Санкт-Петербургский Политехнический Университет Петра Великого

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №1

Курс: «Базы данных»

Тема: «Знакомство с ORM на примере Django»

Выполнил студент:

Бояркин Никита Сергеевич
Группа: 43501/3

Проверил:

Мяснов Александр Владимирович

Санкт-Петербург
2017 г.

Содержание

Лабораторная работа №1

1.1 Цель работы

Получить практические навыки работы с БД через механизм объектно-реляционного отображения.

1.2 Программа работы

- Знакомство с фреймворком Django (установка, создание проекта, конфигурирование).
- Формирование набора моделей, соответствующих схеме БД, полученной по результатам разработки схемы БД и модификации схемы.
- Знакомство с механизмом миграций: автоматическое формирование схемы БД с помощью миграций.
- Создание manage-команд для заполнения БД тестовыми данными (по несколько записей в каждой таблице).

1.3 Программное окружение

- Python 3.6
- Django 1.10.5
- Psycopg 2.6.2
- PostgreSQL 9.5.6

1.4 Ход работы

1.4.1 Создание проекта, конфигурация сервера

Структура проекта:

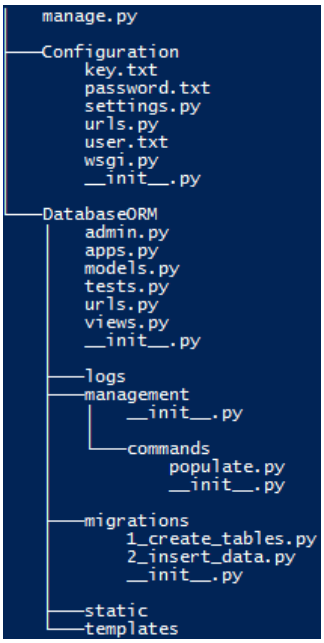

Рис. 1.1: Структура проекта

В проекте присутствует: главный файл проекта *manage.py*; конфигурационный модуль *Configuration*, содержащий конфигурационный файл *settings.py*; основной модуль, содержащий описание моделей (*models.py*), собственные команды (*management/commands/**), миграции (*migrations/**) и др.

Изменения в конфигурационном файле:

```
...

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'DatabaseORM',
]

...

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'pharmacy',
        'USER': open('Configuration/user.txt', 'r').readline().rstrip(),
        'PASSWORD': open('Configuration/password.txt', 'r').readline().rstrip(),
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}

...
```

Логин, пароль и секретный ключ хранятся в специальных файлах, которые не добавляются в систему контроля версий. Также в базу данных по умолчанию добавляется несколько таблиц, связанных с аутентификацией.

1.4.2 Миграция базы данных

С помощью механизма миграций воссоздадим следующую схему базы данных:

