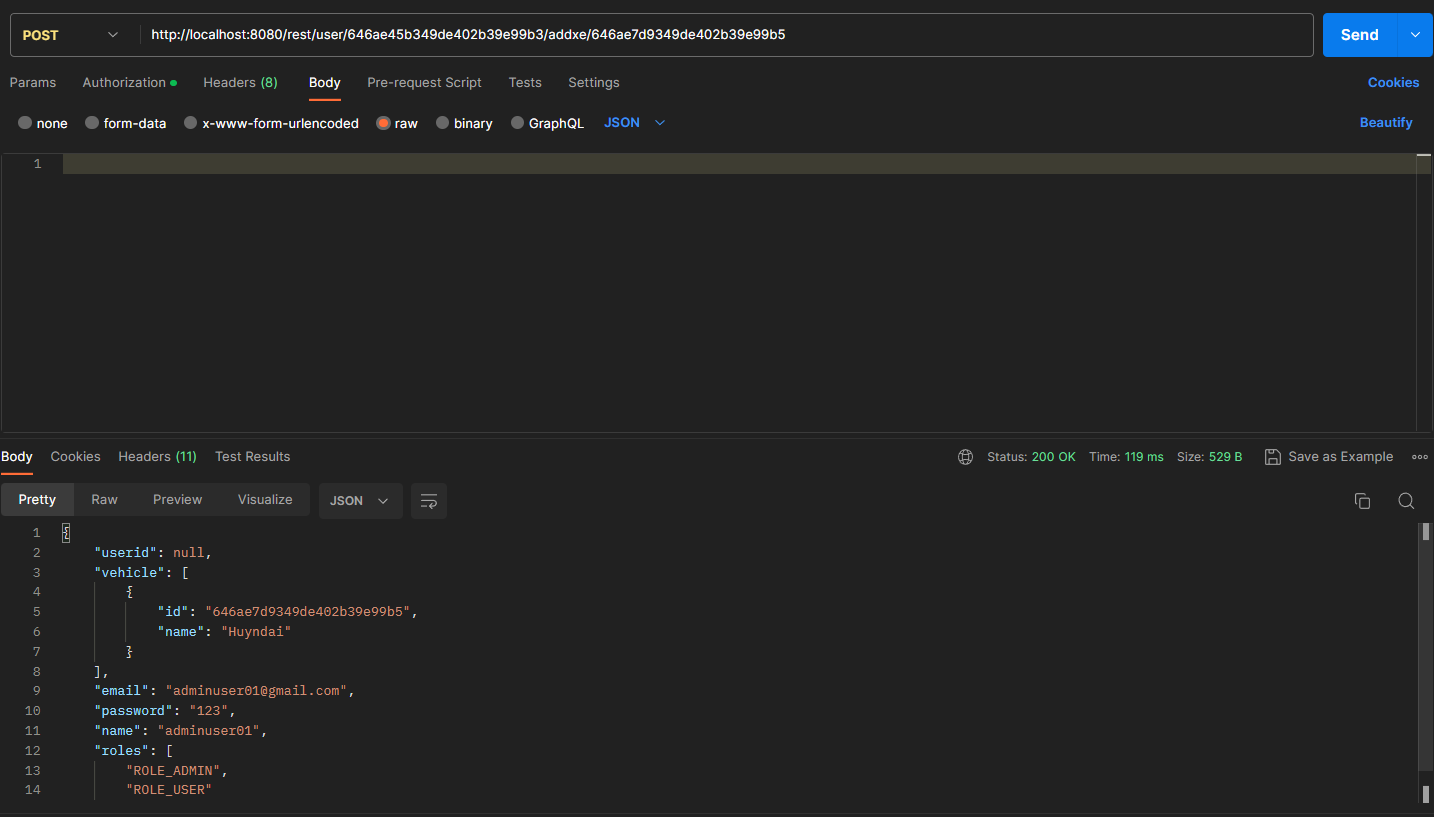
## Xóa người dùng



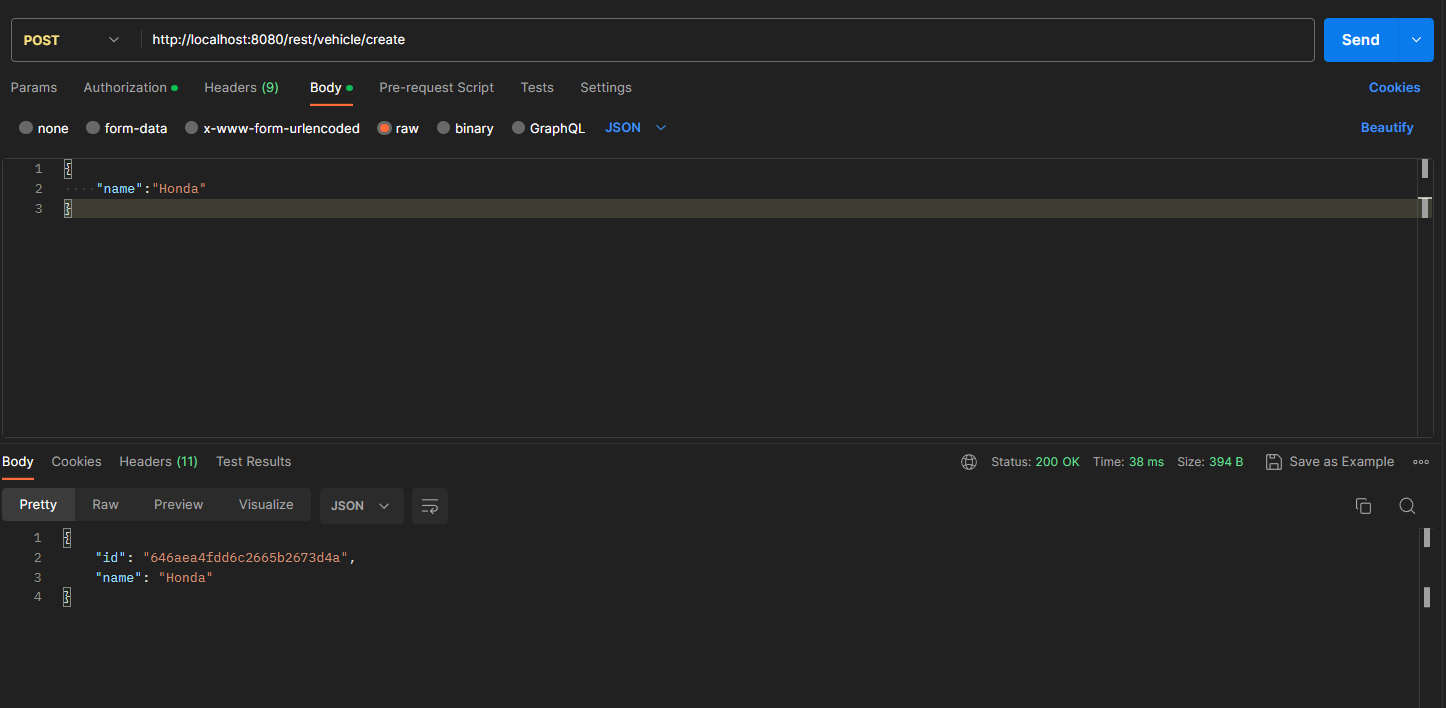
## Update người dùng



