

## ساختار درختی پروژه

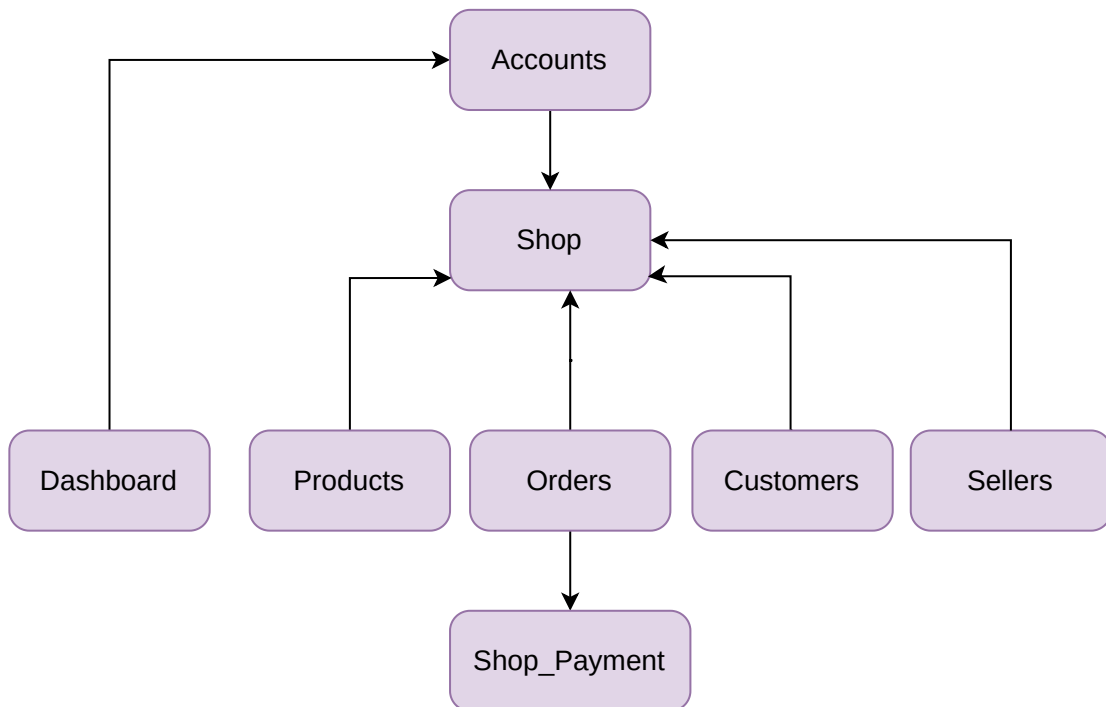
```
insta_crm/  
├── config/  
├── apps/  
│   ├── accounts/  
│   ├── shops/  
│   ├── products/  
│   ├── customers/  
│   ├── orders/  
│   ├── payments/  
│   └── dashboard/  
│   └── ...  
├── common/  
│   ├── utils.py  
│   ├── mixins.py  
│   └── permissions.py  
└── manage.py
```

## ساختار داخل هر اپ

```
insta_crm/  
apps/  
  orders/  
    api/  
      views.py  
      serializers.py  
      urls.py  
    application/  
      create_order.py  
      cancel_order.py  
    domain/  
      entities.py  
      services.py  
    infrastructure/  
      models.py  
      repositories.py  
    products/  
    payments/  
    customers/  
    accounts/
```

## ساختار پوشه تنظیمات

```
config/  
├── settings/  
│   ├── base.py  
│   ├── dev.py  
│   └── prod.py  
├── urls.py  
└── wsgi.py
```



Application Layer (Use Cases) 2

اینجا سناریوهای سیستم نوشته می‌شوند.

هر عملیات مهم = یک Use Case

مثلاً:

Plain text

```
apps/orders/application/  
create_order.py  
cancel_order.py
```

مثال:

Python

```
class CreateOrderUseCase:  
  
    def execute(self, shop, data):  
        # 1 validate  
        # 2 call domain logic  
        # 3 save via repository  
        # 4 return result
```

این لایه:

- منطق سناریویی دارد
- ولی هنوز مستقیم با Django ORM کار نمی‌کند

Domain Layer (قلب سیستم) 3

اینجا قوانین بیزینس هستند.

