System Analysis & Design
Semester II - 2026
Assignment 1

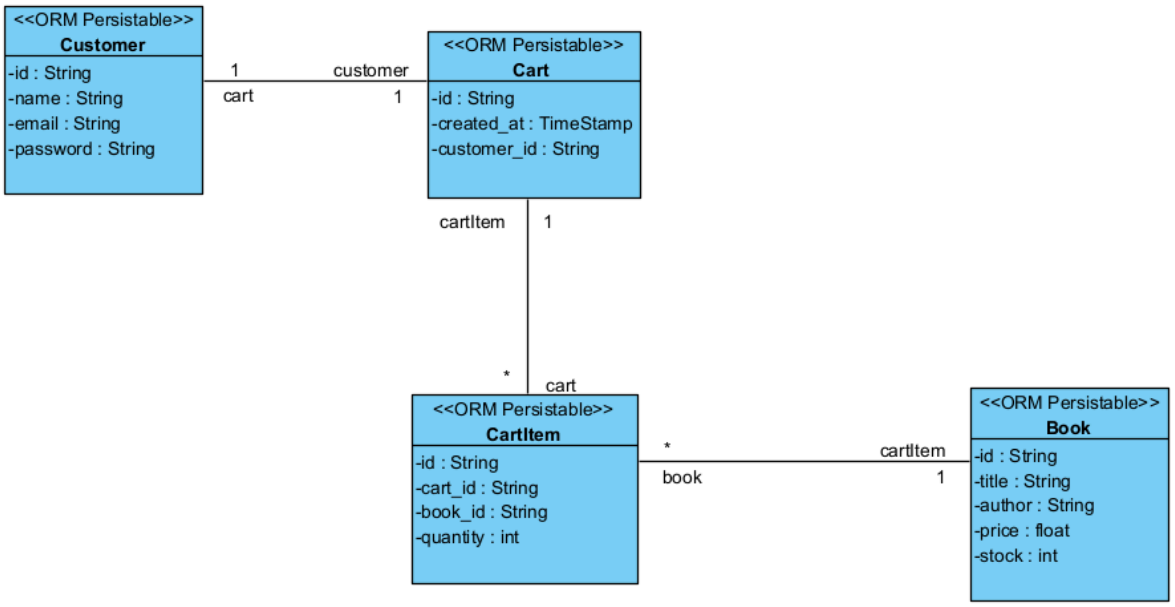
Họ và tên: Lê Đăng Ninh
Mã sinh viên: B22DCCN572

Mục lục

Nội dung

1. UML Design

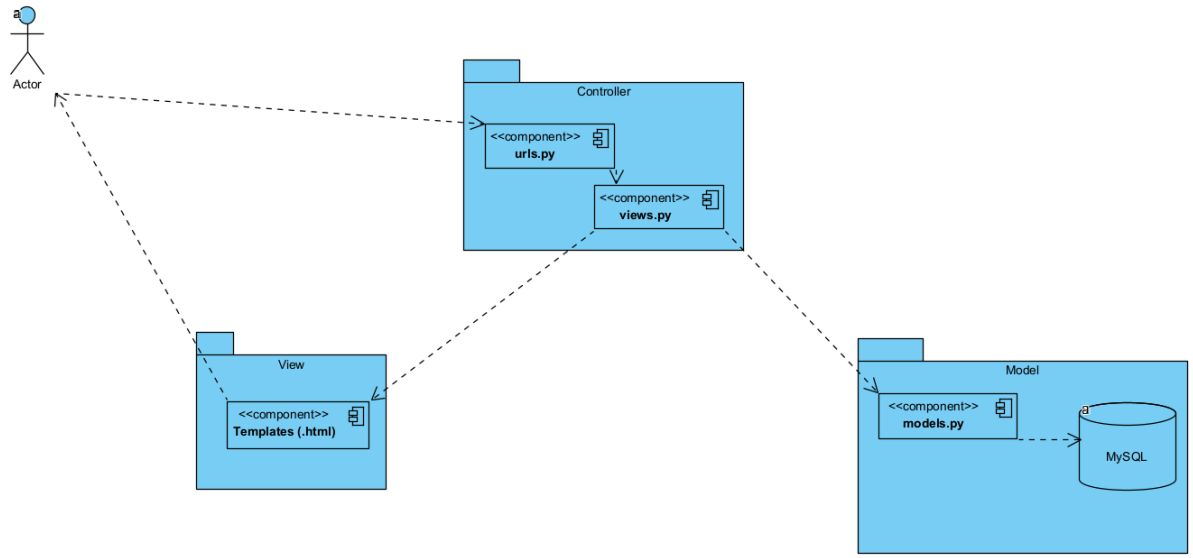
1.1. Class Diagram



- **Các Thực thể (Entities):**
  - **Customer:** Đóng vai trò là người dùng cuối. Các thuộc tính Email và Password phục vụ cho chức năng xác thực (Authentication).
  - **Book:** Đại diện cho sản phẩm. Thuộc tính Stock (tồn kho) là biến số quan trọng quyết định logic nghiệp vụ (có cho phép thêm vào giỏ hay không).
  - **Cart:** Đại diện cho phiên mua sắm. Việc sử dụng varchar(255) cho khóa chính (Id) thay vì Auto Increment Integer cho thấy thiết kế hướng tới khả năng mở rộng (Scalability), có thể sử dụng UUID để tránh xung đột dữ liệu khi phân tán hệ thống.
  - **CartItem:** Đây là thực thể liên kết (Associative Entity) nhằm giải quyết mối quan hệ Nhiều-Nhiều (N-N) giữa Cart và Book.
- **Mối quan hệ (Relationships):**
  - **Customer - Cart (1-1):** Một khách hàng sở hữu một giỏ hàng. Điều này giúp đơn giản hóa logic quản lý phiên: mỗi người dùng luôn có một trạng thái giỏ hàng duy nhất tại một thời điểm.
  - **Cart - CartItem (1-N):** Một giỏ hàng chứa nhiều đầu sách khác nhau (Aggregation).