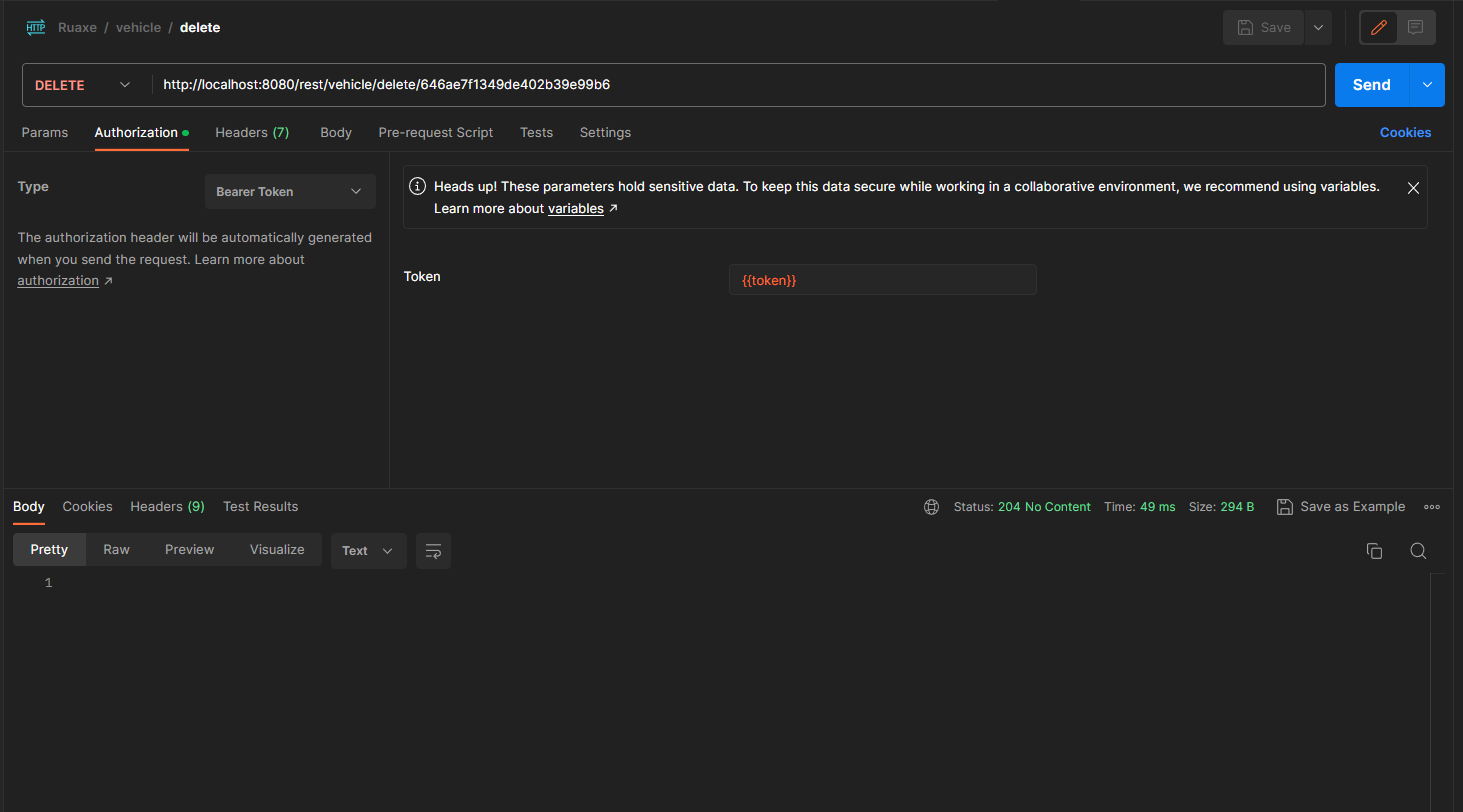
## Thêm 1 phương tiện cho người dùng



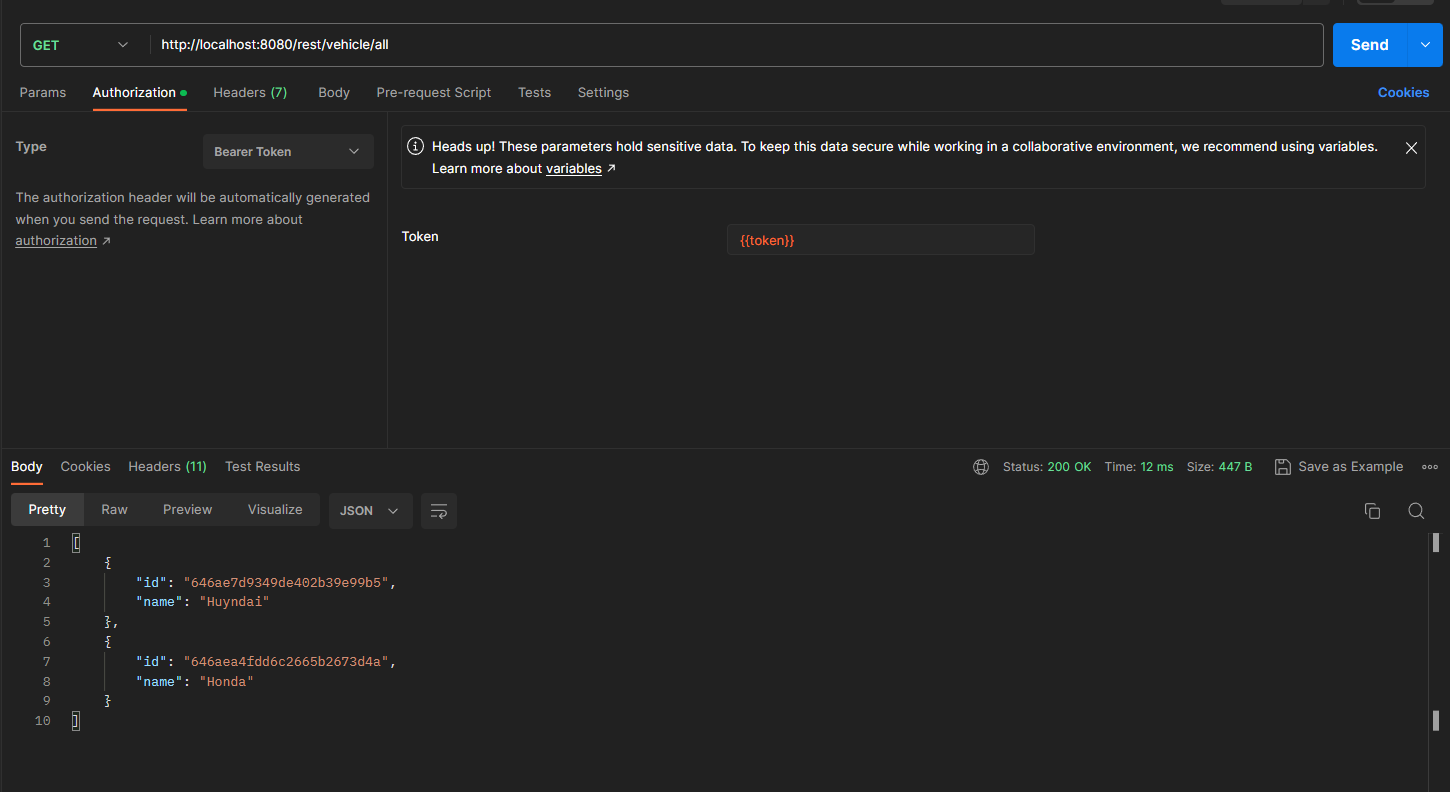