Plain text

```
apps/orders/domain/  
entities.py  
services.py
```

مثلاً:

Python

```
class Order:  
  
    def __init__(self, customer, items):  
        self.customer = customer  
        self.items = items  
  
    def calculate_total(self):  
        return sum(item.price * item.quantity for item in self.items)
```

یا:

Python

```
class InventoryService:  
  
    def check_inventory(product, quantity):  
        if product.inventory < quantity:  
            raise Exception("Not enough inventory")
```

اینجا هیچ import از Django نباید باشد.

Infrastructure Layer 4

اینجا اتصال به دیتابیس، ORM و فایل سیستم است.

Plain text

```
apps/orders/infrastructure/  
models.py  
repositories.py
```

مثلاً:

Python

```
class OrderRepository:  
  
    def create(self, shop, customer):  
        return OrderModel.objects.create(  
            shop=shop,  
            customer=customer  
        )
```

اینجا از Django ORM استفاده می‌کنیم.

Presentation Layer (لایه API) 1

اینجا فقط:

- گرفتن request
- صدا زدن use case
- برگرداندن response

Plain text

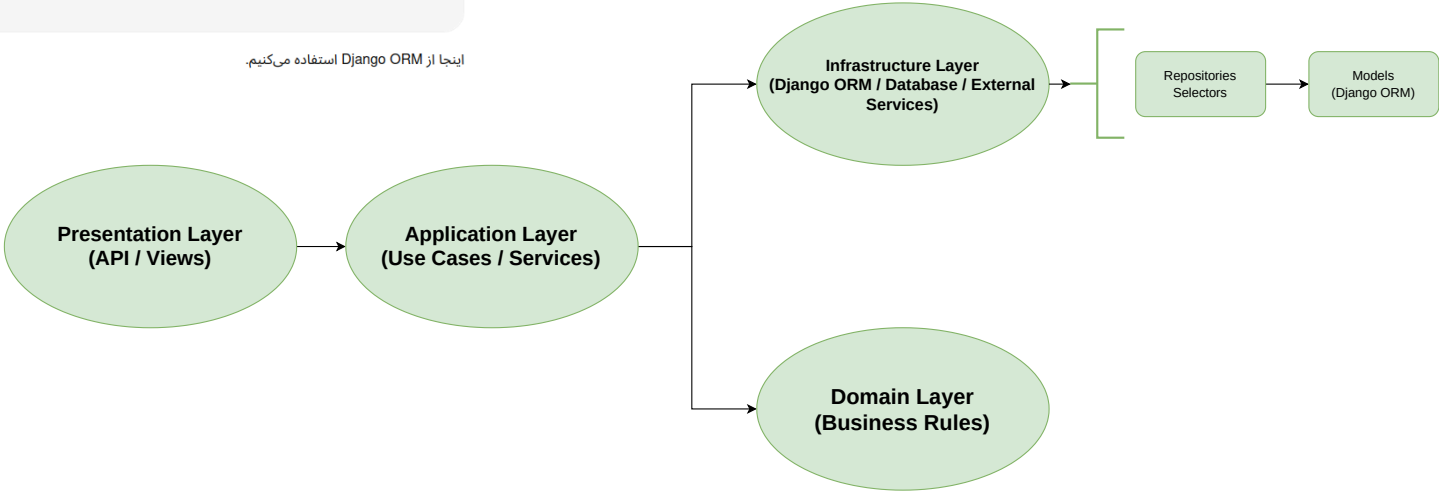
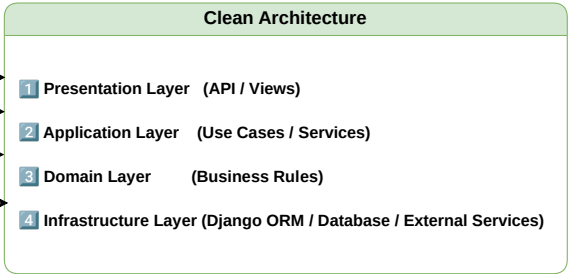
```
apps/orders/api/  
views.py  
serializers.py  
urls.py
```