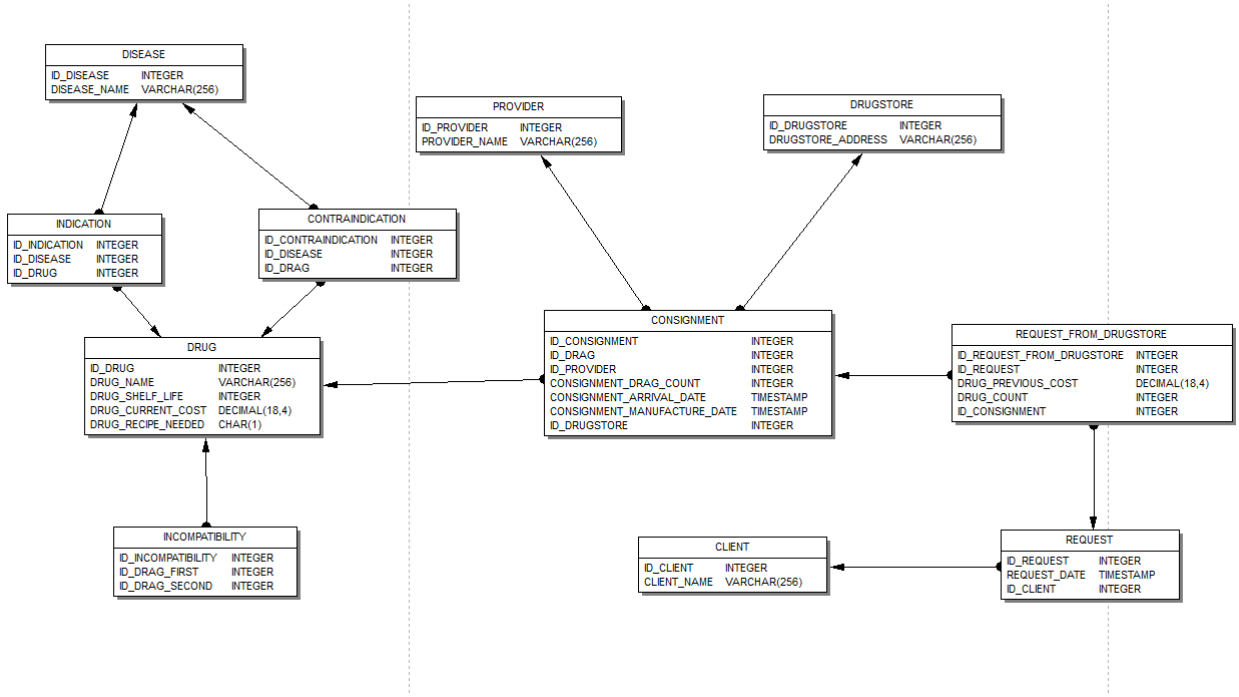

Рис. 1.2: Схема базы данных

Опишем модели в файле *models.py*:

```
from django.db import models

# Create your models here.

DECIMAL_MAX_LENGTH = 18
DECIMAL_PLACES = 4
NAME_LENGTH = 256
ADDRESS_LENGTH = 512


class Disease(models.Model):
    id_disease = models.IntegerField(db_column='id_disease', primary_key=True, unique=True, null=False)
    disease_name = models.CharField(db_column='disease_name', max_length=NAME_LENGTH, unique=True, null=False)

    class Meta:
        db_table = 'disease'


class Drug(models.Model):
    id_drug = models.IntegerField(db_column='id_drug', primary_key=True, unique=True, null=False)
    drug_name = models.CharField(db_column='drug_name', max_length=NAME_LENGTH, unique=True, null=False)
    drug_shelf_life = models.IntegerField(db_column='drug_shelf_life', unique=False, null=False)
    drug_current_cost = models.DecimalField(db_column='drug_current_cost', max_digits=DECIMAL_MAX_LENGTH, decimal_places=DECIMAL_PLACES, unique=False, null=False)
    drug_recipe_needed = models.BooleanField(db_column='drug_recipe_needed', unique=False, null=False)

    class Meta:
        db_table = 'drug'
```

```
28
29
30 class Indication(models.Model):
31     id_indication = models.IntegerField(db_column='id_indication', primary_key=True,
      unique=True, null=False)
32     id_disease = models.ForeignKey(Disease, db_column='id_disease', unique=False, null=
      False)
33     id_drug = models.ForeignKey(Drug, db_column='id_drug', unique=False, null=False)
34
35     class Meta:
36         db_table = 'indication'
37
38
39 class Contraindication(models.Model):
40     id_contraindication = models.IntegerField(db_column='id_contraindication',
      primary_key=True, unique=True, null=False)
41     id_disease = models.ForeignKey(Disease, db_column='id_disease', unique=False, null=
      False)
42     id_drug = models.ForeignKey(Drug, db_column='id_drug', unique=False, null=False)
43
44     class Meta:
45         db_table = 'contraindication'
46
47
48 class Incompatibility(models.Model):
49     id_incompatibility = models.IntegerField(db_column='id_incompatibility', primary_key=
      True, unique=True, null=False)
50     id_drug_first = models.ForeignKey(Drug, db_column='id_drug_first', related_name='
      id_drug_first', unique=False, null=False)
51     id_drug_second = models.ForeignKey(Drug, db_column='id_drug_second', related_name='
      id_drug_second', unique=False, null=False)
52
53     class Meta:
54         db_table = 'incompatibility'
55
56
57 class Provider(models.Model):
58     id_provider = models.IntegerField(db_column='id_provider', primary_key=True, unique=
      True, null=False)
59     provider_name = models.CharField(db_column='provider_name', max_length=NAME_LENGTH,
      unique=True, null=False)
60
61     class Meta:
62         db_table = 'provider'
63
64
65 class Drugstore(models.Model):
66     id_drugstore = models.IntegerField(db_column='id_drugstore', primary_key=True, unique
      =True, null=False)
67     drugstore_address = models.CharField(db_column='drugstore_address', max_length=
      ADDRESS_LENGTH, unique=True, null=False)
68
69     class Meta:
70         db_table = 'drugstore'
71
72
73 class Client(models.Model):
74     id_client = models.IntegerField(db_column='id_client', primary_key=True, unique=True,
       null=False)
75     client_name = models.CharField(db_column='client_name', max_length=NAME_LENGTH,
      unique=False, null=False)
76
77     class Meta:
78         db_table = 'client'
79
80
```

```
class RequestPart(models.Model):
    id_request_part = models.IntegerField(db_column='id_request_part', primary_key=True, unique=True, null=False)
    id_client = models.ForeignKey(Client, db_column='id_client', unique=False, null=False)
    request_part_date = models.DateField(db_column='request_part_date', auto_now=False, auto_now_add=False, unique=False, null=False)

    class Meta:
        db_table = 'request_part'


class Consignment(models.Model):
    id_consignment = models.IntegerField(db_column='id_consignment', primary_key=True, unique=True, null=False)
    id_drug = models.ForeignKey(Drug, db_column='id_drug', unique=False, null=False)
    id_provider = models.ForeignKey(Provider, db_column='id_provider', unique=False, null=False)
    id_drugstore = models.ForeignKey(Drugstore, db_column='id_drugstore', unique=False, null=False)
    consignment_drug_count = models.IntegerField(db_column='consignment_drug_count', unique=False, null=False)
    consignment_arrival_date = models.DateField(db_column='consignment_arrival_date', auto_now=False, auto_now_add=False, unique=False, null=False)
    consignment_manufacture_date = models.DateField(db_column='consignment_manufacture_date', auto_now=False, auto_now_add=False, unique=False, null=False)

    class Meta:
        db_table = 'consignment'


class Request(models.Model):
    id_request = models.IntegerField(db_column='id_request', primary_key=True, unique=True, null=False)
    id_request_part = models.ForeignKey(RequestPart, db_column='id_request_part', unique=False, null=False)
    id_consignment = models.ForeignKey(Consignment, db_column='id_consignment', unique=False, null=False)
    request_drug_previous_cost = models.DecimalField(db_column='request_drug_previous_cost', max_digits=DECIMAL_MAX_LENGTH, decimal_places=DECIMAL_PLACES, unique=False, null=False)
    request_count = models.IntegerField(db_column='request_count', unique=False, null=False)

    class Meta:
        db_table = 'request'
```