## Tạo một phương tiện



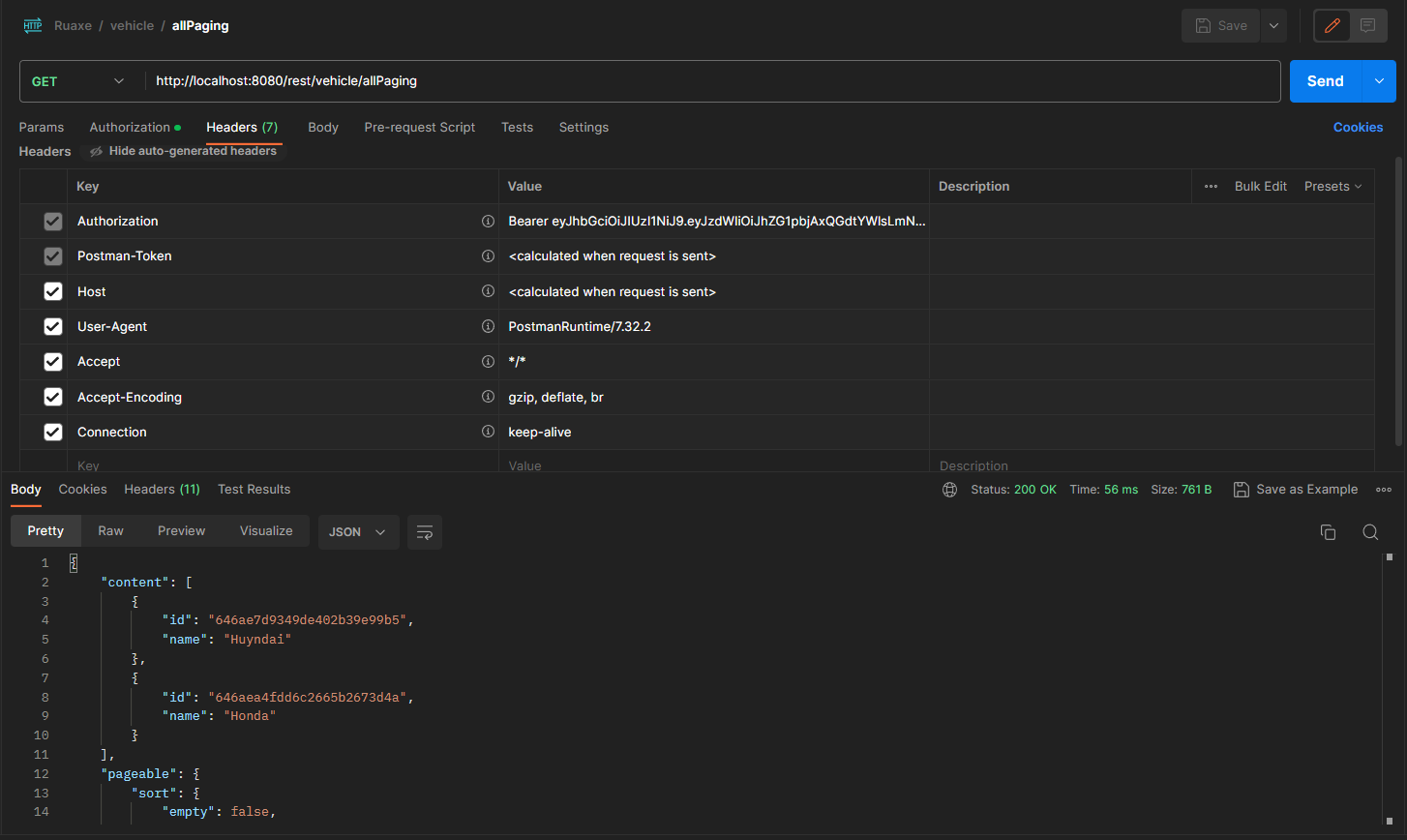
## Xóa phương tiện



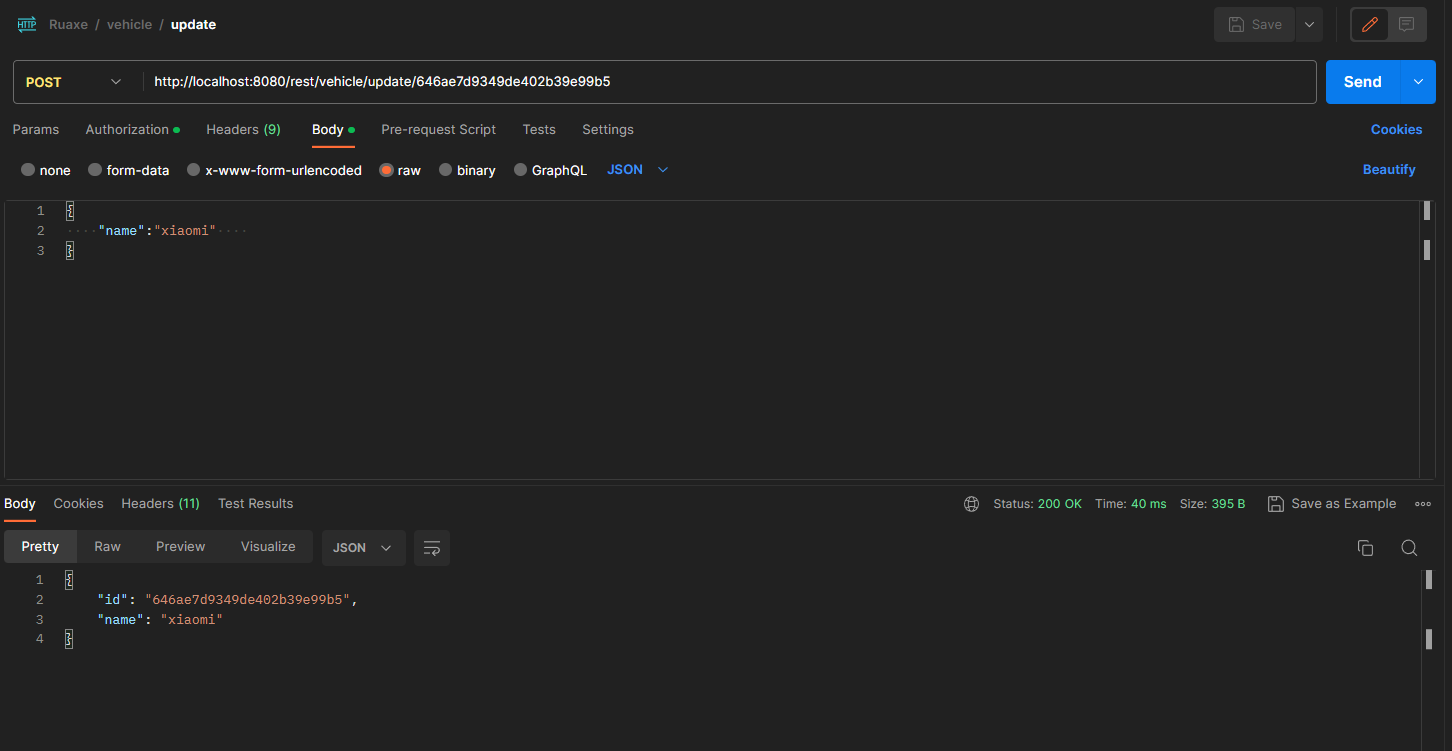