مثال:

Python

```
class OrderCreateAPIView(APIView):  
    def post(self, request):  
        use_case = CreateOrderUseCase()  
        result = use_case.execute(  
            shop=request.user.shop,  
            data=request.data  
        )  
        return Response(result)
```

اینجا هیچ منطق بیزینسی نیست.



accounts/  
├── models.py  
│ ├── User  
│ └── Profile  
├── serializers.py  
├── views.py  
└── permissions.py

#### UserType

1- regular  
2- admin  
3- superuser

#### User

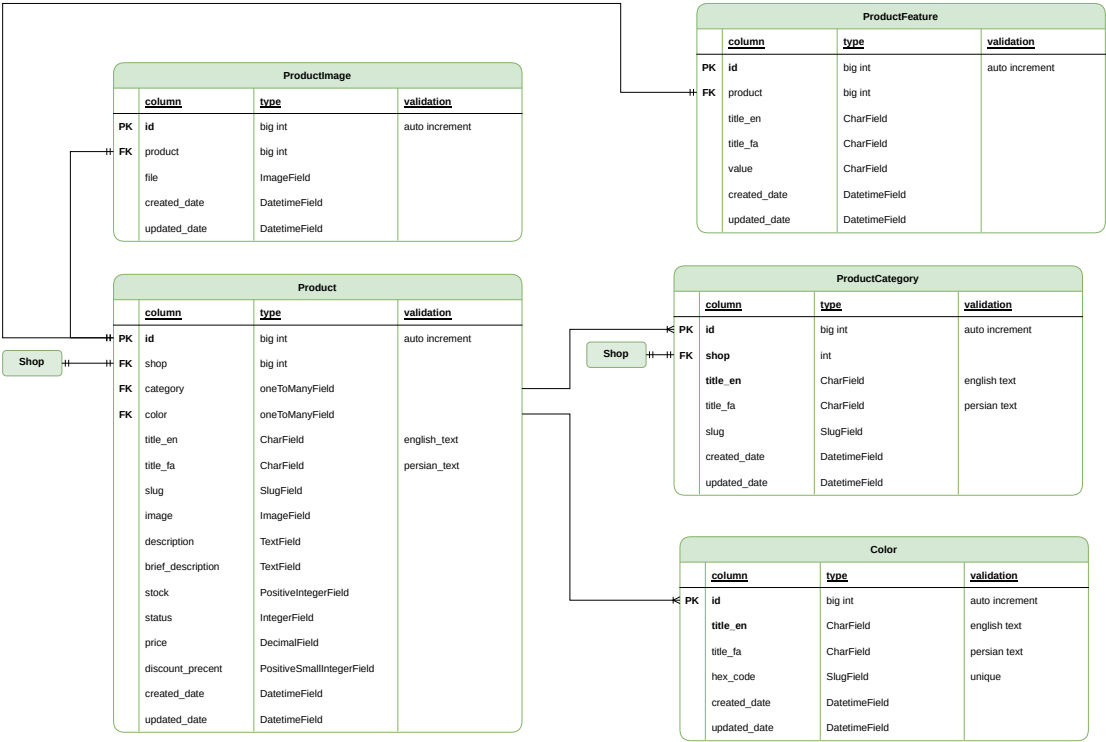
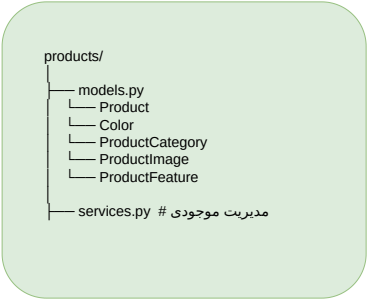
|    | <u>Column</u> | <u>Type</u>   | <u>Validation</u> | <u>Description</u> |
|----|---------------|---------------|-------------------|--------------------|
| PK | id            | BigInt        | auto increment    |                    |
|    | phone         | phone field   | as a user name    |                    |
|    | password      |               |                   |                    |
|    | type          | IntegerField  |                   |                    |
|    | is_active     | BooleanField  |                   |                    |
|    | is_staff      | BooleanField  |                   |                    |
|    | created_date  | DatetimeField |                   |                    |
|    | updated_date  | DatetimeField |                   |                    |