После этого была создана миграция командой *makemigrations*, переименуем миграцию *0001_initial.py* в *1_create_tables.py* для большей читабельности.

Для запуска миграции воспользуемся следующим кодом:

```
manage.py@Server > migrate DatabaseORM 1_create_tables
Operations to perform:
  Target specific migration: 1_create_tables, from DatabaseORM
Running migrations:
  Applying DatabaseORM.1_create_tables... OK
```

Рис. 1.3: Запуск миграции

Проверим, добавились ли таблицы:

```
pharmacy=# \dt
                    ╤яшёюъ юЄэю°хэшщ
                          ╚ь
 ╤їхьр  |                          |  ╥шя   | ┬ырфхыхЎ
--------+--------------------------+--------+----------
 public | auth_group               | Єрсышўр | postgres
 public | auth_group_permissions   | Єрсышўр | postgres
 public | auth_permission          | Єрсышўр | postgres
 public | auth_user                | Єрсышўр | postgres
 public | auth_user_groups         | Єрсышўр | postgres
 public | auth_user_user_permissions | Єрсышўр | postgres
 public | client                   | Єрсышўр | postgres
 public | consignment              | Єрсышўр | postgres
 public | contraindication         | Єрсышўр | postgres
 public | disease                  | Єрсышўр | postgres
 public | django_admin_log         | Єрсышўр | postgres
 public | django_content_type      | Єрсышўр | postgres
 public | django_migrations        | Єрсышўр | postgres
 public | django_session           | Єрсышўр | postgres
 public | drug                     | Єрсышўр | postgres
 public | drugstore                | Єрсышўр | postgres
 public | incompatibility          | Єрсышўр | postgres
 public | indication               | Єрсышўр | postgres
 public | provider                 | Єрсышўр | postgres
 public | request                  | Єрсышўр | postgres
 public | request_part             | Єрсышўр | postgres
(21 ёЄЁюър)
```

Рис. 1.4: Созданные таблицы

Откатим миграцию:

```
manage.py@Server > migrate DatabaseORM zero
Operations to perform:
  Unapply all migrations: DatabaseORM
Running migrations:
  Rendering model states... DONE
  Unapplying DatabaseORM.1_create_tables... OK
```