## Liệt kê danh sách phương tiện



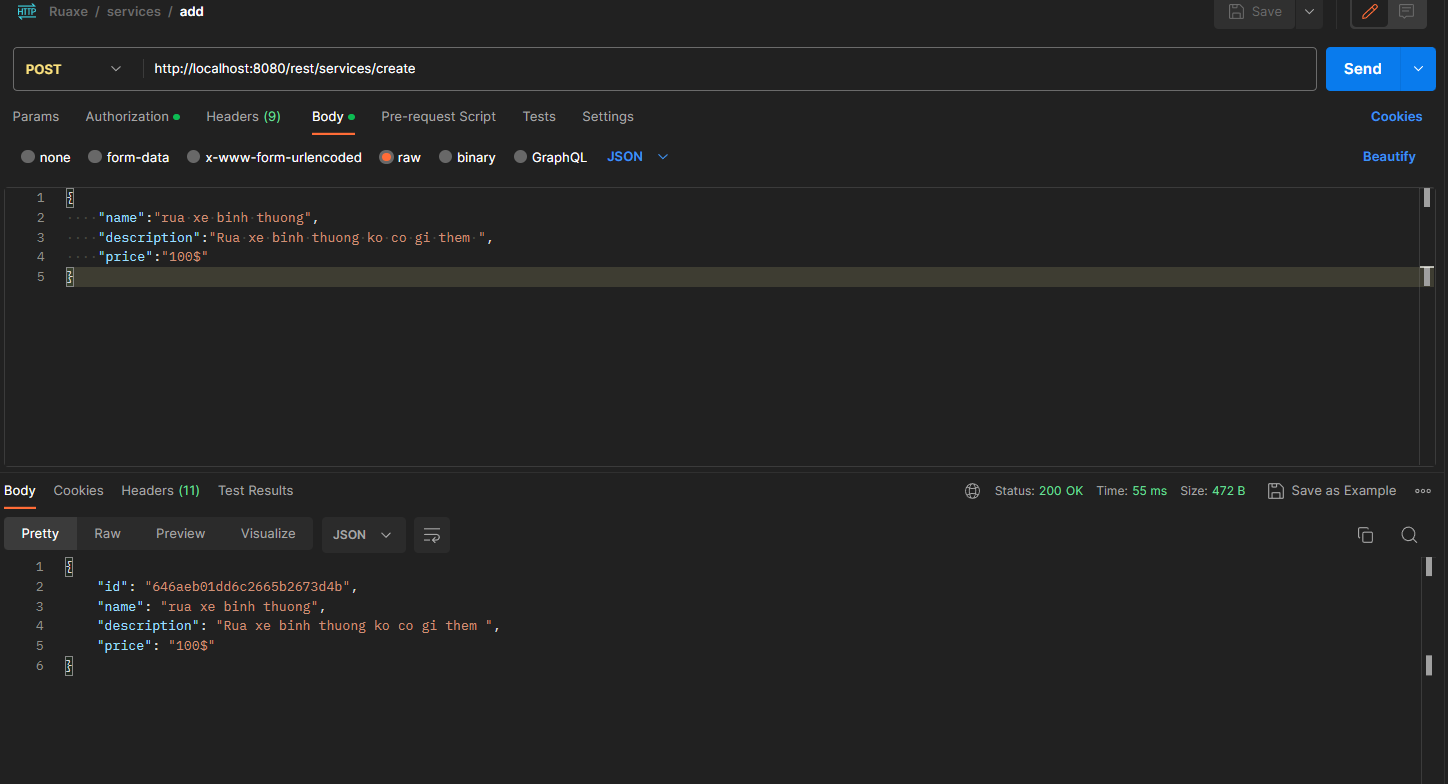
## Liệt kê danh sách phương tiện ( có phân trang)