#### Profile

|    | <u>Column</u> | <u>Type</u>   | <u>Validation</u>        | <u>Description</u> |
|----|---------------|---------------|--------------------------|--------------------|
| PK | id            | BigInt        | auto increment           |                    |
| FK | user          | OneToOne      |                          |                    |
|    | first_name    | CharField     | 250 char                 |                    |
|    | last_name     | CharField     | 250 char                 |                    |
|    | email         | email field   | regex for iranian number |                    |
|    | image         | ImageField    |                          |                    |
|    | created_date  | DatetimeField |                          |                    |
|    | updated_date  | DatetimeField |                          |                    |

```
shops/
├── models.py
│   └── Shop
└── services.py # منطق مربوط به اشتراک
```

| Shop    |                     |               |                   |                    |
|---------|---------------------|---------------|-------------------|--------------------|
|         | <u>Column</u>       | <u>Type</u>   | <u>Validation</u> | <u>Description</u> |
| PK      | id                  | BigInt        | auto increment    |                    |
|         | name_en             | CharField     | english text      |                    |
|         | name_fa             | CharField     | persian text      |                    |
|         | owner               | OneToOne      |                   |                    |
|         | instagram_id        | CharField     | unique            |                    |
|         | telegram_channel_id | CharField     | unique            |                    |
|         | is_active           | BooleanField  |                   |                    |
|         | created_date        | DatetimeField |                   |                    |
|         | updated_date        | DatetimeField |                   |                    |
| Profile | FK                  |               |                   |                    |



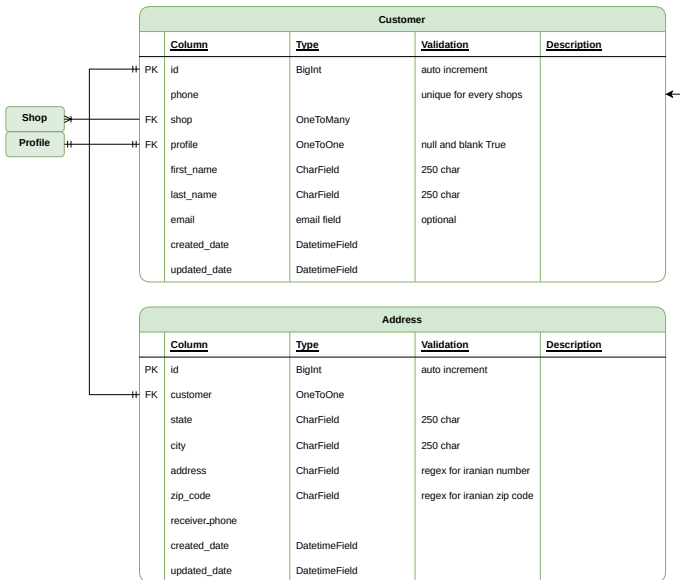
```

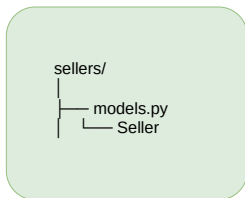
customers/
├── models.py
├── Customer
└── Address

```

Note:

هدف این است که هر فروشگاه برای هر مشتری یک پروفایل داشته باشد. حال ممکن است این مشتری با همین شماره در فروشگاه دیگری هم پروفایل داشته باشد و در این حال در آپ ما هم آکانت داشته باشد. یعنی سه پروفایل متفاوت داشته باشد





| Seller |               |             |                     |                    |
|--------|---------------|-------------|---------------------|--------------------|
|        | <u>Column</u> | <u>Type</u> | <u>Validation</u>   | <u>Description</u> |
| PK     | id            | BigInt      | auto increment      |                    |
|        | FK shop       | OneToMany   |                     |                    |
|        | FK profile    | OneToOne    | null and blank True |                    |

Shop

Profile

```

orders/
├── models.py
│   ├── Order
│   └── OrderItem
├── services.py # منطق ثبت سفارش
└── selectors.py # کوئری‌های پیچیده

```