Рис. 1.5: Откат миграции

Проверим, удалились ли таблицы:

```
pharmacy=# \dt
                    ╤яшёюъ юЄэю°хэшщ
                          ╚ь
 ╤їхьр  |                          |  ╥шя   | ┬ырфхыхЎ
--------+--------------------------+--------+----------
 public | auth_group               | Єрсышўр | postgres
 public | auth_group_permissions   | Єрсышўр | postgres
 public | auth_permission          | Єрсышўр | postgres
 public | auth_user                | Єрсышўр | postgres
 public | auth_user_groups         | Єрсышўр | postgres
 public | auth_user_user_permissions | Єрсышўр | postgres
 public | django_admin_log         | Єрсышўр | postgres
 public | django_content_type      | Єрсышўр | postgres
 public | django_migrations        | Єрсышўр | postgres
 public | django_session           | Єрсышўр | postgres
(10 ёЄЁюъ)
```

Рис. 1.6: Оставшиеся таблицы

1.4.3 Добавление данных

Попробуем добавить данные двумя способами: с помощью миграций и с помощью команд. Была создана миграция *2_create_data.py*, в которой в таблицы добавляются данные:

```
manage.py@Server > migrate DatabaseORM
Operations to perform:
  Apply all migrations: DatabaseORM
Running migrations:
  Applying DatabaseORM.1_create_tables... OK
  Applying DatabaseORM.2_insert_data... OK
```

Рис. 1.7: Запуск всех миграций

Были созданы таблицы с данными:

```
pharmacy=# \dt
                 ╥яшёюъ ю€Єю°хэшщ
 ╥їхьр  |            ╚ь            |   ╥шя    | ╥ырфхыхЎ
--------+---------------------------+---------+----------
 public | auth_group                | ЄрсьшЎр | postgres
 public | auth_group_permissions    | ЄрсьшЎр | postgres
 public | auth_permission           | ЄрсьшЎр | postgres
 public | auth_user                 | ЄрсьшЎр | postgres
 public | auth_user_groups          | ЄрсьшЎр | postgres
 public | auth_user_user_permissions| ЄрсьшЎр | postgres
 public | client                    | ЄрсьшЎр | postgres
 public | consignment               | ЄрсьшЎр | postgres
 public | contraindication          | ЄрсьшЎр | postgres
 public | disease                   | ЄрсьшЎр | postgres
 public | django_admin_log          | ЄрсьшЎр | postgres
 public | django_content_type       | ЄрсьшЎр | postgres
 public | django_migrations         | ЄрсьшЎр | postgres
 public | django_session            | ЄрсьшЎр | postgres
 public | drug                      | ЄрсьшЎр | postgres
 public | drugstore                 | ЄрсьшЎр | postgres
 public | incompatibility           | ЄрсьшЎр | postgres
 public | indication                | ЄрсьшЎр | postgres
 public | provider                  | ЄрсьшЎр | postgres
 public | request                   | ЄрсьшЎр | postgres
 public | request_part              | ЄрсьшЎр | postgres
(21 ёЄЁ№ъЁ)

pharmacy=# select * from drug;
 id_drug | drug_name  | drug_shelf_life | drug_current_cost | drug_recipe_needed
---------+------------+-----------------+-------------------+--------------------
       1 | └ёяшЁшэ    |              30 |           32.2200 | f
       2 | =єЁрIхэ    |              40 |           55.1000 | f
       3 | └ёяюЁърь   |              20 |           69.8000 | t
       4 | =ю°яр      |              35 |           20.9000 | f
       5 | ¬шрчюышэ   |              50 |           11.0000 | f
       6 | ╠хчшь      |              30 |           41.2000 | f
       7 | ╨хушфЁюэ   |              70 |            5.4000 | f
       8 | └ыєюъёхЄшэ |              50 |           15.4000 | t
       9 | ╥єяЁрфшэ   |              30 |           25.0000 | f
(9 ёЄЁ№ъ)
```

