PHÁT TRIỂN BACKEND DỰ ÁN CARTRADEVN

MỤC LỤC

[PHẦN 1: PHÂN TÍCH NGHIỆP VỤ 2](#_Toc194357404)

[**1.** **Tìm kiếm / mua xe** 2](#_Toc194357405)

[**2.** **Chi tiết sản phẩm** 2](#_Toc194357406)

[**3.** **Bán xe** 3](#_Toc194357407)

[**4.** **Database:** 4](#_Toc194357408)

[PHẦN 2: CÔNG NGHỆ SỬ DỤNG 5](#_Toc194357409)

[PHẦN 3: CẤU TRÚC DỰ ÁN 6](#_Toc194357410)

[**1.** **Tổ chức** 6](#_Toc194357411)

[**2.** **Giải thích thành phần:** 9](#_Toc194357412)

# PHẦN 1: PHÂN TÍCH NGHIỆP VỤ

1. **Tìm kiếm / mua xe**

**Nghiệp vụ:**

* **Tìm kiếm sản phẩm**: Người dùng tìm kiếm xe dựa trên:
  + Khu vực (thành phố/tỉnh).
  + Hãng xe (BMW, Honda,…) và mẫu xe (i8, Civic,…).
  + Giá (lọc từ min đến max trong trang Inventory).
  + Loại nhiên liệu (Petrol, Electric,… - tùy chọn hiển thị chi tiết).
  + Trạng thái xe (mới/cũ).
* **Trang Buy**: Hiển thị danh sách xe “hot” hoặc mới đăng bán.

**Phân tích backend:**

* **API cần thiết**:
  + GET /vehicles: Lấy danh sách xe với tham số lọc (city, brand, model, minPrice, maxPrice, condition).
  + GET /vehicles/hot: Lấy danh sách xe “hot” (dựa trên lượt xem hoặc ngày đăng).
* **Xử lý dữ liệu**:
  + Bộ lọc động dựa trên tiêu chí người dùng.
  + Tối ưu truy vấn bằng cách sử dụng index cho các trường lọc phổ biến.

1. **Chi tiết sản phẩm**

**Nghiệp vụ:**

* Hiển thị thông tin xe: hãng, mẫu, giá, loại nhiên liệu, trạng thái,…
* **Liên hệ người bán**:
  + Thông tin người bán: tên, số điện thoại, email.
  + Tùy chọn: yêu cầu thêm ảnh, đặt lịch lái thử, hỏi thêm thông tin.
* **Hình thức thanh toán**:
  + Trả đủ (full payment).
  + Trả góp (installment) với tính toán tài chính.

**Phân tích backend:**

* **API cần thiết**:
  + GET /vehicles/{id}: Lấy thông tin chi tiết xe theo ID.
  + POST /contact-seller: Gửi yêu cầu liên hệ từ khách hàng đến người bán.
  + POST /calculate-finance: Tính toán tài chính (giá xe, thời gian trả góp, lãi suất).
* **Xử lý dữ liệu**:
  + Lưu thông tin chi tiết xe.
  + Gửi thông báo (email/SMS) cho người bán khi có liên hệ.
  + Tính toán trả góp bằng công thức đơn giản hoặc thư viện tài chính.

1. **Bán xe**

**Nghiệp vụ:**

* **Hướng dẫn bán xe**: Các bước chuẩn bị (thủ tục, giấy tờ).
* **Định giá xe**: Form nhập thông tin (hãng, mẫu, năm sản xuất, tình trạng) để ước tính giá.
* **Đăng bán xe**: Form chi tiết để đăng thông tin xe.
* **Phí hoa hồng**: Thông tin hoa hồng cho trung gian (nếu có).

**Phân tích backend:**

* **API cần thiết**:
  + POST /valuate-vehicle: Nhận thông tin xe và trả về giá định giá.
  + POST /vehicles: Tạo bài đăng bán xe (yêu cầu đăng nhập).
  + GET /selling-guide: Lấy nội dung hướng dẫn bán xe (static content).
* **Xử lý dữ liệu**:
  + Định giá dựa trên dữ liệu nội bộ hoặc API bên thứ 3.
  + Quản lý bài đăng với trạng thái (pending/approved/sold).
  + Tính phí hoa hồng nếu có trung gian.