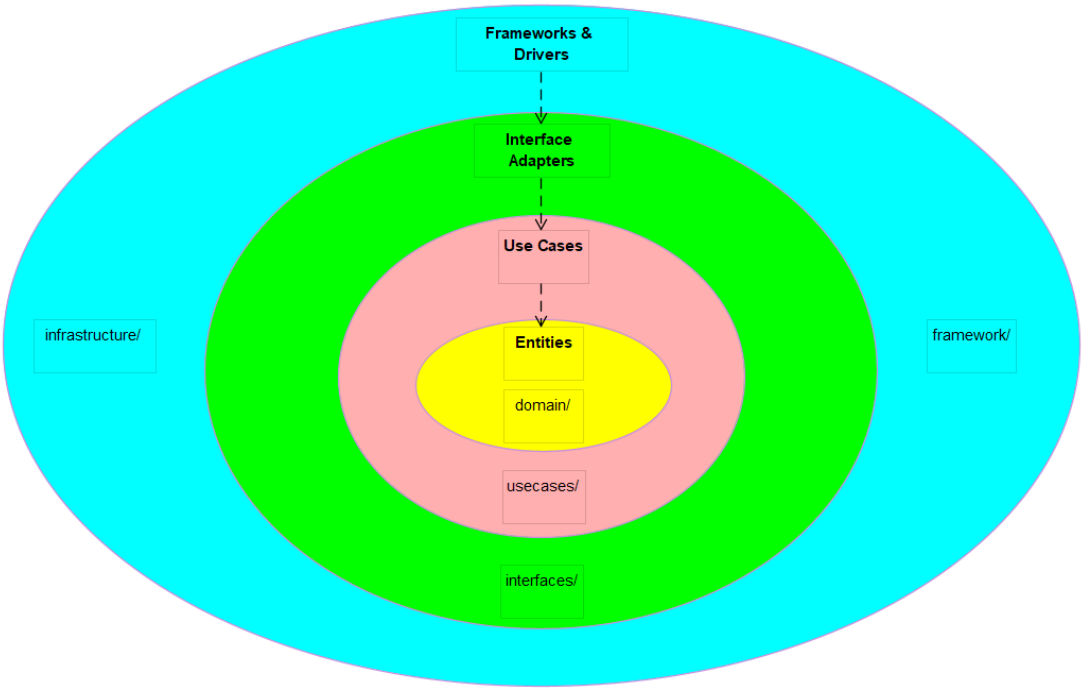
    - **Book - CartItem (1-N):** Một cuốn sách cụ thể có thể xuất hiện trong nhiều giỏ hàng của nhiều người khác nhau.
- **Nhận xét thiết kế:** Thiết kế này đã được chuẩn hóa (Normalized) để tránh dư thừa dữ liệu. Việc tách CartItem giúp hệ thống linh hoạt, có thể mở rộng thêm các thuộc tính như discount (giảm giá) cho từng món hàng sau này mà không ảnh hưởng đến bảng Cart hay Book.

1.2. MVC Layer Diagram



- **Các thành phần chính:**
    - **Actor:** Người dùng gửi yêu cầu (Request) từ trình duyệt.
    - **Controller (Điều phối):** Trong Django, vai trò này được đảm nhiệm bởi urls.py (định tuyến) và views.py (xử lý logic).
        - urls.py: Nhận URL từ người dùng và chuyển đến hàm xử lý tương ứng.
        - views.py: Gọi xuống Model để lấy dữ liệu, xử lý nghiệp vụ, sau đó đẩy dữ liệu ra Template.
    - **Model (Dữ liệu):** File models.py đóng vai trò ORM (Object Relational Mapping), ánh xạ trực tiếp các lớp Python sang bảng MySQL.
    - **View (Giao diện):** Thư mục Templates (.html) nhận dữ liệu từ Controller để hiển thị cho người dùng.
- **Luồng hoạt động (Data Flow):**
    - Mũi tên nét đứt thể hiện sự phụ thuộc trực tiếp. Controller phụ thuộc vào Model để lấy dữ liệu và phụ thuộc vào View để trả về giao diện.
    - Tất cả các thành phần nằm trong cùng một khối (Deployment Unit).

1.3. Clean Architecture Diagram



1.3.1. Cấu trúc Các lớp Đồng tâm (The Concentric Layers)

Biểu đồ thể hiện hệ thống được chia thành 4 lớp bảo vệ lẫn nhau, đi từ trong ra ngoài:

1.3.1.1. Lớp Entities (Màu vàng - Core Domain)

- **Vị trí:** Trung tâm của biểu đồ, tương ứng với thư mục domain/ trong dự án.
- **Chức năng:** Chứa các "Doanh nghiệp vụ cốt lõi" (Enterprise Business Rules). Đây là các class mô tả đối tượng thuần túy nhất của hệ thống như Book, Cart.
- **Đặc điểm:** Lớp này độc lập hoàn toàn, **không phụ thuộc vào bất kỳ lớp nào bên ngoài**. Nó không biết Django là gì, không biết Database là MySQL hay SQLite. Mã nguồn ở đây là Python thuần (Pure Python).

1.3.1.2. Lớp Use Cases (Màu hồng - Application Business Rules)

- **Vị trí:** Bao quanh Entities, tương ứng với thư mục usecases/.
- **Chức năng:** Chứa các logic nghiệp vụ cụ thể của ứng dụng. Ví dụ: quy trình "Thêm vào giỏ hàng" (AddToCartUseCase).
- **Đặc điểm:** Lớp này điều phối luồng dữ liệu đến và đi từ các Entities. Nó chỉ đạo các Entities thực hiện nhiệm vụ nhưng không quan tâm dữ liệu được hiển thị trên Web hay Mobile, hay được lưu xuống SQL hay file text.

