

# پروژه فروشگاه آنلاین کتاب

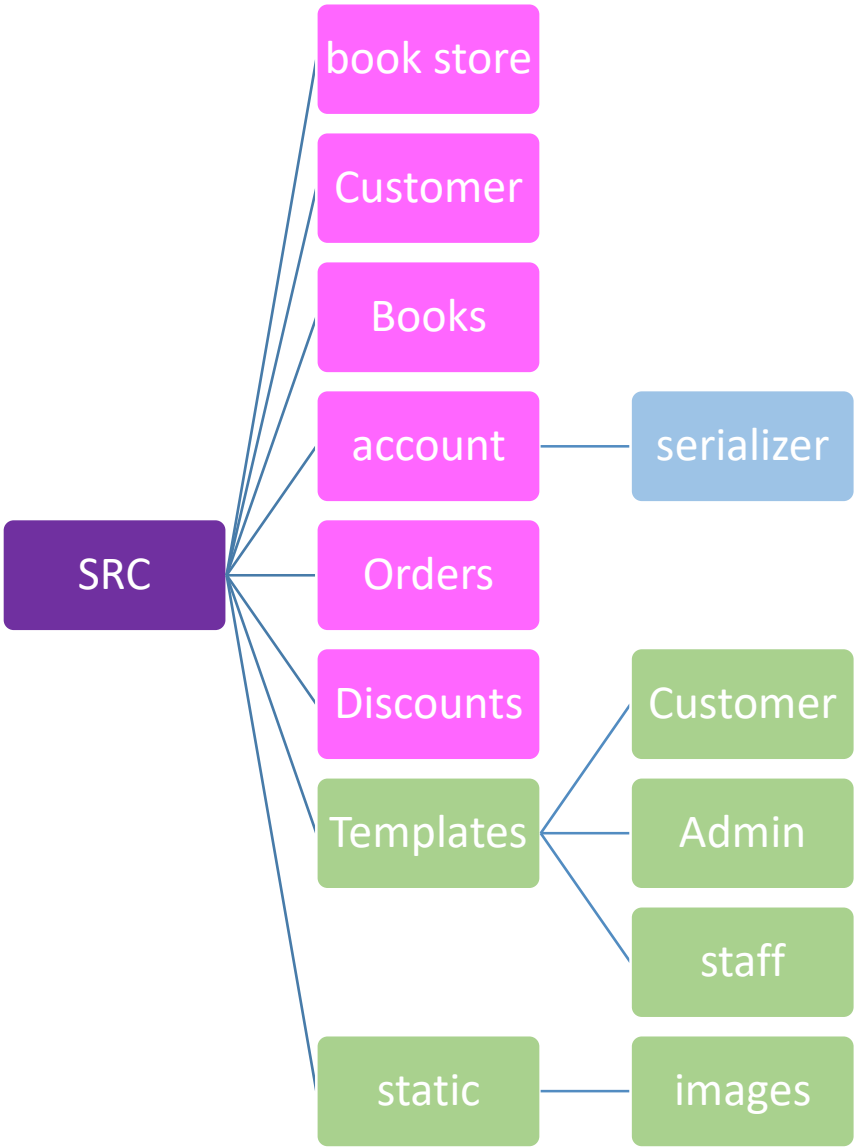
آرزو مرادی چگنی

گزارش فنی

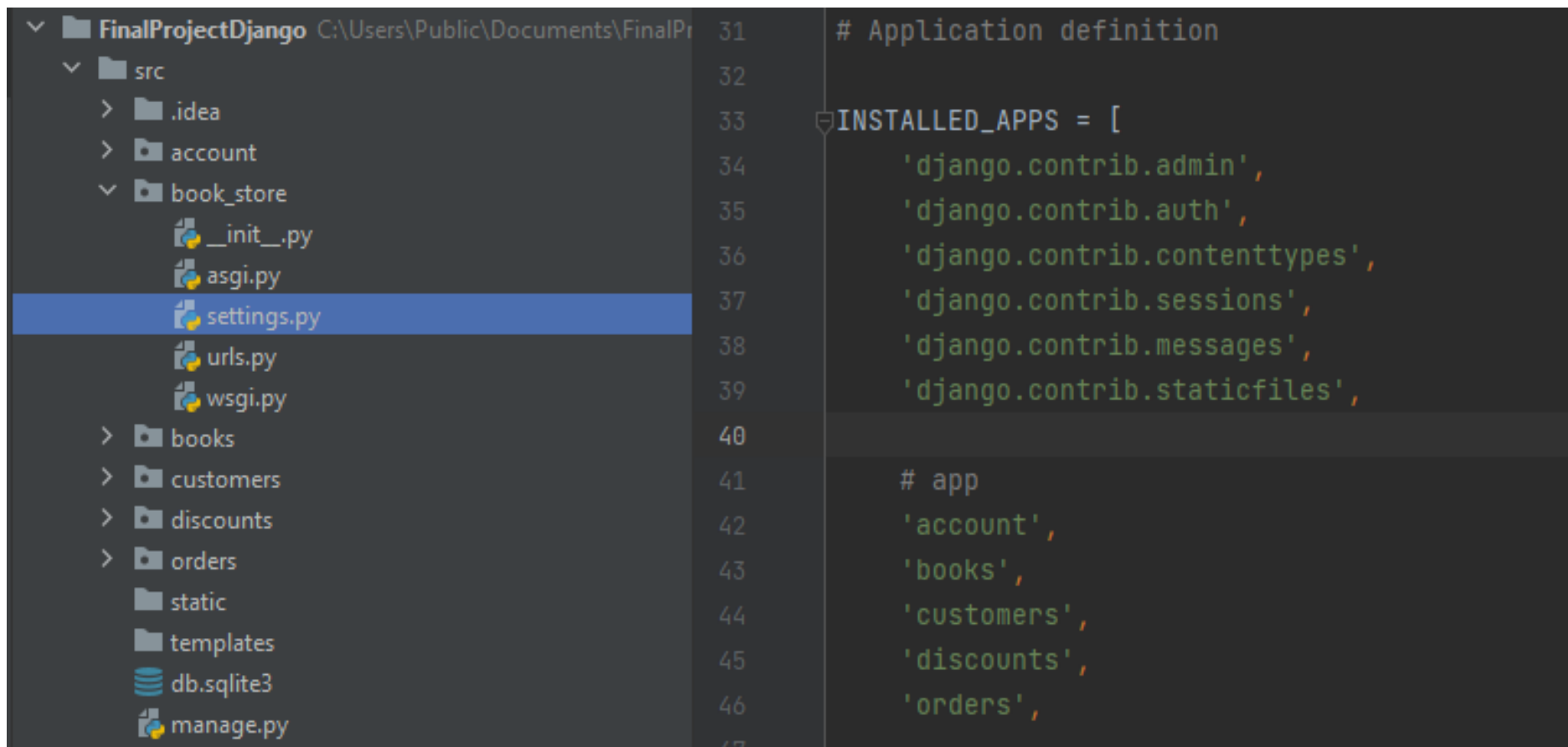
مرداد ماه

- directory
- .py
- package

معماری کلی پروژه به شرح زیر می باشد:

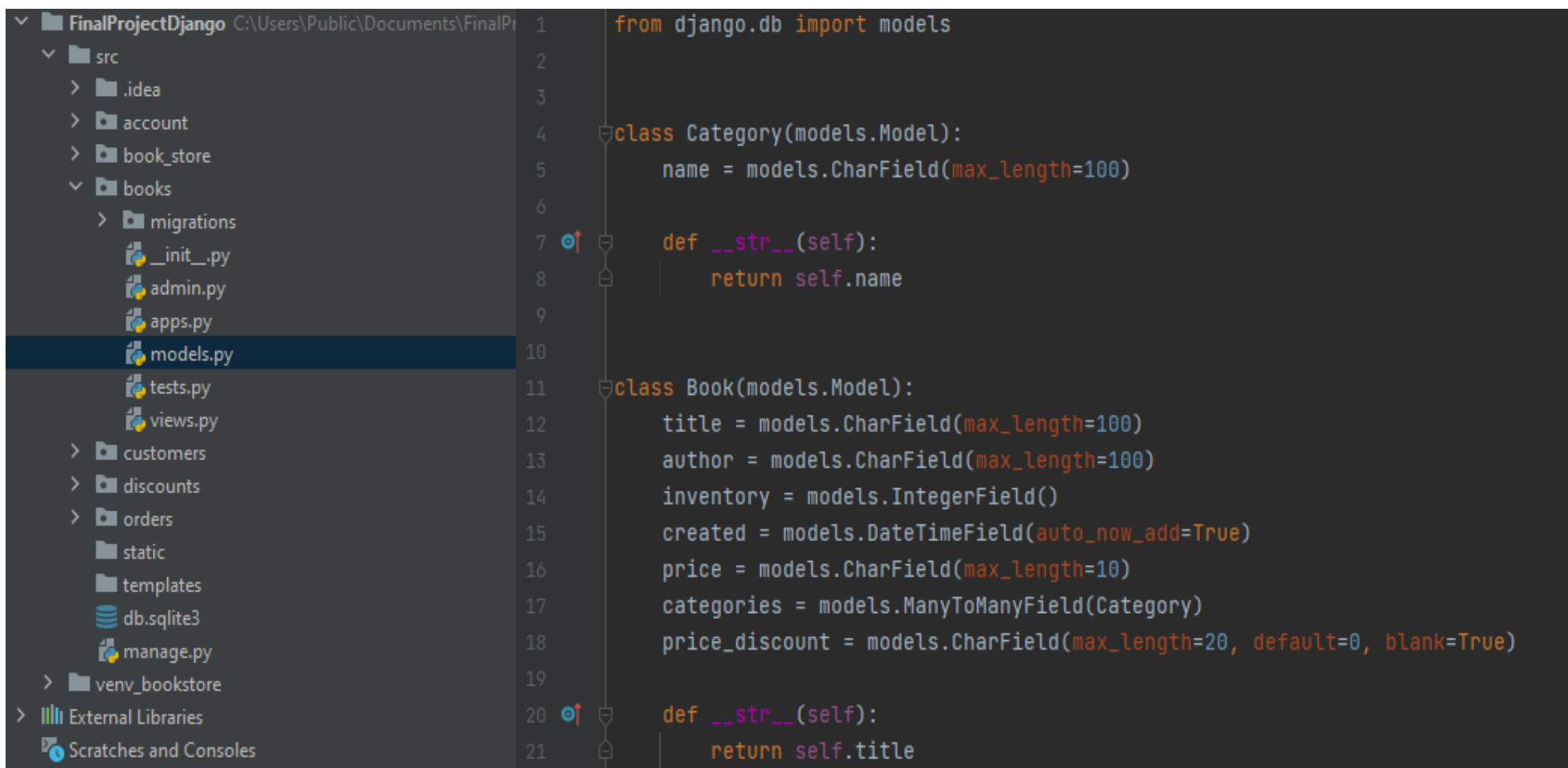


به فایل setting پروژه تمامی آپ‌های ساخته شده رو اضافه می‌کنیم



```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40
41     # app
