Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



**Отчет**

#### По лабораторной работе №5

#### По курсу «Разработка интернет приложений»

Вариант 19

**ИСПОЛНИТЕЛЬ:**

Попов Илья

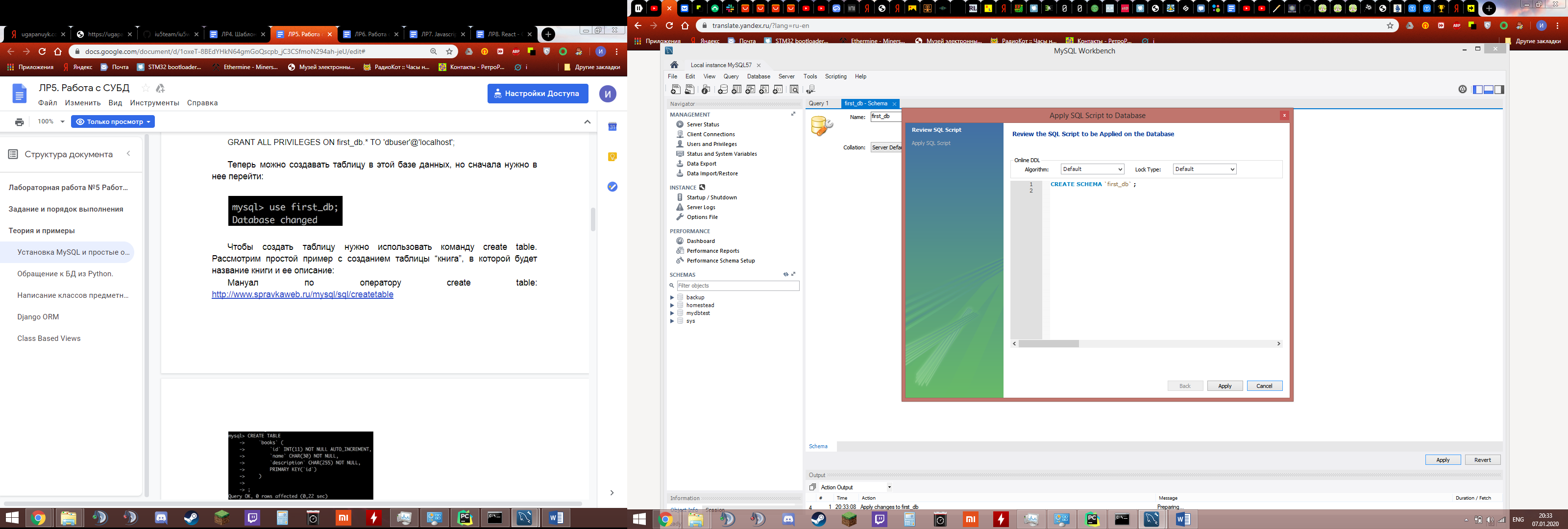
Группа ИУ5-51Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

Москва 2020

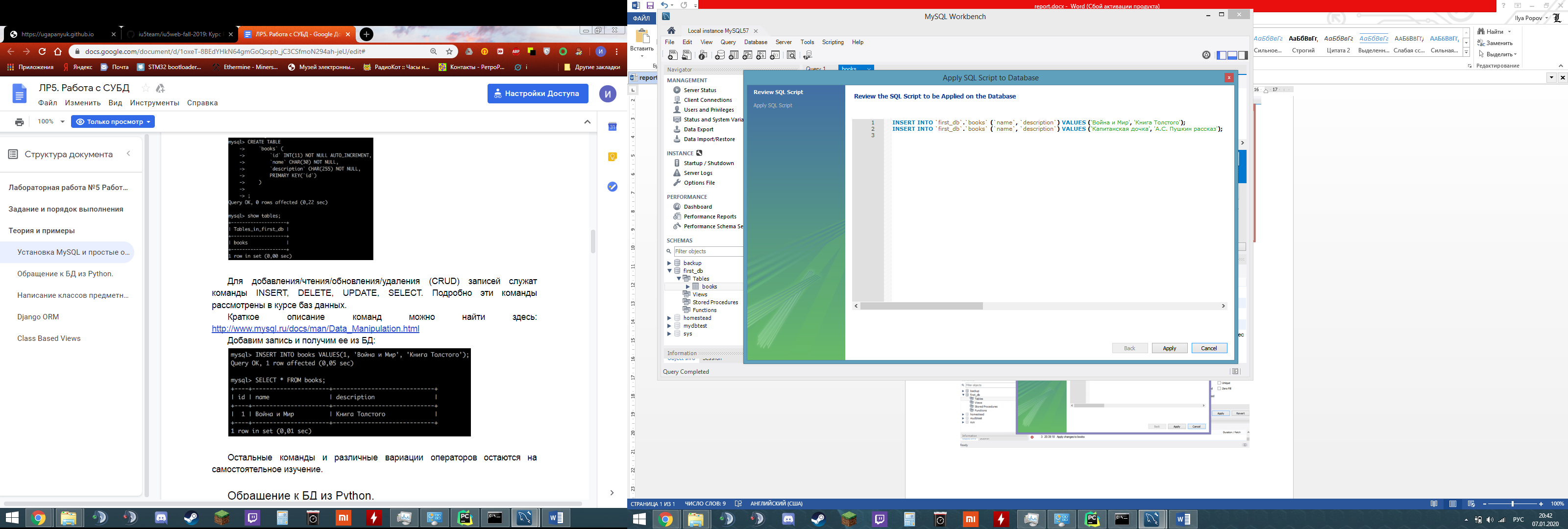
Создание базы данных через MySQL Workbench