1. **Database:**

**Bảng users (Người dùng)**

* `id` (PK)
* `username`
* `email`
* `password\_hash`
* `role` (buyer/seller/admin)

**Bảng vehicles (Xe)**

* `id` (PK)
* `user\_id` (FK -> users, người đăng bán)
* `brand` (hãng xe)
* `model` (mẫu xe)
* `price` (giá)
* `condition` (new/used)
* `fuel\_type` (Petrol/Electric,… - tùy chọn)
* `city` (thành phố/tỉnh)
* `description` (mô tả chi tiết)
* `status` (pending/approved/sold)
* `created\_at`

**Bảng vehicle\_images (Hình ảnh xe)**

* `id` (PK)
* `vehicle\_id` (FK -> vehicles)
* `image\_url`

**Bảng contacts (Liên hệ người bán)**

* `id` (PK)
* `vehicle\_id` (FK -> vehicles)
* `user\_id` (FK -> users, người liên hệ)
* `message` (nội dung)
* `request\_type` (more\_photos/test\_drive/more\_info - tùy chọn)
* `created\_at`

**Bảng posts (Bài viết Blog)**

* `id` (PK)
* `user\_id` (FK -> users, tác giả)
* `title`
* `content`
* `category` (car\_news/buying\_guides/...)
* `created\_at`

**Bảng comments (Bình luận)**

* `id` (PK)
* `post\_id` (FK -> posts)
* `user\_id` (FK -> users)
* `content`
* `created\_at`

# PHẦN 2: CÔNG NGHỆ SỬ DỤNG

* **Spring Boot** (Backend chính):
  + **Project**: Maven
  + **Language**: Java
  + **Version**: 3.4.4
  + **Group**: com.cartradevn
  + **Artifact**: cartradevn
  + **Name**: CarTradeVN
  + **Java**: 17
  + **Dependencies**:
    - Spring Web
    - Spring Data JPA
    - MySQL Driver
    - Lombok
    - Spring Security
    - Validation
    - Spring Boot Actuator
    - Spring Boot Starter Test
* **PHP**: Sử dụng cho API phụ hoặc frontend (nếu cần).
* **MySQL**
* **Link tham khảo**: <https://tinyurl.com/28cbqk5d>

# PHẦN 3: CẤU TRÚC DỰ ÁN

* + - 1. **Tổ chức**

**Project Structure**

**Source Path:** src/main/java/com/cartradevn/cartradevn

**Directory Overview**

* **administration**
* **controller**
* **dto**
* **entity**
* **repository**
* **service**
* **blog**
* **contact**

**Controller Files**

* UserController.java
* VehicleController.java
* ContactController.java
* PostController.java

**DTO Files**

* UserDTO.java
* VehicleDTO.java
* PostDTO.java
* CommentDTO.java

**Entity Files**

* User.java
* Vehicle.java
* Post.java
* Comment.java
* Contact.java

**Repository Files**

* UserRepository.java
* VehicleRepository.java
* PostRepository.java
* CommentRepository.java
* ContactRepository.java

**Main Application File**

Application.java

**Hình ảnh hình dung:**A screenshot of a computer

AI-generated content may be incorrect.

* + - 1. **Giải thích thành phần:**
* **Entity (Thực thể)**:
  + **Là gì?**: Entity là một lớp Java đại diện cho một bảng trong cơ sở dữ liệu. Mỗi thuộc tính trong lớp tương ứng với một cột trong bảng.
  + **Vai trò**: Được dùng để ánh xạ (mapping) dữ liệu từ database lên ứng dụng và ngược lại. Trong Spring Boot, nó sử dụng JPA (Java Persistence API) để tự động tạo bảng nếu cần.
  + **Ví dụ**: User.java ánh xạ bảng users với các cột id, username, email, v.v.
* **DTO (Data Transfer Object - Đối tượng truyền dữ liệu)**:
  + **Là gì?**: DTO là một lớp đơn giản dùng để truyền dữ liệu giữa các lớp hoặc giữa backend và frontend. Nó không liên quan trực tiếp đến cơ sở dữ liệu.
  + **Vai trò**: Giúp kiểm soát dữ liệu được gửi/nhận, tránh lộ thông tin nhạy cảm (như passwordHash). DTO thường nhẹ hơn Entity và chỉ chứa các trường cần thiết.
  + **Ví dụ**: UserDTO.java chỉ chứa id, username, email, không có passwordHash.
* **Repository (Kho lưu trữ)**:
  + **Là gì?**: Repository là một giao diện (interface) trong Spring Data JPA, dùng để thực hiện các thao tác với cơ sở dữ liệu như thêm, xóa, sửa, tìm kiếm.
  + **Vai trò**: Cung cấp các phương thức sẵn có (như findAll(), save()) mà không cần viết SQL thủ công. Có thể thêm các phương thức tùy chỉnh nếu cần.
  + **Ví dụ**: UserRepository.java cung cấp cách lấy tất cả người dùng hoặc lưu một người dùng mới.
* **Controller (Bộ điều khiển)**:
  + **Là gì?**: Controller là lớp xử lý các yêu cầu HTTP từ người dùng (như GET, POST) và trả về phản hồi.
  + **Vai trò**: Đóng vai trò trung gian giữa frontend và backend, nhận dữ liệu từ DTO, gọi Repository để xử lý, và trả kết quả. Nó định nghĩa các endpoint API (ví dụ: /api/users).
  + **Ví dụ**: UserController.java xử lý yêu cầu lấy danh sách người dùng hoặc tạo người dùng mới.