42     'account',
43     'books',
44     'customers',
45     'discounts',
46     'orders',
47 ]
```

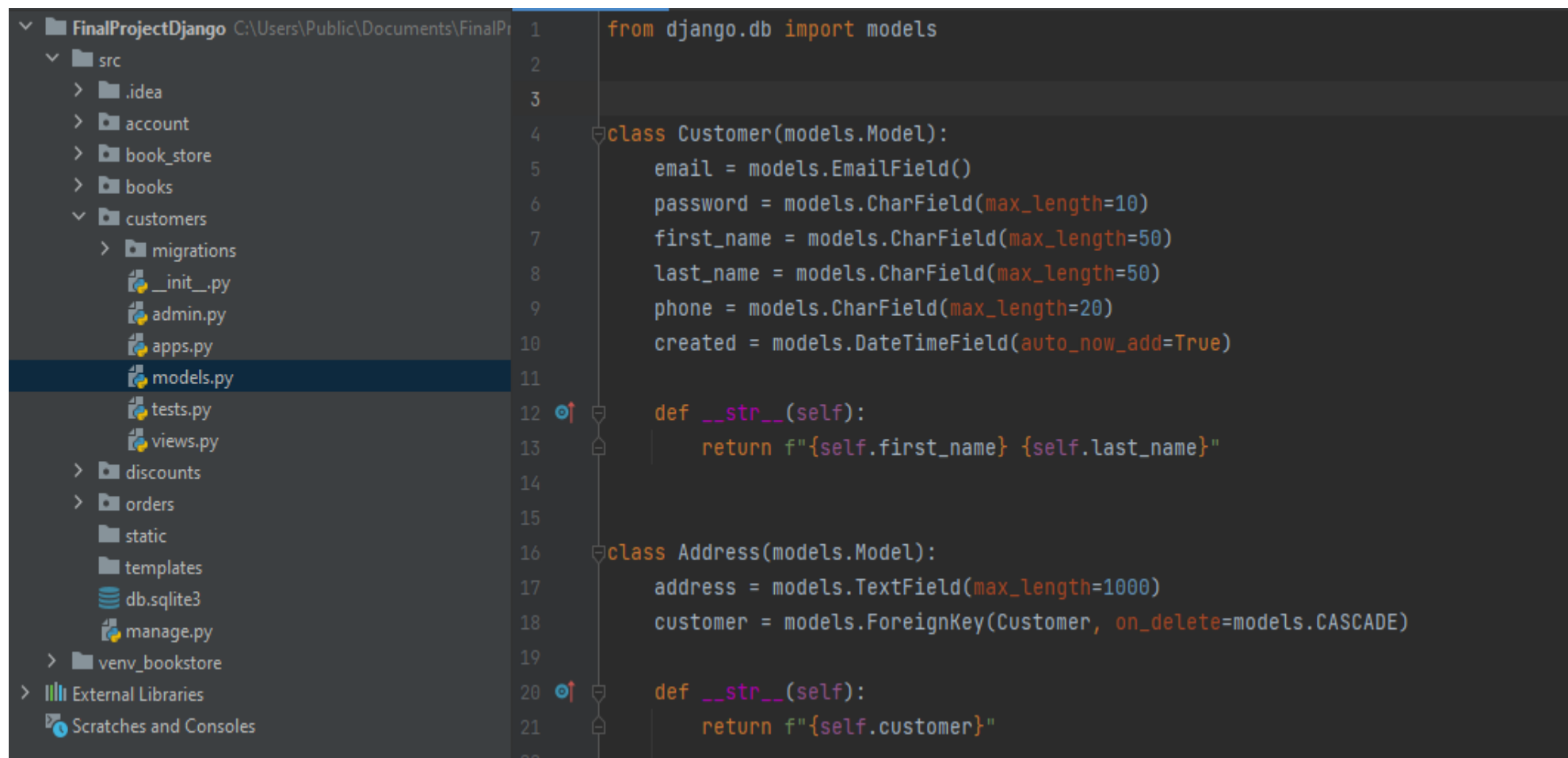
در آپ books، دو مدل Category و Book تعریف شده است.



