# Electric Car Garage Microservices

**Domain Analysis of Catalog microservice**

**Domain Model**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Note** |
| ID | Guid |  |
| Name | String |  |
| Categories | List | Repair, Paint, Maintenance, Installation |
| Description | String |  |
| Price | Decimal |  |

**Application Use Cases**

|  |  |
| --- | --- |
| List Offerings and Gategories |  |
| Get Offering by ID |  |
| Get Offerings by category |  |
| Create Offering |  |
| Update Offering |  |
| Delete Offering |  |

**REST API Endpoints**

|  |  |  |
| --- | --- | --- |
| get | /offerings | list all offerings |
| get | /offerings /{id} | fetch single offering |
| get | /offerings /category | get offerings by category |
| post | /offerings | create new offering |
| put | /offerings /{id} | update offering |
| delete | /offerings /{id} | delete offering |

**Data Structures**

Document Database stored as JSON data in PostgreSQL DB as Json columns using Marten library which transforms PostgreSQL into a .net transactional document DB. It’s JSON column features allows to store and query data as JSON documents.

**Technical Analysis of Catalog microservice**

**Application Architecture Style**Vertical Slice architecture.  
Code is organized into feature folders.  
each feature is encapsulated in a single .cs file.  
a feature includes presentation, application, domain and infrastructure.

**Patterns and Principles**CQRS Pattern  
Mediator Pattern  
Dependency injection  
Minimal APIs and Routing  
ORM Pattern – Object relational mapping between database object using high-level code

**Libraries (Nuget Packages)**  
MediatR for CQRS – simplifies inplementation of CQRS pattern  
Carter for API Endpoints – Routing and handling http request, easy to define api endpoints  
Marten for PostgreSQL interaction – document DB using LightWeightDocumentSession  
Mapster for object mapping – maps objects from one type to another  
FluentValidation for input validation – for building strongly-type validation rules, ensure inputs are correct before processing.

**Project Folder Structure**  
- Folders are Model, Features, Data, Abstraction (interface)  
- Feature exp: CreateOffering, GetOffering with dedicated handler and endpoint  
- Data folder is not needed unless you’re dealing with loading initial data to work with  
- Model folder are for entities/dto