Рис. 1.8: Созданные таблицы с данными

Однако, в миграции обычно добавляют только важнейшие данные, которые не будут изменяться. Для добавления обычных данных в таблицы рекомендуется использовать *manage commands*:

```python
from django.core.management.base import BaseCommand
from DatabaseORM.models import *
from datetime import datetime


class Command(BaseCommand):

    @staticmethod
    def create_disease():
        Disease(id_disease=1, disease_name='Ветрянка').save()
        Disease(id_disease=2, disease_name='Корь').save()
        Disease(id_disease=3, disease_name='ОРЗ').save()
        Disease(id_disease=4, disease_name='Язва').save()
        Disease(id_disease=5, disease_name='Импотенция').save()

    @staticmethod
    def create_drug():
        Drug(id_drug=1, drug_name='Аспирин', drug_shelf_life=30, drug_current_cost=32.22, drug_recipe_needed=False).save()
        Drug(id_drug=2, drug_name='Нурафен', drug_shelf_life=40, drug_current_cost=55.10, drug_recipe_needed=False).save()
        Drug(id_drug=3, drug_name='Аспоркам', drug_shelf_life=20, drug_current_cost=69.80, drug_recipe_needed=True).save()
        Drug(id_drug=4, drug_name='Ношпа', drug_shelf_life=35, drug_current_cost=20.90, drug_recipe_needed=False).save()
        Drug(id_drug=5, drug_name='Диазолин', drug_shelf_life=50, drug_current_cost=11.0, drug_recipe_needed=False).save()
        Drug(id_drug=6, drug_name='Мезим', drug_shelf_life=30, drug_current_cost=41.20, drug_recipe_needed=False).save()
        Drug(id_drug=7, drug_name='Регидрон', drug_shelf_life=70, drug_current_cost=5.40, drug_recipe_needed=False).save()
```

```
25          Drug(id_drug=8, drug_name='Флуоксетин', drug_shelf_life=50, drug_current_cost
    =15.40, drug_recipe_needed=True).save()
26          Drug(id_drug=9, drug_name='Супрадин', drug_shelf_life=30, drug_current_cost=25.0,
    drug_recipe_needed=False).save()
27
28      @staticmethod
29      def create_incompatibility():
30          Incompatibility(id_incompatibility=1, id_drug_first=Drug.objects.get(id_drug=1),
    id_drug_second=Drug.objects.get(id_drug=8)).save()
31          Incompatibility(id_incompatibility=2, id_drug_first=Drug.objects.get(id_drug=2),
    id_drug_second=Drug.objects.get(id_drug=3)).save()
32          Incompatibility(id_incompatibility=3, id_drug_first=Drug.objects.get(id_drug=1),
    id_drug_second=Drug.objects.get(id_drug=2)).save()
33          Incompatibility(id_incompatibility=4, id_drug_first=Drug.objects.get(id_drug=4),
    id_drug_second=Drug.objects.get(id_drug=3)).save()
34          Incompatibility(id_incompatibility=5, id_drug_first=Drug.objects.get(id_drug=6),
    id_drug_second=Drug.objects.get(id_drug=8)).save()
35          Incompatibility(id_incompatibility=6, id_drug_first=Drug.objects.get(id_drug=3),
    id_drug_second=Drug.objects.get(id_drug=7)).save()
36          Incompatibility(id_incompatibility=7, id_drug_first=Drug.objects.get(id_drug=9),
    id_drug_second=Drug.objects.get(id_drug=1)).save()
37          Incompatibility(id_incompatibility=8, id_drug_first=Drug.objects.get(id_drug=3),
    id_drug_second=Drug.objects.get(id_drug=4)).save()
38
39      @staticmethod
40      def create_indication():
41          Indication(id_indication=1, id_disease=Disease.objects.get(id_disease=1), id_drug
    =Drug.objects.get(id_drug=1)).save()
42          Indication(id_indication=2, id_disease=Disease.objects.get(id_disease=2), id_drug
    =Drug.objects.get(id_drug=2)).save()
43          Indication(id_indication=3, id_disease=Disease.objects.get(id_disease=3), id_drug
    =Drug.objects.get(id_drug=4)).save()
44          Indication(id_indication=4, id_disease=Disease.objects.get(id_disease=4), id_drug
    =Drug.objects.get(id_drug=4)).save()
45          Indication(id_indication=5, id_disease=Disease.objects.get(id_disease=5), id_drug
    =Drug.objects.get(id_drug=5)).save()
46          Indication(id_indication=6, id_disease=Disease.objects.get(id_disease=5), id_drug
    =Drug.objects.get(id_drug=7)).save()
47          Indication(id_indication=7, id_disease=Disease.objects.get(id_disease=4), id_drug
    =Drug.objects.get(id_drug=7)).save()
48          Indication(id_indication=8, id_disease=Disease.objects.get(id_disease=4), id_drug
    =Drug.objects.get(id_drug=9)).save()
49          Indication(id_indication=9, id_disease=Disease.objects.get(id_disease=3), id_drug