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'FinalProjectDjango' with a 'src' folder containing several subfolders and files. The 'books' folder is expanded, showing 'migrations', '\_\_init\_\_.py', 'admin.py', 'apps.py', 'models.py' (selected), 'tests.py', and 'views.py'. The code editor shows the content of 'models.py'.

```
1 from django.db import models
2
3
4 class Category(models.Model):
5     name = models.CharField(max_length=100)
6
7     def __str__(self):
8         return self.name
9
10
11 class Book(models.Model):
12     title = models.CharField(max_length=100)
13     author = models.CharField(max_length=100)
14     inventory = models.IntegerField()
15     created = models.DateTimeField(auto_now_add=True)
16     price = models.CharField(max_length=10)
17     categories = models.ManyToManyField(Category)
18     price_discount = models.CharField(max_length=20, default=0, blank=True)
19
20     def __str__(self):
21         return self.title
```

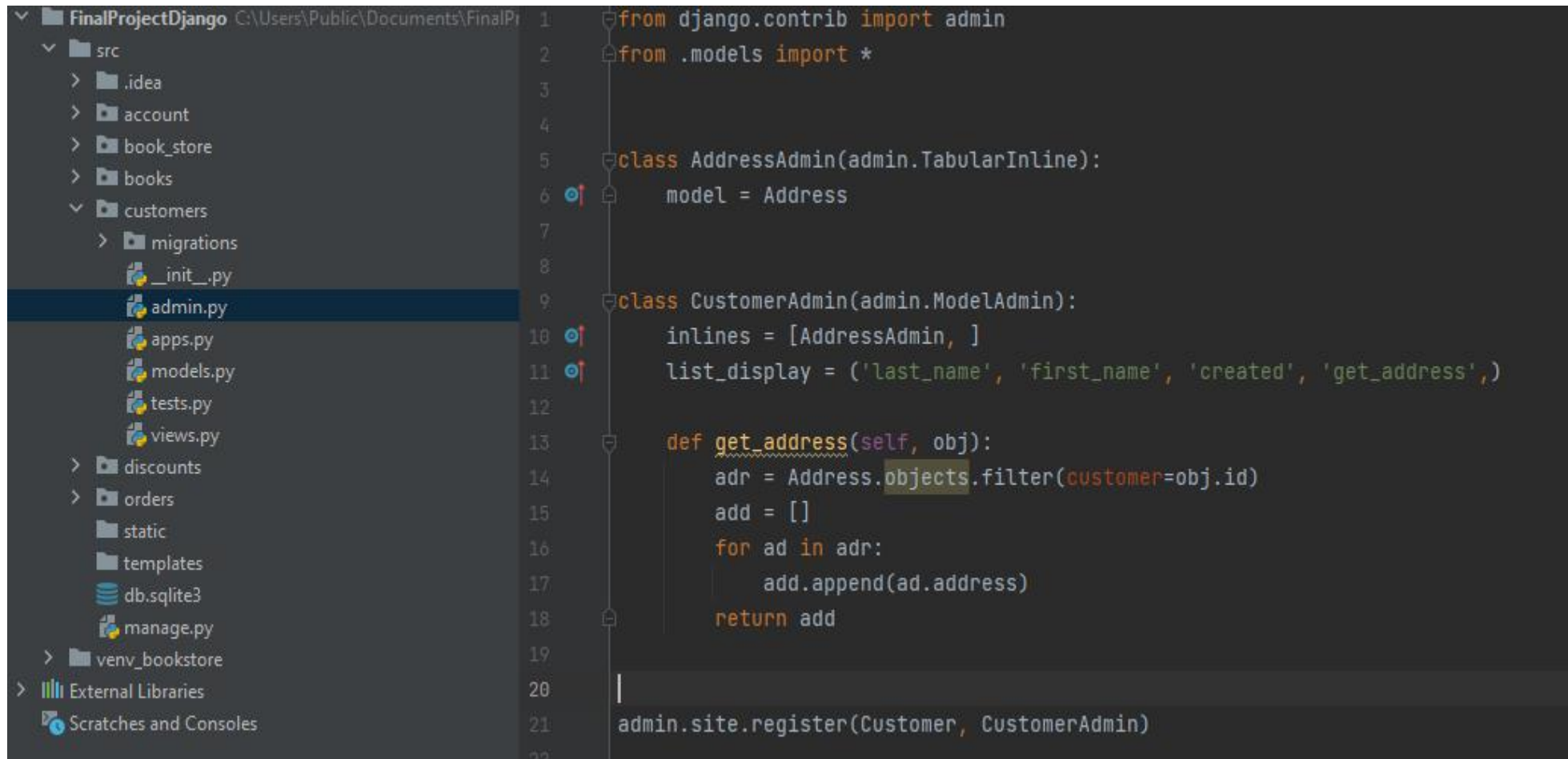
در آپ Customer، دو مدل Customer و Address تعریف شده است.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'FinalProjectDjango' with a 'src' folder containing various subfolders and files. The 'customers' folder is expanded, showing 'migrations', '\_\_init\_\_.py', 'admin.py', 'apps.py', 'models.py' (selected), 'tests.py', and 'views.py'. The code editor shows the content of 'models.py'.