1.3.1.3. Lớp Interface Adapters (Màu xanh lá - Adapters)

- **Vị trí:** Tương ứng với thư mục interfaces/.
- **Chức năng:** Đóng vai trò là "Người phiên dịch" (Translator). Nó chuyển đổi dữ liệu từ định dạng thuận tiện cho Use Cases và Entities sang định dạng thuận tiện cho các tác nhân bên ngoài (như Web, Database) và ngược lại.
- **Trong dự án:** Đây là nơi chứa các Views (Controllers) nhận Request từ Django và gọi vào Use Case, hoặc các Repositories (bản chất là Adapter) thực thi giao tiếp với Database.

1.3.1.4. Lớp Frameworks & Drivers (Màu xanh dương - Infrastructure)

- **Vị trí:** Lớp ngoài cùng, tương ứng với thư mục infrastructure/ và framework/. +1
- **Chức năng:** Chứa các công cụ, thư viện, framework cụ thể. Đây là nơi duy nhất trong hệ thống mà mã nguồn phụ thuộc vào Django Framework hay thư viện MySQL.
- **Trong dự án:** File models.py (Django ORM) nằm ở đây. Đây là nơi các chi tiết kỹ thuật "bẩn" (dirty details) được thực hiện để kết nối với thế giới thực.

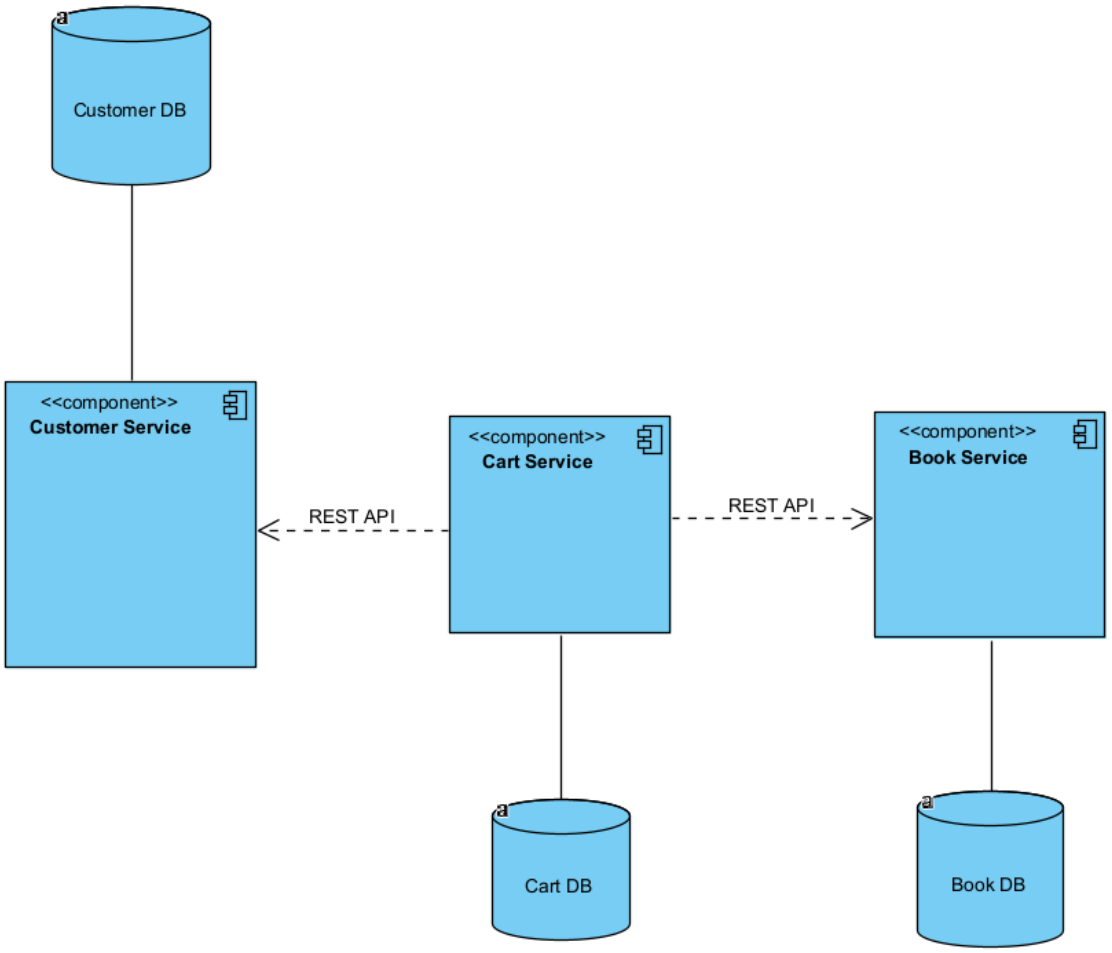
1.3.2. Quy tắc Phụ thuộc (The Dependency Rule)

- **Mũi tên nét đứt:** Các mũi tên trong hình đều hướng từ ngoài vào trong. Đây là quy tắc quan trọng nhất: **Dependency Rule**.
- **Ý nghĩa:** Các lớp bên trong **tuyệt đối không được biết** bất cứ điều gì về các lớp bên ngoài.
    - domain không được import django.
    - usecases không được import infrastructure.
- **Tác dụng:** Điều này đảm bảo rằng khi bạn thay đổi giao diện (UI) hoặc thay đổi Database (từ MySQL sang PostgreSQL), các logic nghiệp vụ cốt lõi (Use Cases và Entities) không hề bị ảnh hưởng hay phải sửa đổi code.

1.3.3. Đánh giá Thiết kế trong Dự án

- **Tính độc lập của Framework:** Hệ thống không bị khóa chặt vào Django. Nếu cần chuyển sang Flask hay FastAPI, chỉ cần viết lại lớp ngoài cùng (Frameworks & Drivers), giữ nguyên 3 lớp bên trong.
- **Khả năng kiểm thử (Testability):** Vì Business Logic (Use Cases) không dính đến UI hay Database, ta có thể viết Unit Test cực nhanh mà không cần khởi động Web Server hay kết nối SQL Server.
- **Sự đánh đổi:** Việc áp dụng kiến trúc này tạo ra nhiều file và thư mục hơn so với bản Monolithic (Version A), đòi hỏi viết nhiều code "boilerplate" (code chuyển đổi) hơn, nhưng đổi lại là sự bền vững và dễ bảo trì cho các hệ thống lớn.