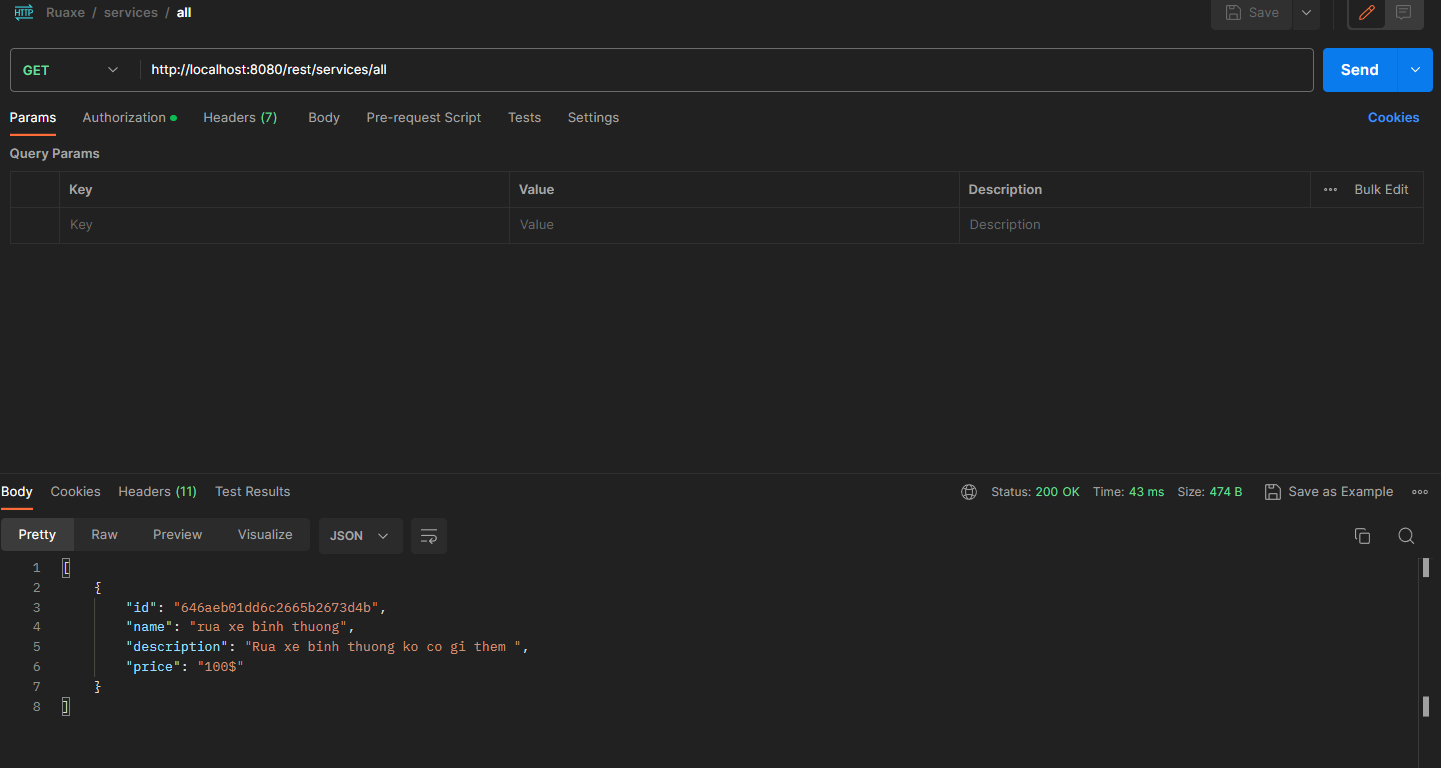
## Cập nhật phương tiện



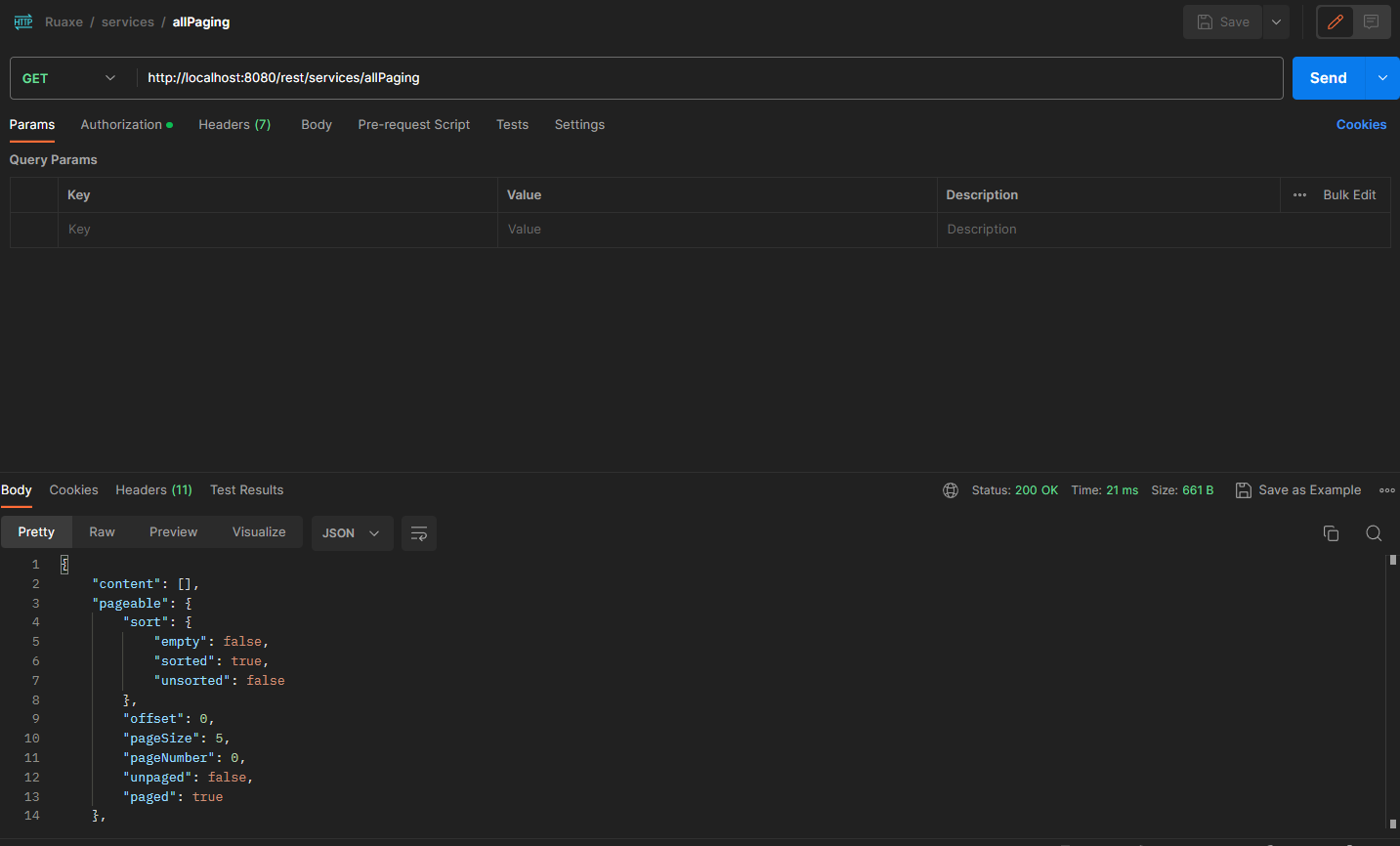
## Thêm dịch vụ