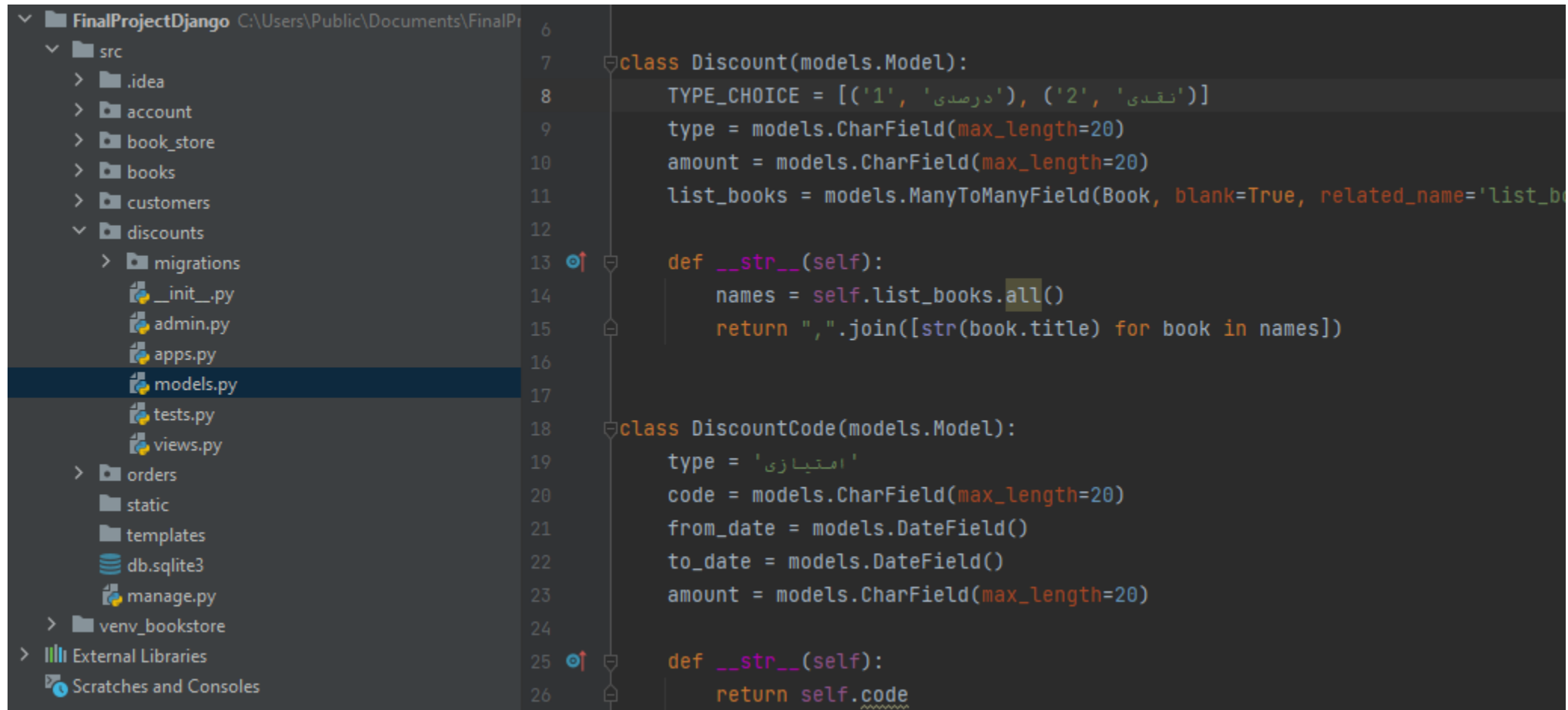
```
1 from django.db import models
2
3
4 class Customer(models.Model):
5     email = models.EmailField()
6     password = models.CharField(max_length=10)
7     first_name = models.CharField(max_length=50)
8     last_name = models.CharField(max_length=50)
9     phone = models.CharField(max_length=20)
10    created = models.DateTimeField(auto_now_add=True)
11
12    def __str__(self):
13        return f"{self.first_name} {self.last_name}"
14
15
16 class Address(models.Model):
17     address = models.TextField(max_length=1000)
18     customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
19
20    def __str__(self):
21        return f"{self.customer}"
22
```