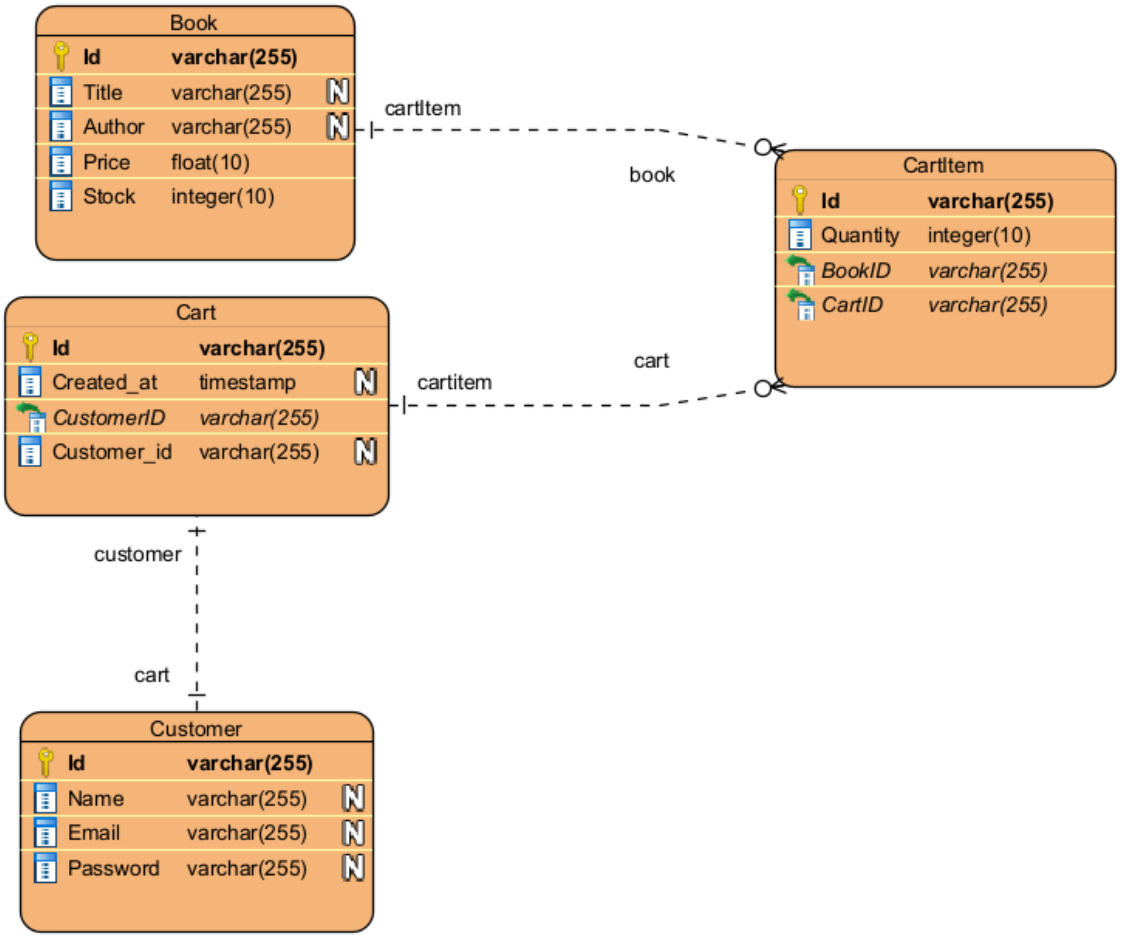
1.4. Microservices Architecture Diagram



- **Nguyên lý "Database per Service" (Mỗi dịch vụ một CSDL):**
  - Đây là điểm quan trọng nhất của biểu đồ.
  - **Customer Service** sở hữu Customer DB.
  - **Cart Service** sở hữu Cart DB.
  - **Book Service** sở hữu Book DB.
  - Ý nghĩa: Không dịch vụ nào được phép truy cập trực tiếp vào DB của dịch vụ khác. Điều này đảm bảo tính đóng gói (Encapsulation) dữ liệu tuyệt đối.
- **Cơ chế giao tiếp (Communication):**
  - Mũi tên nét đứt có nhãn **REST API** thể hiện giao tiếp qua mạng (HTTP Request).
  - **Cart Service** đóng vai trò là "Orchestrator" (Người điều phối) trong ngữ cảnh này:
    - Nó gọi sang Customer Service để xác thực người mua.
    - Nó gọi sang Book Service để lấy thông tin giá và tồn kho.
- **Phân tích luồng nghiệp vụ:**

- Khi người dùng thêm sách vào giỏ: Cart Service không thể tự biết giá sách (vì nó không có bảng Book trong Cart DB). Nó bắt buộc phải gửi một request GET /api/books/{id} sang Book Service.
- Điều này giải thích tại sao trong code thực tế chúng ta cần thư viện requests.

2. Database



2.1. Chi tiết các Thực thể (Entities)

2.1.1. Bảng Customer (Khách hàng)

Bảng này lưu trữ thông tin định danh của người dùng trong hệ thống.

- **Id** (varchar(255) - PK): Khóa chính định danh duy nhất cho mỗi khách hàng. Việc sử dụng kiểu chuỗi (varchar) thay vì số nguyên (integer) gợi ý việc sử dụng UUID, phù hợp cho khả năng mở rộng sau này.
- **Name** (varchar(255)): Tên hiển thị của khách hàng.

- **Email** (varchar(255)): Địa chỉ thư điện tử, thường được sử dụng làm tên đăng nhập.
- **Password** (varchar(255)): Lưu trữ mật khẩu (trong thực tế sẽ lưu mã hash bảo mật).

2.1.2. Bảng Book (Sách)

Lưu trữ thông tin về sản phẩm sách trong kho.