Создание таблицы “**books**”

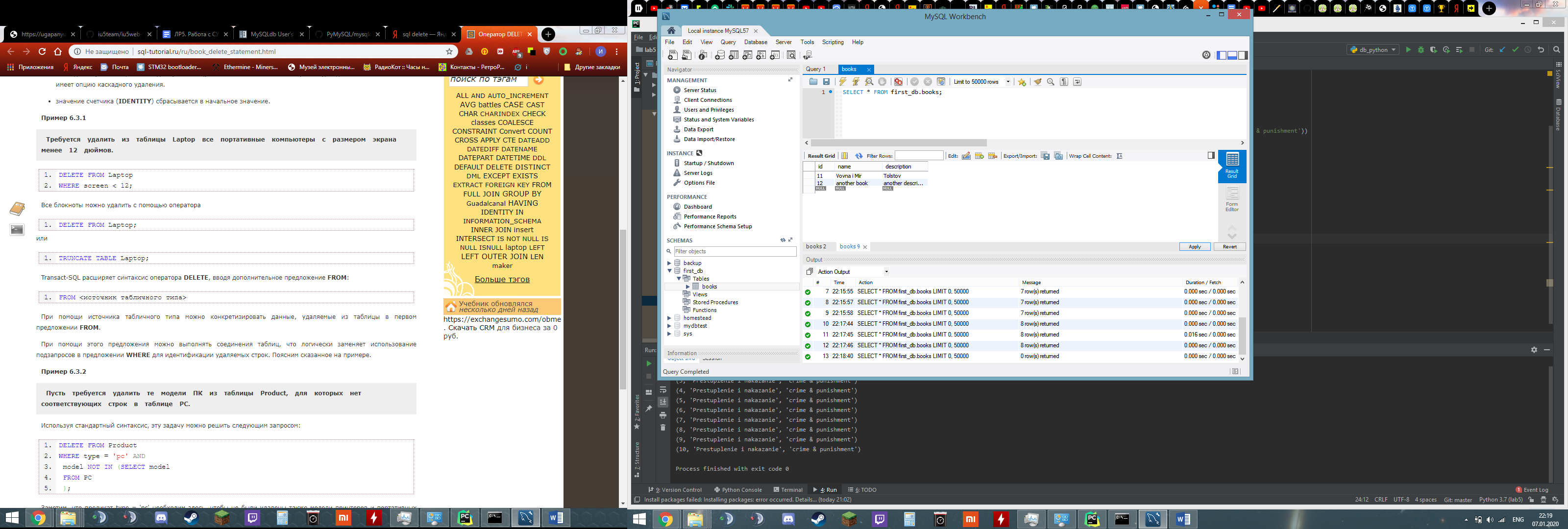


Добавление записей к таблице “**books**”