برای کاستومایز کردن Customer و Address به شکل زیر عمل شده است.  
برای اینکه یک مشتری بتواند چندین آدرس داشته باشد از inline استفاده شده است. حال با استفاده از list\_display فیلدهای مورد نیاز را نمایش میدهیم.  
با استفاده از تابع get\_address آدرس‌های هر مشتری براساس id آن دریافت و برای نمایش آن‌ها از یک لیست استفاده میکنیم.



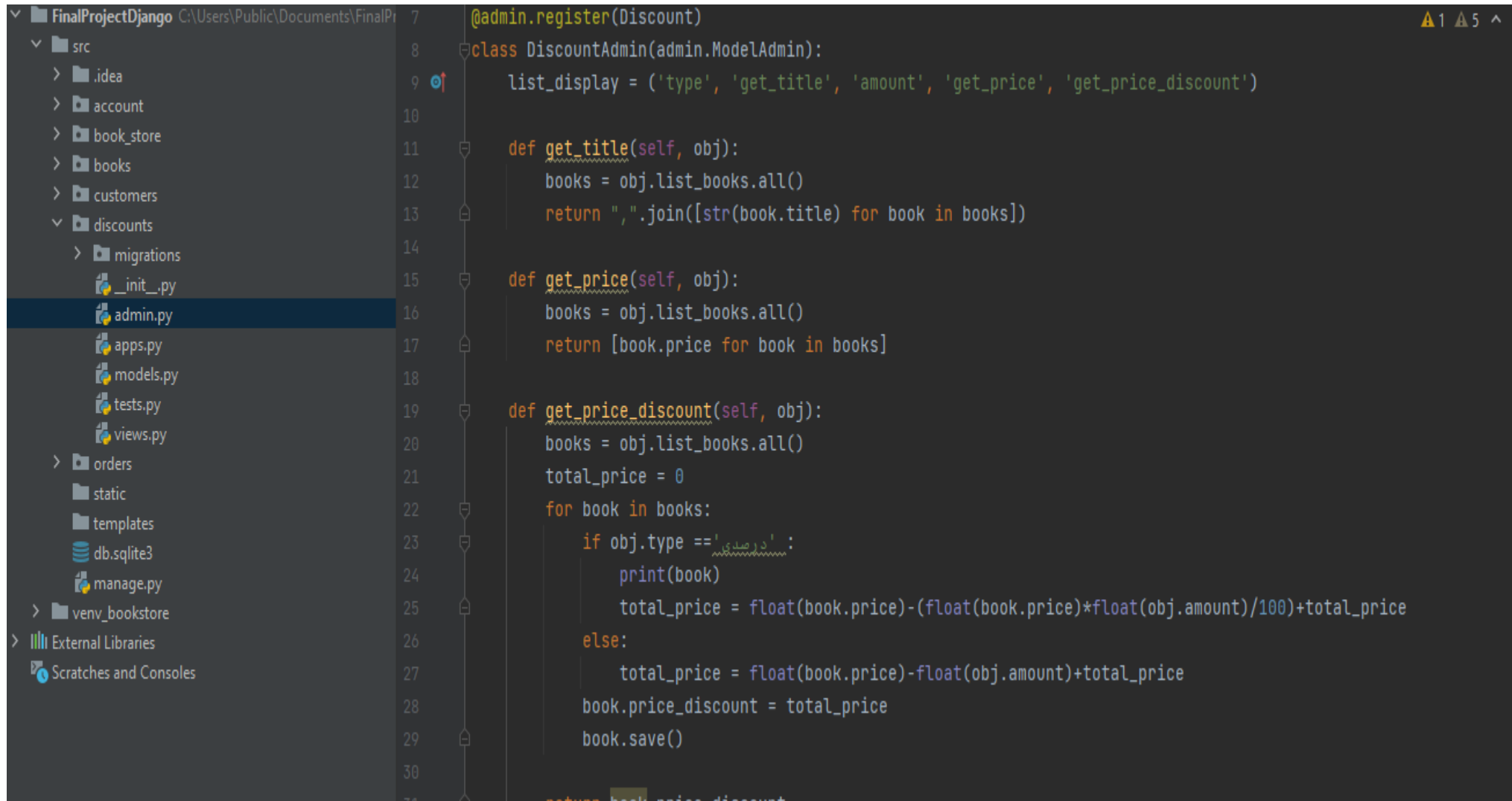
```
1 from django.contrib import admin
2 from .models import *
3
4
5 class AddressAdmin(admin.TabularInline):
6     model = Address
7
8
9 class CustomerAdmin(admin.ModelAdmin):
10     inlines = [AddressAdmin, ]
11     list_display = ('last_name', 'first_name', 'created', 'get_address',)
12
13     def get_address(self, obj):
14         adr = Address.objects.filter(customer=obj.id)
15         add = []
16         for ad in adr:
17             add.append(ad.address)
18         return add
19
20
21 admin.site.register(Customer, CustomerAdmin)
```