- **Id** (varchar(255) - PK): Khóa chính định danh cuốn sách.
- **Title** (varchar(255)): Tên cuốn sách.
- **Author** (varchar(255)): Tên tác giả.
- **Price** (float(10)): Giá bán của sách.
- **Stock** (integer(10)): Số lượng tồn kho. Đây là trường quan trọng để kiểm tra tính khả dụng của sản phẩm trước khi thêm vào giỏ.

2.1.3. Bảng Cart (Giỏ hàng)

Đại diện cho "phiên mua sắm" của khách hàng.

- **Id** (varchar(255) - PK): Khóa chính của giỏ hàng.
- **Created_at** (timestamp): Thời điểm giỏ hàng được tạo.
- **Customer_id** (varchar(255) - FK): Khóa ngoại tham chiếu đến bảng Customer. Trường này thiết lập mối quan hệ giữa khách hàng và giỏ hàng của họ.

2.1.4. Bảng CartItem (Chi tiết giỏ hàng)

Đây là bảng trung gian (Association Table) giải quyết mối quan hệ nhiều-nhiều giữa Giỏ hàng và Sách.

- **Id** (varchar(255) - PK): Khóa chính của dòng chi tiết.
- **CartID** (varchar(255) - FK): Khóa ngoại tham chiếu đến bảng Cart.
- **BookID** (varchar(255) - FK): Khóa ngoại tham chiếu đến bảng Book.
- **Quantity** (integer(10)): Số lượng của cuốn sách cụ thể đó trong giỏ hàng này.

2.2. Phân tích Mối quan hệ (Relationships)

2.2.1. Mối quan hệ Customer - Cart (1-1 hoặc 1-n)

- **Mô tả:** Sơ đồ thể hiện liên kết giữa Customer và Cart thông qua trường Customer_id.
- **Bản chất:** Trong ngữ cảnh bài toán này, mỗi khách hàng tại một thời điểm thường chỉ sở hữu một giỏ hàng đang hoạt động (Active Cart). Do đó, đây là quan hệ **1-1** (One-to-One) về mặt logic nghiệp vụ hiện tại.

2.2.2. Mối quan hệ Cart - Book (n-n)

- **Mô tả:** Một giỏ hàng có thể chứa nhiều loại sách khác nhau. Ngược lại, một cuốn sách (ví dụ: "Lập trình Python") có thể nằm trong giỏ hàng của nhiều người khác nhau.
- **Giải pháp:** Mối quan hệ Nhiều-Nhiều (Many-to-Many) này không thể thể hiện trực tiếp trong cơ sở dữ liệu quan hệ, do đó bảng **CartItem** được sinh ra để phá vỡ quan hệ này thành hai quan hệ 1-n:
    - **Cart - CartItem:** Quan hệ **1-n** (Một giỏ hàng có nhiều dòng chi tiết).
    - **Book - CartItem:** Quan hệ **1-n** (Một cuốn sách có thể xuất hiện trong nhiều dòng chi tiết của các giỏ khác nhau).

2.3. Đánh giá Thiết kế

- **Tính linh hoạt:** Việc tách biệt Cart (chứa thông tin chung) và CartItem (chứa thông tin chi tiết) giúp hệ thống dễ dàng mở rộng. Ví dụ: Nếu sau này cần tính năng "Lưu lại mua sau", chỉ cần thay đổi trạng thái của Cart hoặc CartItem.
- **Kiểu dữ liệu:** Sử dụng varchar cho ID là một lựa chọn thiết kế hiện đại, hỗ trợ tốt cho việc chuyển đổi sang kiến trúc Microservices (nơi các ID cần tính duy nhất toàn cục mà không phụ thuộc vào auto-increment của một DB cụ thể).
- **Tính toàn vẹn:** Các khóa ngoại (Customer_id, CartID, BookID) đảm bảo rằng không thể tạo một mục giỏ hàng cho cuốn sách không tồn tại hoặc cho một giỏ hàng chưa được khởi tạo.

3. Implementation

3.1. Version A - Monolithic Django

```
C:\ninh_proj1>python manage.py startapp accounts

C:\ninh_proj1>python manage.py startapp books

C:\ninh_proj1>python manage.py startapp cart

C:\ninh_proj1>dir
 Volume in drive C is Acer
 Volume Serial Number is BAE0-ACB1

 Directory of C:\ninh_proj1

01/20/2026  01:59 AM    <DIR>          .
01/20/2026  01:59 AM    <DIR>          accounts
01/20/2026  01:59 AM    <DIR>          books
01/20/2026  01:59 AM    <DIR>          cart
01/13/2026  09:41 AM           131,072 db.sqlite3
01/13/2026  09:34 AM               688 manage.py
01/13/2026  09:35 AM    <DIR>          ninh_proj1
               2 File(s)        131,760 bytes
               5 Dir(s)  43,583,311,872 bytes free
```

```
C:\ninh_proj1>pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-2.2.7-cp311-cp311-win_amd64.whl.metadata (4.8 kB)
Downloading mysqlclient-2.2.7-cp311-cp311-win_amd64.whl (207 kB)
Installing collected packages: mysqlclient
Successfully installed mysqlclient-2.2.7

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\ninh_proj1>python manage.py makemigrations
C:\Users\Admin\AppData\Local\Programs\Python\Python311\Lib\site-packages\django\core\management\commands\makemigrations.
py:161: RuntimeWarning: Got an error checking a consistent migration history performed for database connection 'default'
: (1045, "Access denied for user 'root'@'localhost' (using password: NO)")
  warnings.warn(
Migrations for 'books':
  books\migrations\0001_initial.py
    + Create model Book
Migrations for 'accounts':
  accounts\migrations\0001_initial.py
    + Create model Customer
Migrations for 'cart':
  cart\migrations\0001_initial.py
    + Create model Cart
    + Create model CartItem
```

```
C:\ninh_proj1>python manage.py migrate
Operations to perform:
  Apply all migrations: accounts, admin, auth, books, cart, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying accounts.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying books.0001_initial... OK
  Applying cart.0001_initial... OK
  Applying sessions.0001_initial... OK
```

```
C:\ninh_proj1>python manage.py createsuperuser
Username: admin
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