    =Drug.objects.get(id_drug=8)).save()
50          Indication(id_indication=10, id_disease=Disease.objects.get(id_disease=3),
    id_drug=Drug.objects.get(id_drug=5)).save()
51          Indication(id_indication=11, id_disease=Disease.objects.get(id_disease=2),
    id_drug=Drug.objects.get(id_drug=6)).save()
52          Indication(id_indication=12, id_disease=Disease.objects.get(id_disease=1),
    id_drug=Drug.objects.get(id_drug=5)).save()
53
54      @staticmethod
55      def create_contraindication():
56          Contraindication(id_contraindication=1, id_disease=Disease.objects.get(id_disease
    =4), id_drug=Drug.objects.get(id_drug=8)).save()
57          Contraindication(id_contraindication=2, id_disease=Disease.objects.get(id_disease
    =4), id_drug=Drug.objects.get(id_drug=3)).save()
58          Contraindication(id_contraindication=3, id_disease=Disease.objects.get(id_disease
    =1), id_drug=Drug.objects.get(id_drug=6)).save()
59          Contraindication(id_contraindication=4, id_disease=Disease.objects.get(id_disease
    =1), id_drug=Drug.objects.get(id_drug=2)).save()
60          Contraindication(id_contraindication=5, id_disease=Disease.objects.get(id_disease
    =3), id_drug=Drug.objects.get(id_drug=3)).save()
61          Contraindication(id_contraindication=6, id_disease=Disease.objects.get(id_disease
    =3), id_drug=Drug.objects.get(id_drug=7)).save()
62          Contraindication(id_contraindication=7, id_disease=Disease.objects.get(id_disease
```

```
    =3), id_drug=Drug.objects.get(id_drug=9)).save()
63          Contraindication(id_contraindication=8, id_disease=Disease.objects.get(id_disease
    =4), id_drug=Drug.objects.get(id_drug=2)).save()
64          Contraindication(id_contraindication=9, id_disease=Disease.objects.get(id_disease
    =5), id_drug=Drug.objects.get(id_drug=6)).save()
65          Contraindication(id_contraindication=10, id_disease=Disease.objects.get(
    id_disease=5), id_drug=Drug.objects.get(id_drug=1)).save()
66          Contraindication(id_contraindication=11, id_disease=Disease.objects.get(
    id_disease=2), id_drug=Drug.objects.get(id_drug=4)).save()
67          Contraindication(id_contraindication=12, id_disease=Disease.objects.get(
    id_disease=2), id_drug=Drug.objects.get(id_drug=5)).save()
68
69      @staticmethod
70      def create_provider():
71          Provider(id_provider=1, provider_name='Добрый доктор').save()
72          Provider(id_provider=2, provider_name='Доставщики9000').save()
73          Provider(id_provider=3, provider_name='Фармаштекер').save()
74          Provider(id_provider=4, provider_name='Компания Бориса').save()
75          Provider(id_provider=5, provider_name='Доктор силач').save()
76
77      @staticmethod
78      def create_drugstore():
79          Drugstore(id_drugstore=1, drugstore_address='Москва, ул. Ленина к211').save()
80          Drugstore(id_drugstore=2, drugstore_address='СанктПетербург-, пр. Энгельса к102').
    save()
81          Drugstore(id_drugstore=3, drugstore_address='СанктПетербург-, пр. Невский к51').save
    ()
82          Drugstore(id_drugstore=4, drugstore_address='СанктПетербург-, ул. Парашютная к121').
    save()
83
84      @staticmethod
85      def create_consignment():
86          Consignment(id_consignment=1, id_drug=Drug.objects.get(id_drug=5), id_provider=
    Provider.objects.get(id_provider=3),
87                      id_drugstore=Drugstore.objects.get(id_drugstore=1),
88                      consignment_drug_count=1200, consignment_arrival_date=datetime(2011,
    1, 10), consignment_manufacture_date=datetime(2010, 9, 25)).save()
89          Consignment(id_consignment=2, id_drug=Drug.objects.get(id_drug=2), id_provider=
    Provider.objects.get(id_provider=5),
90                      id_drugstore=Drugstore.objects.get(id_drugstore=1),
91                      consignment_drug_count=1000, consignment_arrival_date=datetime(2011,
    2, 13), consignment_manufacture_date=datetime(2011, 2, 11)).save()
92          Consignment(id_consignment=3, id_drug=Drug.objects.get(id_drug=7), id_provider=
    Provider.objects.get(id_provider=4),
93                      id_drugstore=Drugstore.objects.get(id_drugstore=4),
94                      consignment_drug_count=3000, consignment_arrival_date=datetime(2011,