در آپدیت‌ها، دو مدل ساخته شده است: مدل Discount برای تخفیف‌های درصدی و نقدی و مدل Discountcode برای تخفیف امتیازی در نظر گرفته شده است.



```
6
7 class Discount(models.Model):
8     TYPE_CHOICE = [('1', 'درصدی'), ('2', 'نقدی')]
9     type = models.CharField(max_length=20)
10    amount = models.CharField(max_length=20)
11    list_books = models.ManyToManyField(Book, blank=True, related_name='list_b
12
13    def __str__(self):
14        names = self.list_books.all()
15        return ",".join([str(book.title) for book in names])
16
17
18 class DiscountCode(models.Model):
19     type = 'امتیازی'
20     code = models.CharField(max_length=20)
21     from_date = models.DateField()
22     to_date = models.DateField()
23     amount = models.CharField(max_length=20)
24
25    def __str__(self):
26        return self.code
```

برای کاستومایز کردن Discount به شکل زیر عمل شده است. حال برای نمایش فیلدهای آن از `list_display` استفاده شده است. از تابع `get_title` برای گرفتن اسمی کتاب‌ها و از تابع `get_price` برای گرفتن قیمت هر کتاب و از تابع `get_price_discount` برای گرفتن قیمت هر کتاب با اعمال تخفیف درصدی یا نقدی بر روی آن است.



```
@admin.register(Discount)
class DiscountAdmin(admin.ModelAdmin):
    list_display = ('type', 'get_title', 'amount', 'get_price', 'get_price_discount')

    def get_title(self, obj):
        books = obj.list_books.all()
        return ",".join([str(book.title) for book in books])

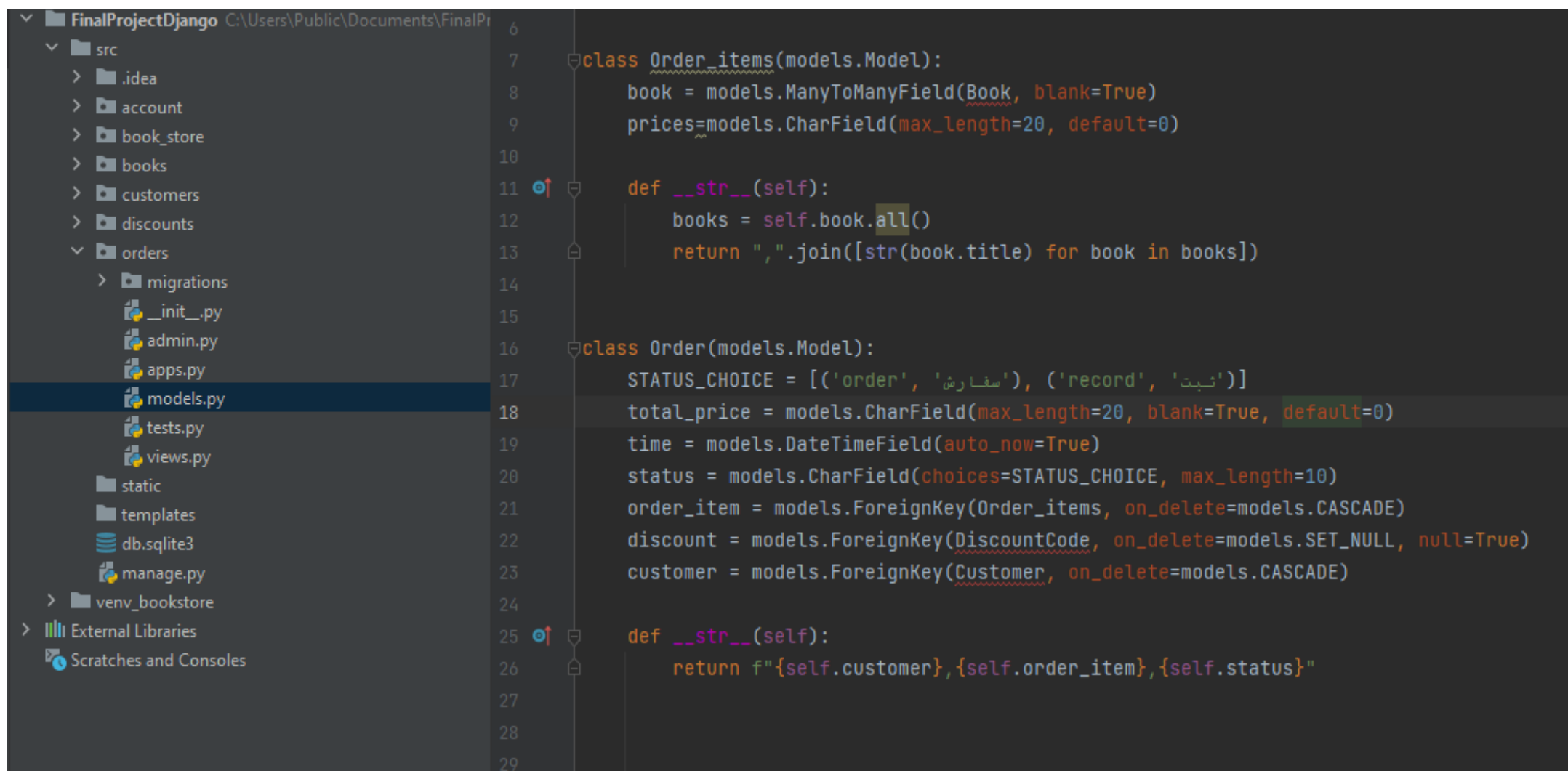
    def get_price(self, obj):
        books = obj.list_books.all()
        return [book.price for book in books]

    def get_price_discount(self, obj):
        books = obj.list_books.all()
        total_price = 0
        for book in books:
            if obj.type == 'درصدی':
                print(book)
                total_price = float(book.price) - (float(book.price) * float(obj.amount) / 100) + total_price
            else:
                total_price = float(book.price) - float(obj.amount) + total_price
        book.price_discount = total_price
        book.save()

    return book.price_discount
```