```
C:\ninh_proj1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2026 - 02:13:36
Django version 5.2.10, using settings 'ninh_proj1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server inste
ad.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[20/Jan/2026 02:13:41] "GET / HTTP/1.1" 200 12068
Not Found: /favicon.ico
[20/Jan/2026 02:13:42] "GET /favicon.ico HTTP/1.1" 404 2212
[20/Jan/2026 02:14:00] "GET /admin HTTP/1.1" 301 0
[20/Jan/2026 02:14:00] "GET /admin/ HTTP/1.1" 302 0
[20/Jan/2026 02:14:00] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 4173
[20/Jan/2026 02:14:01] "GET /static/admin/css/dark_mode.css HTTP/1.1" 200 2808
[20/Jan/2026 02:14:01] "GET /static/admin/css/nav_sidebar.css HTTP/1.1" 200 2810
[20/Jan/2026 02:14:01] "GET /static/admin/css/base.css HTTP/1.1" 200 22120
[20/Jan/2026 02:14:01] "GET /static/admin/css/login.css HTTP/1.1" 200 951
[20/Jan/2026 02:14:01] "GET /static/admin/js/theme.js HTTP/1.1" 200 1653
[20/Jan/2026 02:14:01] "GET /static/admin/css/responsive.css HTTP/1.1" 200 16565
[20/Jan/2026 02:14:01] "GET /static/admin/js/nav_sidebar.js HTTP/1.1" 200 3063
[20/Jan/2026 02:14:09] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[20/Jan/2026 02:14:09] "GET /admin/ HTTP/1.1" 200 9247
```

3.1.1. Cài đặt môi trường và Khởi tạo

- **Khởi tạo Project:** Dự án ninh_proj1 được tạo với 3 ứng dụng con (apps): accounts, books, cart.
- **Cài đặt thư viện:** Quá trình cài đặt mysqlclient để kết nối MySQL thành công sau khi tải wheel package.

3.1.2. Các vấn đề gặp phải và Cách giải quyết

Trong quá trình phát triển (Implementation), các lỗi sau đã được ghi nhận và xử lý:

- **Lỗi 1: Thiếu thư viện kết nối Database**
  - Hiện tượng: Báo lỗi ImproperlyConfigured: Error loading MySQLdb module.
  - Giải quyết: Chạy lệnh pip install mysqlclient.
- **Lỗi 2: Quyền truy cập Database**
  - Hiện tượng: Lỗi (1045, "Access denied for user 'root'@'localhost' (using password: NO)") khi chạy migration.
  - Giải quyết: Cấu hình lại settings.py với mật khẩu chính xác và đảm bảo user root có quyền truy cập.
- **Lỗi 3: Chưa tạo Database**
  - Hiện tượng: Lỗi (1049, "Unknown database 'db_monolith'") (xuất hiện trong log quá trình debug).
  - Giải quyết: Tạo database trong MySQL Workbench trước khi migrate.

3.1.3. Kết quả thực hiện

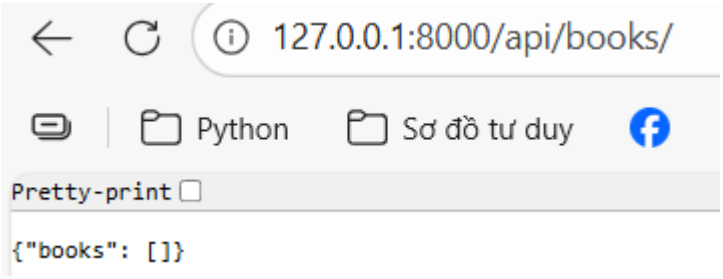
- **Migration:** Sau khi sửa lỗi, lệnh python manage.py migrate đã chạy thành công, tạo đầy đủ các bảng cho books, accounts, cart.
- **Giao diện Admin:** Trang quản trị Django hoạt động, hiển thị đầy đủ các models Users, Books, Carts, Cart items.
- **API Testing:**
  - API lấy danh sách sách (GET /books/) hoạt động, trả về JSON rỗng ban đầu {"books": []}.
  - API thêm vào giỏ (POST /cart/add/) ban đầu báo lỗi 404 do sách chưa tồn tại.
  - Sau khi thêm dữ liệu, API hoạt động thành công: {message: 'Da them vao gio hang thanh cong!', cart_id: 1}.

3.2. Version B - Clean Architecture Django

```
C:\>mkdir ninh_proj1_CleanArch

C:\>cd ninh_proj1_CleanArch

C:\ninh_proj1_CleanArch>django-admin startproject clean_arch .
```

```
C:\ninh_proj1_CleanArch>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

C:\ninh_proj1_CleanArch>python manage.py makemigrations infrastructure
Migrations for 'infrastructure':
  infrastructure\migrations\0001_initial.py
    + Create model BookORM
    + Create model CartORM
    + Create model CartItemORM
```

```
C:\ninh_proj1_CleanArch>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, infrastructure, sessions
Running migrations:
  Applying infrastructure.0001_initial... OK

C:\ninh_proj1_CleanArch>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2026 - 02:53:51
Django version 5.2.10, using settings 'clean_arch.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server inste
ad.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[20/Jan/2026 02:53:56] "GET /api/books/ HTTP/1.1" 200 13
```



3.2.1. Cấu trúc thư mục

Dự án được tái cấu trúc hoàn toàn để tuân thủ Clean Architecture, tách biệt rõ ràng các lớp :

- **domain:** Chứa Entities (Business Objects) thuần túy, không phụ thuộc Framework.
- **usecases:** Chứa logic nghiệp vụ (Managers), ví dụ AddToCartUseCase trong file managers.py.
- **infrastructure:** Chứa việc hiện thực hóa kết nối Database (Repositories, Models ORM).
- **interfaces:** Chứa các Adapters để giao tiếp với bên ngoài (Views/Controllers).

3.2.2. Quá trình tích hợp và Xử lý lỗi

- **Cấu hình App:** Cần đăng ký ứng dụng infrastructure vào INSTALLED_APPS trong settings.py để Django nhận diện được Models.
- **Lỗi Migration:** Ban đầu chạy makemigrations hệ thống báo No changes detected.
    - Nguyên nhân: Django không tự động quét thư mục infrastructure nếu không chỉ định rõ hoặc thiếu __init__.py.
    - Giải quyết: Chạy lệnh chỉ định rõ app: python manage.py makemigrations infrastructure -> Kết quả thành công tạo BookORM, CartORM.
- **Lỗi Database:** Gặp lỗi Unknown database 'db_clean'. Đã khắc phục bằng cách tạo DB mới cho phiên bản Clean Arch.

3.2.3. Kết quả

API hoạt động chính xác theo luồng Clean Architecture, trả về dữ liệu JSON {"books": []} tại endpoint /api/books/

3.3. Version C - Microservices Django