    1, 14), consignment_manufacture_date=datetime(2011, 1, 10)).save()
95          Consignment(id_consignment=4, id_drug=Drug.objects.get(id_drug=1), id_provider=
    Provider.objects.get(id_provider=2),
96                      id_drugstore=Drugstore.objects.get(id_drugstore=3),
97                      consignment_drug_count=3000, consignment_arrival_date=datetime(2011,
    2, 15), consignment_manufacture_date=datetime(2011, 2, 1)).save()
98          Consignment(id_consignment=5, id_drug=Drug.objects.get(id_drug=3), id_provider=
    Provider.objects.get(id_provider=1),
99                      id_drugstore=Drugstore.objects.get(id_drugstore=3),
100                     consignment_drug_count=6000, consignment_arrival_date=datetime(2011,
    1, 10), consignment_manufacture_date=datetime(2010, 11, 11)).save()
101         Consignment(id_consignment=6, id_drug=Drug.objects.get(id_drug=4), id_provider=
    Provider.objects.get(id_provider=1),
102                     id_drugstore=Drugstore.objects.get(id_drugstore=2),
103                     consignment_drug_count=2000, consignment_arrival_date=datetime(2011,
    1, 20), consignment_manufacture_date=datetime(2010, 12, 20)).save()
104         Consignment(id_consignment=7, id_drug=Drug.objects.get(id_drug=5), id_provider=
    Provider.objects.get(id_provider=5),
105                     id_drugstore=Drugstore.objects.get(id_drugstore=2),
106                     consignment_drug_count=1000, consignment_arrival_date=datetime(2011,
```

```
    1, 24), consignment_manufacture_date=datetime(2011, 1, 21)).save()
107         Consignment(id_consignment=8, id_drug=Drug.objects.get(id_drug=6), id_provider=
    Provider.objects.get(id_provider=3),
108                     id_drugstore=Drugstore.objects.get(id_drugstore=4),
109                     consignment_drug_count=1000, consignment_arrival_date=datetime(2011,
    2, 10), consignment_manufacture_date=datetime(2011, 2, 9)).save()
110
111     @staticmethod
112     def create_client():
113         Client(id_client=1, client_name='Валера').save()
114         Client(id_client=2, client_name='Игнат').save()
115         Client(id_client=3, client_name='Борис').save()
116
117     @staticmethod
118     def create_request_part():
119         RequestPart(id_request_part=1, id_client=Client.objects.get(id_client=1),
    request_part_date=datetime(2011, 3, 17)).save()
120         RequestPart(id_request_part=2, id_client=Client.objects.get(id_client=3),
    request_part_date=datetime(2011, 4, 2)).save()
121         RequestPart(id_request_part=3, id_client=Client.objects.get(id_client=2),
    request_part_date=datetime(2011, 3, 11)).save()
122         RequestPart(id_request_part=4, id_client=Client.objects.get(id_client=1),
    request_part_date=datetime(2011, 4, 22)).save()
123         RequestPart(id_request_part=5, id_client=Client.objects.get(id_client=1),
    request_part_date=datetime(2011, 4, 22)).save()
124         RequestPart(id_request_part=6, id_client=Client.objects.get(id_client=2),
    request_part_date=datetime(2011, 4, 28)).save()
125         RequestPart(id_request_part=7, id_client=Client.objects.get(id_client=3),
    request_part_date=datetime(2011, 4, 10)).save()
126         RequestPart(id_request_part=8, id_client=Client.objects.get(id_client=3),
    request_part_date=datetime(2011, 3, 2)).save()
127
128     @staticmethod
129     def create_request():
130         Request(id_request=1, id_request_part=RequestPart.objects.get(id_request_part=2),
     id_consignment=Consignment.objects.get(id_consignment=1),
131                 request_drug_previous_cost=15.0, request_count=100).save()
132         Request(id_request=2, id_request_part=RequestPart.objects.get(id_request_part=3),
     id_consignment=Consignment.objects.get(id_consignment=2),
133                 request_drug_previous_cost=25.0, request_count=50).save()
134         Request(id_request=3, id_request_part=RequestPart.objects.get(id_request_part=4),
     id_consignment=Consignment.objects.get(id_consignment=3),
135                 request_drug_previous_cost=70.0, request_count=20).save()
136         Request(id_request=4, id_request_part=RequestPart.objects.get(id_request_part=1),
     id_consignment=Consignment.objects.get(id_consignment=4),
137                 request_drug_previous_cost=15.0, request_count=200).save()