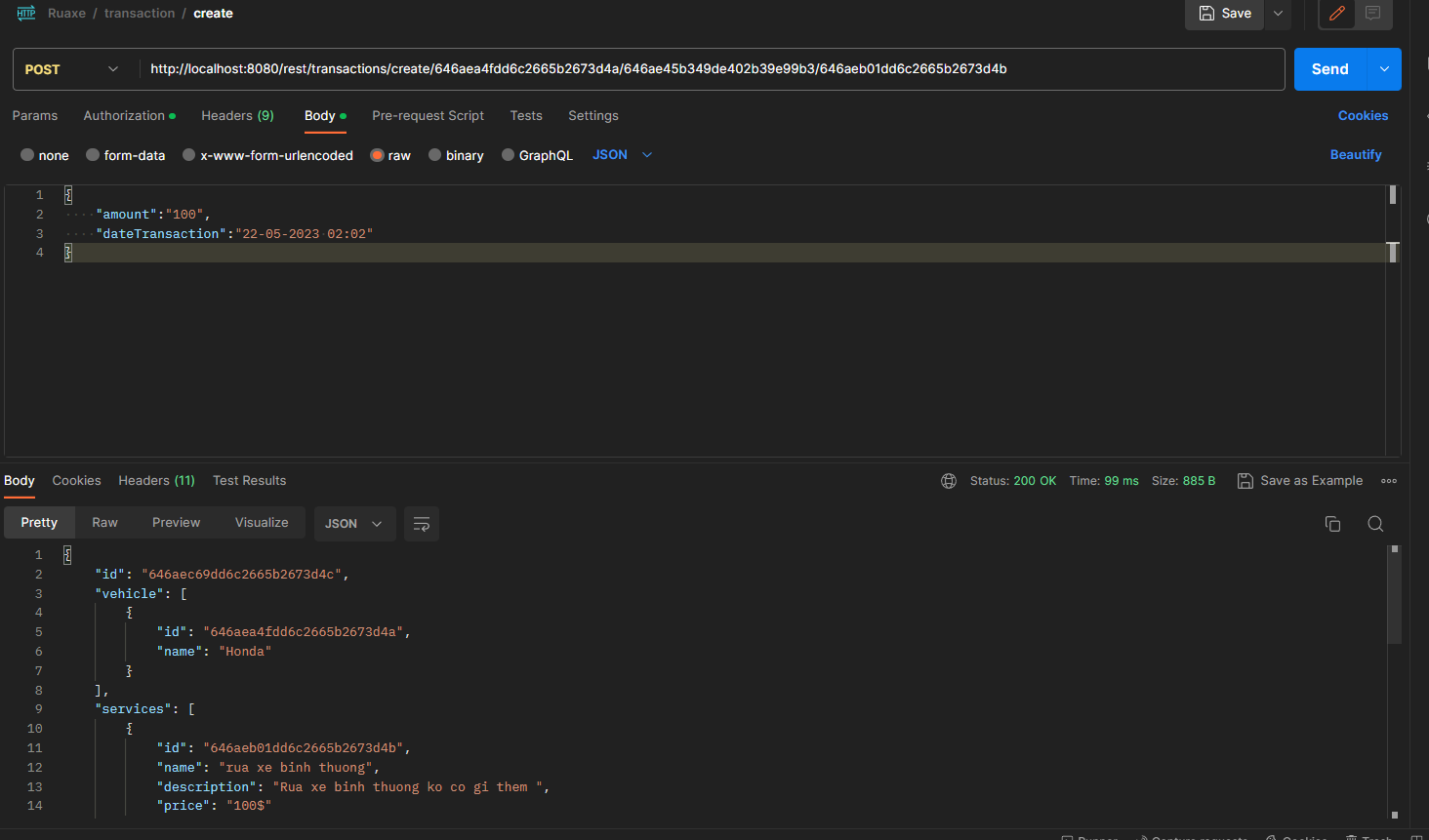
## Liệt kê dịch vụ



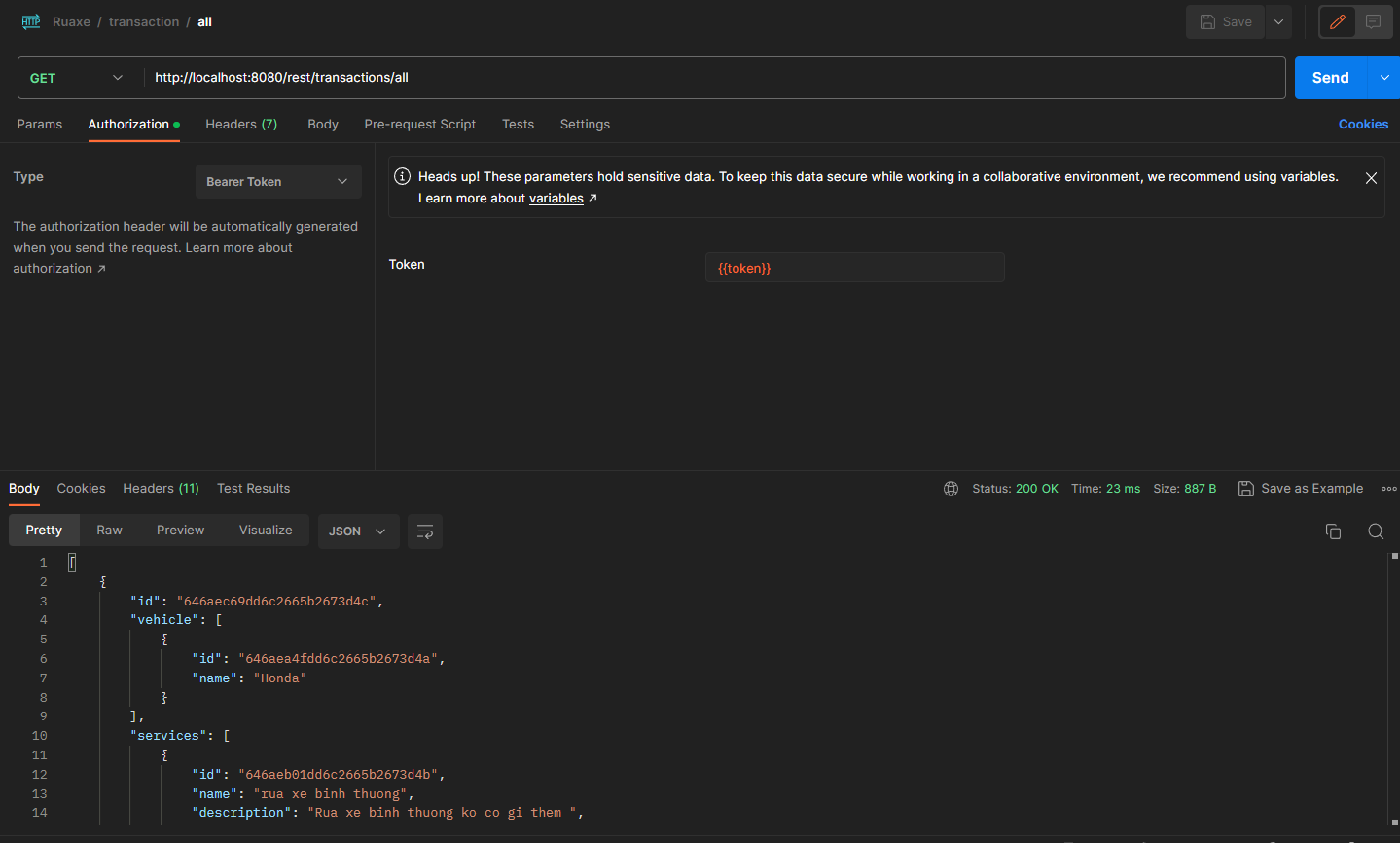
## Liệt kê dịch vụ (có phân trang)