```
C:\>mkdir ninh_proj1_Microservices

C:\>cd ninh_proj1_Microservices

C:\ninh_proj1_Microservices>django-admin startproject book_service

C:\ninh_proj1_Microservices>django-admin startproject customer_service

C:\ninh_proj1_Microservices>django-admin startproject cart_service

C:\ninh_proj1_Microservices>cd book_service

C:\ninh_proj1_Microservices\book_service>python manage.py startapp books
```

```
C:\ninh_proj1_Microservices\book_service>python manage.py makemigrations
Migrations for 'books':
  books\migrations\0001_initial.py
    + Create model Book

C:\ninh_proj1_Microservices\book_service>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying books.0001_initial... OK
  Applying sessions.0001_initial... OK

C:\ninh_proj1_Microservices\book_service>python manage.py runserver 8001
```

```
C:\ninh_proj1_Microservices\book_service>python manage.py runserver 8001
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2026 - 03:02:44
Django version 5.2.10, using settings 'book_service.settings'
Starting development server at http://127.0.0.1:8001/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server inste
ad.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[20/Jan/2026 03:02:48] "GET /api/books/list/ HTTP/1.1" 200 13
Not Found: /favicon.ico
[20/Jan/2026 03:02:48] "GET /favicon.ico HTTP/1.1" 404 2367
```

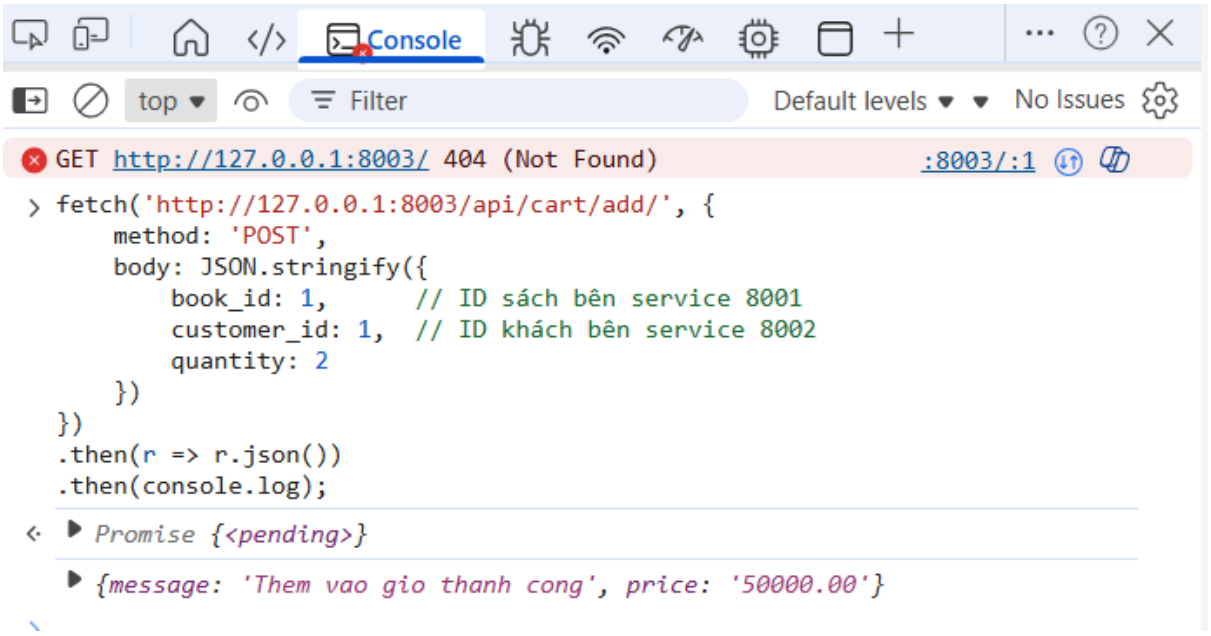
```
C:\ninh_proj1_Microservices>cd customer_service

C:\ninh_proj1_Microservices\customer_service>python manage.py startapp accounts

C:\ninh_proj1_Microservices\customer_service>python manage.py makemigrations
Migrations for 'accounts':
  accounts\migrations\0001_initial.py
    + Create model Customer

C:\ninh_proj1_Microservices\customer_service>python manage.py migrate
Operations to perform:
  Apply all migrations: accounts, admin, auth, contenttypes, sessions
Running migrations:
  Applying accounts.0001_initial... OK
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
```

```
C:\ninh_proj1_Microservices\customer_service>python manage.py runserver 8002
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2026 - 03:05:51
Django version 5.2.10, using settings 'customer_service.settings'
Starting development server at http://127.0.0.1:8002/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server inste
ad.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[20/Jan/2026 03:05:57] "GET /api/customers/list/ HTTP/1.1" 200 17
Not Found: /favicon.ico
[20/Jan/2026 03:05:57] "GET /favicon.ico HTTP/1.1" 404 2375
```

```
C:\ninh_proj1_Microservices>pip install requests
Requirement already satisfied: requests in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from requests) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from requests) (2025.8.3)
```