در آپ orders دو مدل orders\_items و order تعریف شده است.



```
6
7 class Order_items(models.Model):
8     book = models.ManyToManyField(Book, blank=True)
9     prices=models.CharField(max_length=20, default=0)
10
11 def __str__(self):
12     books = self.book.all()
13     return ",".join([str(book.title) for book in books])
14
15
16 class Order(models.Model):
17     STATUS_CHOICE = [('order', 'سفارش'), ('record', 'ثبت')]
18     total_price = models.CharField(max_length=20, blank=True, default=0)
19     time = models.DateTimeField(auto_now=True)
20     status = models.CharField(choices=STATUS_CHOICE, max_length=10)
21     order_item = models.ForeignKey(Order_items, on_delete=models.CASCADE)
22     discount = models.ForeignKey(DiscountCode, on_delete=models.SET_NULL, null=True)
23     customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
24
25 def __str__(self):
26     return f"{self.customer},{self.order_item},{self.status}"
27
28
29
```

برای کاستومایز کردن orders به شکل زیر عمل شده است. حال از تابع `get_title` برای گرفتن نام هر کتاب و از تابع `get_price` برای گرفتن قیمت هر کتاب و هم چنین تابع `calculate_price` برای محاسبه قیمت هر کتاب می باشد.

```
src
├── .idea
├── account
├── book_store
├── books
├── customers
├── discounts
├── orders
│   ├── migrations
│   ├── __init__.py
│   └── admin.py
├── apps.py
├── models.py
├── tests.py
├── views.py
├── static
├── templates
├── db.sqlite3
├── manage.py
├── venv_bookstore
├── External Libraries
└── Scratches and Consoles
```

```
9 @admin.register(Order_items)
10 class OrderItemAdmin(admin.ModelAdmin):
11     list_display = ('get_title', 'get_price')
12
13     def get_title(self, obj):
14         books = obj.book.all()
15
16         return ",".join([str(book.title) for book in books])
17
18     def get_price(self, obj):
19         books = obj.book.all()
20
21         lst_price = []
22         for book in books:
23             if book.price_discount != '0':
24                 lst_price.append(float(book.price_discount))
25             else:
26                 lst_price.append(float(book.price))
27         total = sum(lst_price)
28
29         obj.prices = total
30         obj.save()
31         return total
```

```
src
├── .idea
├── account
├── book_store
├── books
├── customers
├── discounts
├── orders
│   ├── migrations
│   ├── __init__.py
│   └── admin.py
├── apps.py
├── models.py
├── tests.py
├── views.py
├── static
├── templates
├── db.sqlite3
├── manage.py
├── venv_bookstore
├── External Libraries
└── Scratches and Consoles
```

```
33
34 @admin.register(Order)
35 class OrdersAdmin(admin.ModelAdmin):
36     list_display = ('customer', 'order_item', 'calculate_price', 'time')
37
38     def calculate_price(self, obj):
39
40         prices = int(obj.order_item.prices)
41
42         dis = obj.discount.amount
43
44         if dis != "":
45             total = prices - prices * int(dis) / 100
46             print(total)
47         else:
48             total = prices
49         obj.total_price = total
50         obj.save()
51         return total
```