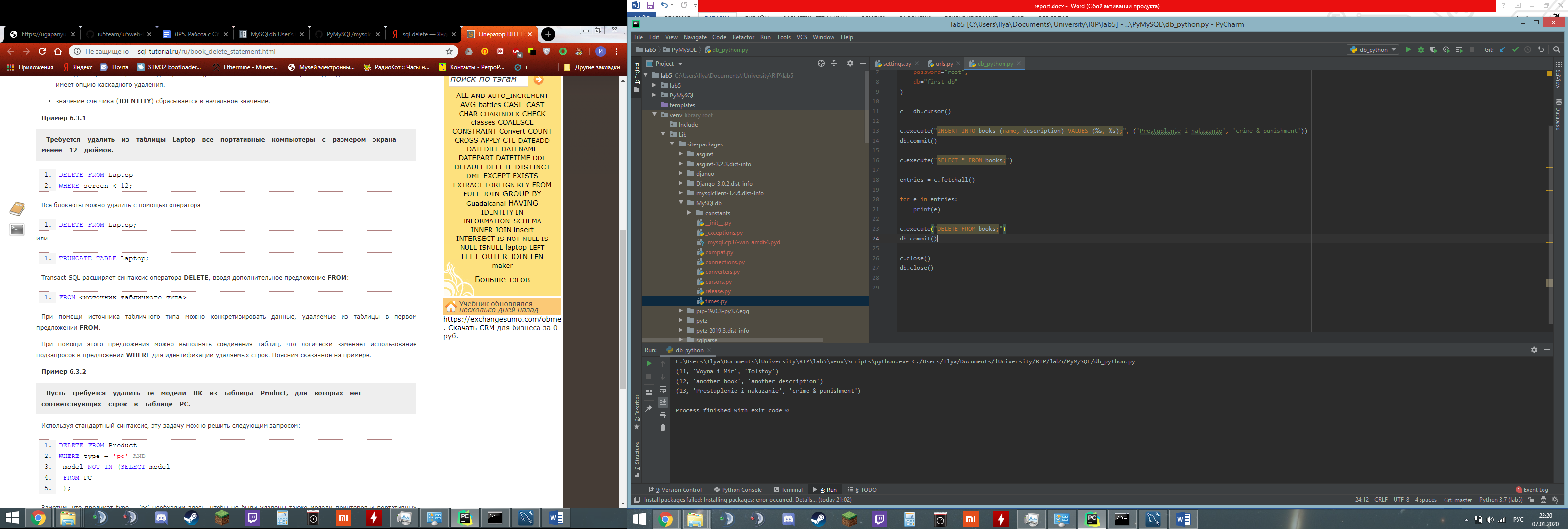


Обращение к БД из Python

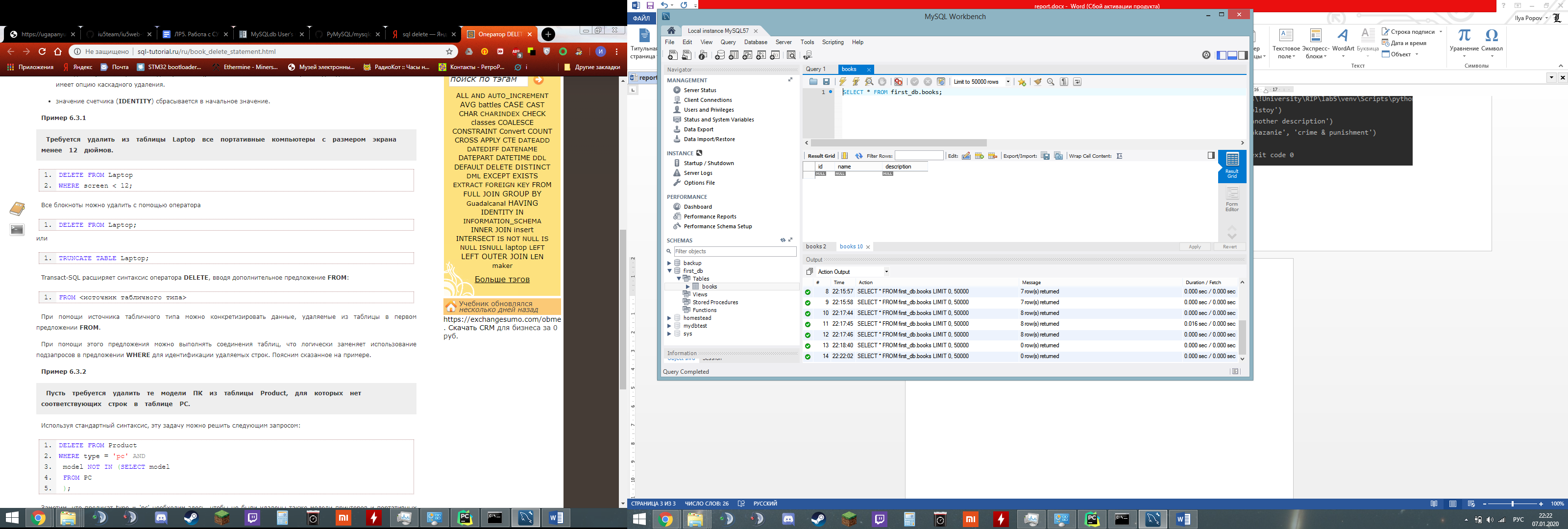
Состояние таблицы до применения скрипта



Вывод скрипта



Состояние таблицы после выполнения скрипта (все записи удалены)