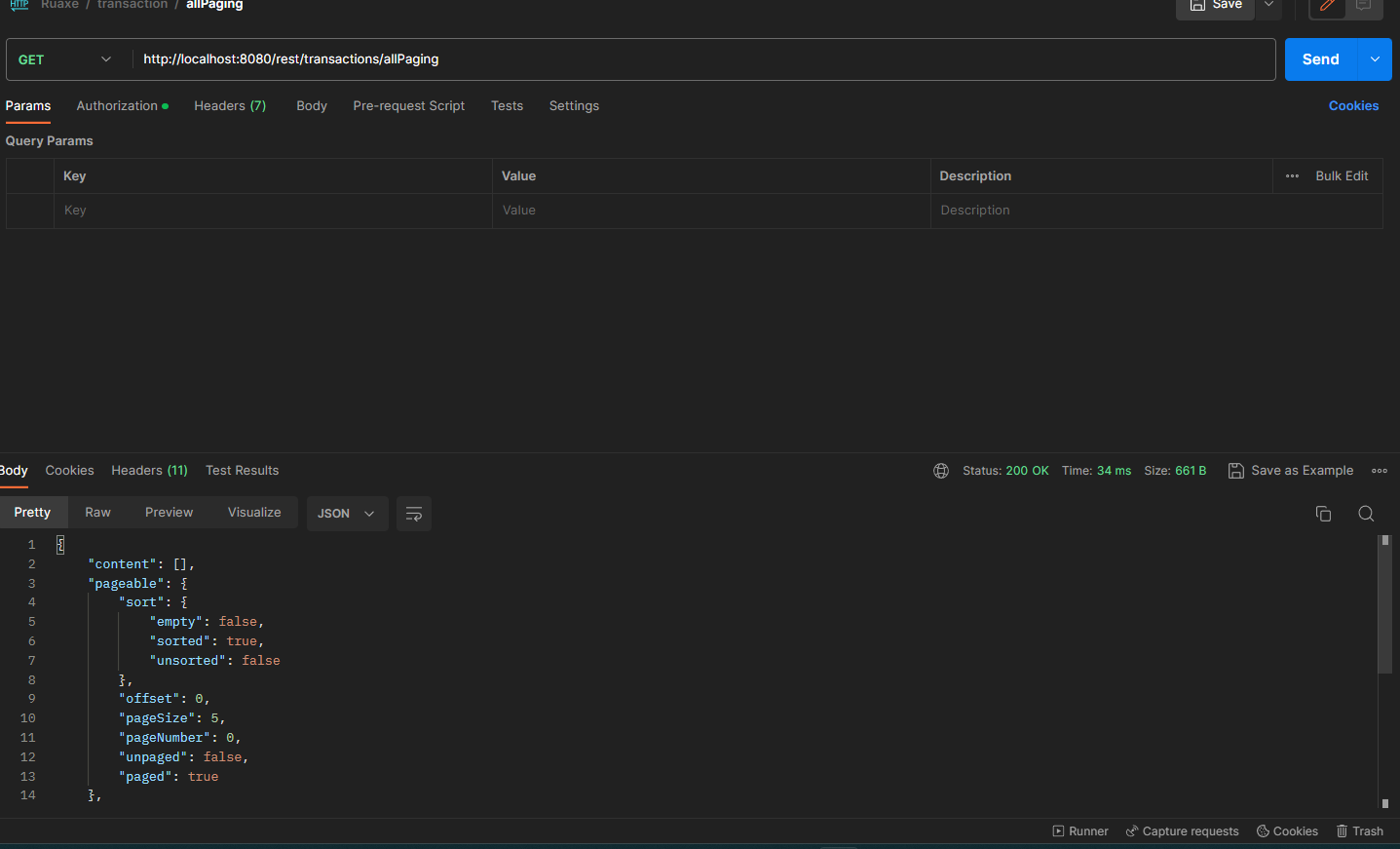
## Tạo mới giao dịch



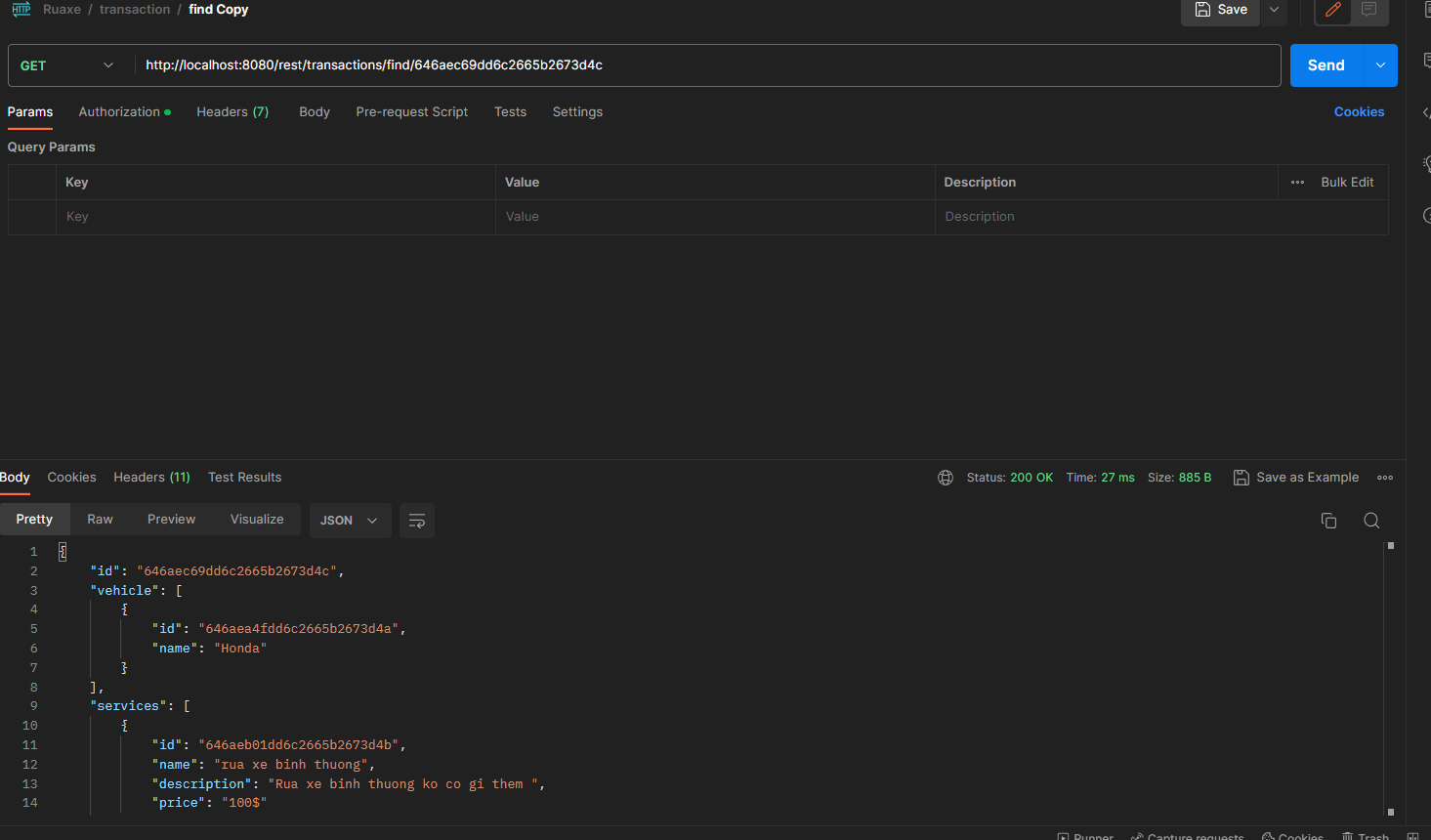
## Liệt kê danh sách các giao dịch



## Liệt kê danh sách các giao dịch (có phân trang)



## Tìm kiếm giao dịch



@RequestParam(name = "sort", defaultValue = "asc") String sort,  
 @RequestParam(name = "column", defaultValue = "id") String column) {  
 Page<Amenities> amenities = amenitiesService.filter(search, page, size, sort, column);  
 return ResponseEntity.*ok*(amenities);  
}

@Override  
public Page<Amenities> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
  
 if (search == null) {  
 return Page.*empty*(pageable);  
 }  
  
 return amenitiesRepository.findByIdContainingOrNameContainingAllIgnoreCase(search, search, pageable);  
}

**- Bảng khách sạn**

@PreAuthorize("hasAnyRole('ADMIN','USER')")  
@GetMapping("/filter")  
public ResponseEntity<Page<Hotel>> filter(  
 @RequestParam(value = "search", required = false, defaultValue = "") String search,  
 @RequestParam(value = "page", required = false, defaultValue = "0") int page,  
 @RequestParam(value = "size", required = false, defaultValue = "10") int size,  
 @RequestParam(value = "sort", required = false, defaultValue = "asc") String sort,  
 @RequestParam(value = "column", required = false, defaultValue = "id") String column) {  
 Page<Hotel> hotels = hotelService.filter(search, page, size, sort, column);  
 return new ResponseEntity<>(hotels, HttpStatus.*OK*);  
}

@Override  
public Page<Hotel> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
 return hotelRepository.findByIdContainingOrNameContainingAllIgnoreCase(search, search, pageable);  
}

**- Bảng thanh toán**

@PreAuthorize("hasAnyRole('ADMIN','USER')")  
@GetMapping("/filter")  
public Page<Payment> filter(@RequestParam(value = "search", required = false) String search,  
 @RequestParam(value = "page", defaultValue = "0") int page,  
 @RequestParam(value = "size", defaultValue = "10") int size,  
 @RequestParam(value = "sort", defaultValue = "asc") String sort,  
 @RequestParam(value = "column", defaultValue = "id") String column) {  
 return paymentService.filter(search, page, size, sort, column);  
}

@Override  
public Page<Payment> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
  
 if (search == null) {  
  
 return Page.*empty*(pageable);  
 }  
  
 return paymentRepository.findByIdContainingOrAmountContainingAllIgnoreCase(search, search, pageable);  
}

**- Bảng đặt chỗ**

@PreAuthorize("hasAnyRole('ADMIN','USER')")  
@GetMapping("/filter")  
public ResponseEntity<Page<ReservationDto>> filter(  
 @RequestParam(required = false) String search,  
 @RequestParam(defaultValue = "0") int page,  
 @RequestParam(defaultValue = "10") int size,  