138         Request(id_request=5, id_request_part=RequestPart.objects.get(id_request_part=7),
     id_consignment=Consignment.objects.get(id_consignment=5),
139                 request_drug_previous_cost=35.0, request_count=100).save()
140         Request(id_request=6, id_request_part=RequestPart.objects.get(id_request_part=8),
     id_consignment=Consignment.objects.get(id_consignment=6),
141                 request_drug_previous_cost=10.0, request_count=400).save()
142         Request(id_request=7, id_request_part=RequestPart.objects.get(id_request_part=2),
     id_consignment=Consignment.objects.get(id_consignment=7),
143                 request_drug_previous_cost=10.0, request_count=100).save()
144         Request(id_request=8, id_request_part=RequestPart.objects.get(id_request_part=3),
     id_consignment=Consignment.objects.get(id_consignment=8),
145                 request_drug_previous_cost=15.0, request_count=50).save()
146         Request(id_request=9, id_request_part=RequestPart.objects.get(id_request_part=2),
     id_consignment=Consignment.objects.get(id_consignment=1),
147                 request_drug_previous_cost=15.0, request_count=50).save()
148         Request(id_request=10, id_request_part=RequestPart.objects.get(id_request_part=1)
    , id_consignment=Consignment.objects.get(id_consignment=2),
149                 request_drug_previous_cost=25.0, request_count=150).save()
150         Request(id_request=11, id_request_part=RequestPart.objects.get(id_request_part=1)
    , id_consignment=Consignment.objects.get(id_consignment=3),
```

```
151                 request_drug_previous_cost=70.0, request_count=120).save()
152         Request(id_request=12, id_request_part=RequestPart.objects.get(id_request_part=7)
    , id_consignment=Consignment.objects.get(id_consignment=4),
153                 request_drug_previous_cost=15.0, request_count=250).save()
154         Request(id_request=13, id_request_part=RequestPart.objects.get(id_request_part=7)
    , id_consignment=Consignment.objects.get(id_consignment=5),
155                 request_drug_previous_cost=35.0, request_count=130).save()
156         Request(id_request=14, id_request_part=RequestPart.objects.get(id_request_part=6)
    , id_consignment=Consignment.objects.get(id_consignment=6),
157                 request_drug_previous_cost=10.0, request_count=100).save()
158         Request(id_request=15, id_request_part=RequestPart.objects.get(id_request_part=3)
    , id_consignment=Consignment.objects.get(id_consignment=7),
159                 request_drug_previous_cost=10.0, request_count=150).save()
160         Request(id_request=16, id_request_part=RequestPart.objects.get(id_request_part=8)
    , id_consignment=Consignment.objects.get(id_consignment=8),
161                 request_drug_previous_cost=15.0, request_count=250).save()
162
163     def handle(self, *args, **options):
164         self.create_disease()
165         self.create_drug()
166         self.create_incompatibility()
167         self.create_indication()
168         self.create_contraindication()
169         self.create_provider()
170         self.create_drugstore()
171         self.create_consignment()
172         self.create_client()
173         self.create_request_part()
174         self.create_request()
```

Теперь можно миграцией создать таблицы, после чего командой добавить данные:

```
manage.py@Server > migrate DatabaseORM 1_create_tables
Operations to perform:
  Target specific migration: 1_create_tables, from DatabaseORM
Running migrations:
  Applying DatabaseORM.1_create_tables... OK

Process finished with exit code 0
manage.py@Server > populate
Process finished with exit code 0
```

Рис. 1.9: Создание таблиц и добавление данных

1.5 Вывод

В результате данной работы было проведено знакомство с миграциями моделей используя Django, а также с manage-командами для наполнения базы данных. К достоинствам миграции Django можно отнести:

- Ускорение процесса изменения схемы базы данных.
- Возможность отслеживания схемы базы данных.
- Поддержка многими бэкендами (PostgreSQL, MySQL, SQLite).
- Возможность отката.

Также был проведен опыт использования manage-команд. Использование данного инструмента позволяет расширить проект, написанием каких-либо собственных команд. Так например, можно написать генератор для заполнения полей таблиц в базе данных.