```
C:\ninh_proj1_Microservices>cd cart_service

C:\ninh_proj1_Microservices\cart_service>python manage.py startapp cart

C:\ninh_proj1_Microservices\cart_service>python manage.py makemigrations
Migrations for 'cart':
  cart\migrations\0001_initial.py
    + Create model Cart
    + Create model CartItem

C:\ninh_proj1_Microservices\cart_service>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, cart, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
```

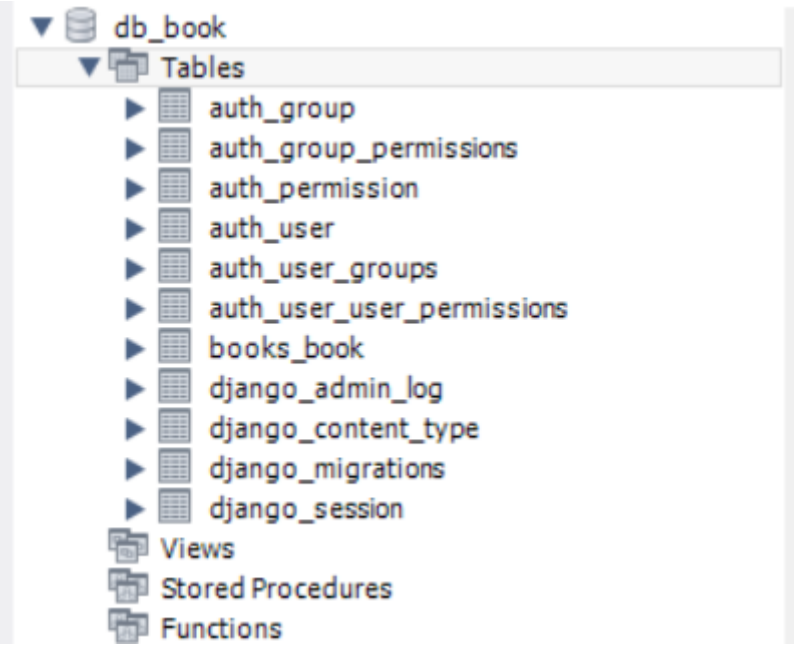
```
C:\ninh_proj1_Microservices\cart_service>python manage.py runserver 8003
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2026 - 03:10:05
Django version 5.2.10, using settings 'cart_service.settings'
Starting development server at http://127.0.0.1:8003/
Quit the server with CTRL-BREAK.
```



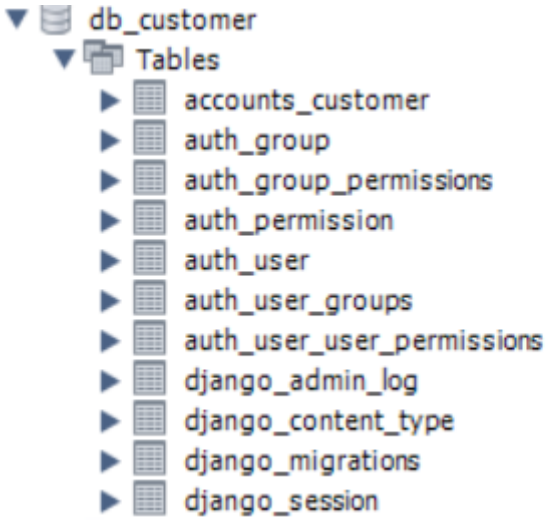
3.3.1. Kiến trúc phân tán

Hệ thống được tách thành 3 dự án Django riêng biệt :

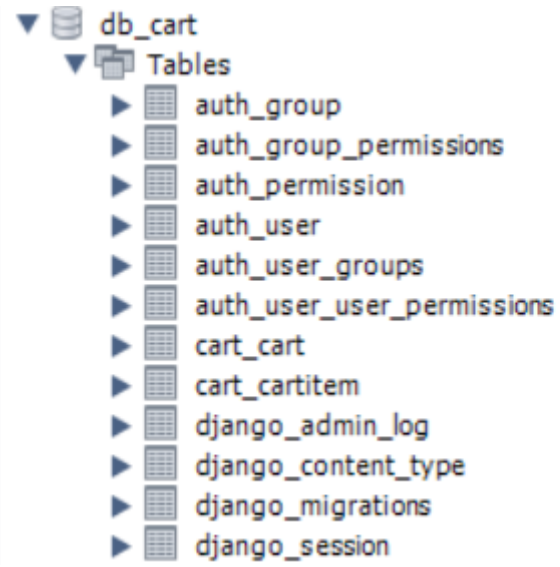
1. **Book Service:** Chạy cổng **8001**. Database: db_book. Quản lý thông tin sách.



2. **Customer Service:** Chạy cổng **8002**. Database: db_customer. Quản lý người dùng.



3. **Cart Service:** Chạy cổng **8003**. Database: db_cart. Xử lý logic giỏ hàng.



3.3.2. Giao tiếp giữa các dịch vụ (Inter-service Communication)

- **Cơ chế:** Cart Service sử dụng thư viện requests để gọi REST API sang Book Service và Customer Service nhằm lấy thông tin giá và xác thực trước khi tạo đơn hàng.

3.3.3. Quá trình tích hợp và Kiểm thử

- **Khởi chạy:** Cả 3 dịch vụ phải được chạy song song trên 3 terminal khác nhau.
- **Lỗi kết nối:**
    - Gặp lỗi ConnectionRefusedError khi Cart Service cố gọi Book Service nhưng Book Service chưa bật hoặc sai cổng.
    - Gặp lỗi Logic Sach khong ton tai ben Book Service. Nguyên nhân do database của Book Service rỗng.
- **Kết quả cuối cùng (Final Integration):**
    - Sau khi bật đủ services và thêm dữ liệu sách mẫu.
    - Thực hiện lệnh fetch từ Console trình duyệt giả lập Client.
    - Hệ thống trả về kết quả thành công: {message: 'Them vao gio thanh cong', price: '50000.00'}.
    - Điều này chứng minh Cart Service (8003) đã giao tiếp thành công với Book Service (8001) để lấy giá tiền.