 @RequestParam(defaultValue = "desc") String sort,  
 @RequestParam(defaultValue = "id") String column) {  
  
 Page<Reservation> reservations = reservationService.filter(search, page, size, sort, column);  
 List<ReservationDto> reservationDtos = reservations.stream()  
 .map(reservation -> modelMapper.map(reservation, ReservationDto.class))  
 .collect(Collectors.*toList*());  
  
 Page<ReservationDto> reservationDtosPage = new PageImpl<>(reservationDtos, reservations.getPageable(), reservations.getTotalElements());  
 return ResponseEntity.*ok*(reservationDtosPage);  
}

@Override  
public Page<Reservation> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
  
 if (search == null) {  
 return Page.*empty*(pageable);  
 }  
  
 return reservationRepository.findByIdContainingOrPaymentContainingAllIgnoreCase(search, search, pageable);  
}

**- Bảng phòng**

@PreAuthorize("hasAnyRole('ADMIN','USER')")  
@GetMapping("/filter")  
public ResponseEntity<Page<Room>> filter(  
 @RequestParam(required = false, defaultValue = "") String search,  
 @RequestParam(required = false, defaultValue = "0") int page,  
 @RequestParam(required = false, defaultValue = "10") int size,  
 @RequestParam(required = false, defaultValue = "ASC") String sort,  
 @RequestParam(required = false, defaultValue = "id") String column) {  
 Page<Room> roomPage = roomService.filter(search, page, size, sort, column);  
 return ResponseEntity.*ok*(roomPage);  
}

@Override  
public Page<Room> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
 return roomRepository.findByIdContainingOrNumberContainingAllIgnoreCase(search,search, pageable);  
}

**- Bảng user**

@PreAuthorize("hasAnyRole('ADMIN','USER')")  
@GetMapping("/filter")  
public ResponseEntity<Page<UserDto>> filter(@RequestParam(required = false) String search,  
 @RequestParam(defaultValue = "0") int page,  
 @RequestParam(defaultValue = "10") int size,  
 @RequestParam(defaultValue = "asc") String sort,  
 @RequestParam(defaultValue = "id") String column) {  
 Page<User> users = userService.filter(search, page, size, sort, column);  
 List<UserDto> userDtos = users.getContent().stream()  
 .map(user -> new UserDto(user.getId(),user.getName(), user.getEmail()))  
 .collect(Collectors.*toList*());  
 Page<UserDto> userDtoPage = new PageImpl<>(userDtos, users.getPageable(), users.getTotalElements());  
 return ResponseEntity.*ok*(userDtoPage);  
}

@Override  
public Page<User> filter(String search, int page, int size, String sort, String column) {  
 Pageable pageable = PageUtils.*createPageable*(page, size, sort, column);  
  
 if (search == null) {  
 return Page.*empty*(pageable);  
 }  
  
 return userRepository.findByIdContainingOrEmailContainingAllIgnoreCase(search, search, pageable);  
}