Код скрипта

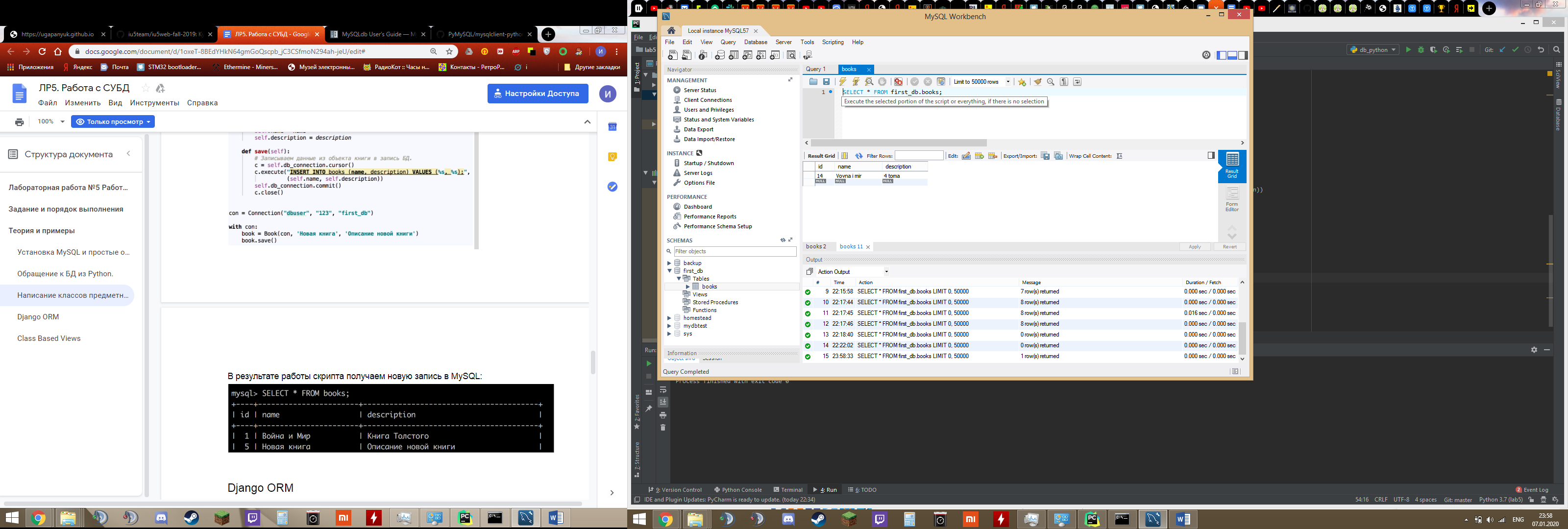
import MySQLdb  
  
  
db = MySQLdb.connect(  
 host="localhost",  
 user="root",  
 password="root",  
 db="first\_db"  
)  
  
c = db.cursor()  
c.execute("INSERT INTO books (name, description) VALUES (%s, %s);", ('Prestuplenie i nakazanie', 'crime & punishment'))  
db.commit()  
c.execute("SELECT \* FROM books;")  
entries = c.fetchall()  
for e in entries:  
 print(e)  
c.execute("DELETE FROM books;")  
db.commit()  
c.close()  
db.close()

Написание классов предметной области с соединением с БД

Код скрипта

import MySQLdb  
  
  
class Connection:  
 def \_\_init\_\_(self, user,password, db, host='localhost'):  
 self.host = host  
 self.user = user  
 self.password = password  
 self.db = db  
 self.\_connection = None  
  
 @property  
 def connection(self):  
 return self.\_connection  
  
 def \_\_enter\_\_(self):  
 self.connect()  
  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 self.disconnect()  
  
 def connect(self):  
 if not self.\_connection:  
 self.\_connection = MySQLdb.connect(  
 host=self.host,  
 user=self.user,  
 passwd=self.password,  
 db=self.db  
 )  
  
 def disconnect(self):  
 if self.\_connection:  
 self.\_connection.close()  
  
  
class Book:  
  
 def \_\_init\_\_(self, db\_connection, name, description):  
 self.db\_connection = db\_connection.connection  
 self.name = name  
 self.description = description  
  
 def save(self):  
 c = self.db\_connection.cursor()  
 c.execute("INSERT INTO books (name, description) VALUES (%s, %s);", (self.name, self.description))  
 self.db\_connection.commit()  
 c.close()  
  
  
con = Connection("root", "root", "first\_db")  
  
with con:  
 book = Book(con,'Voyna i mir', '4 toma')  
 book.save()

БД после выполнения скрипта

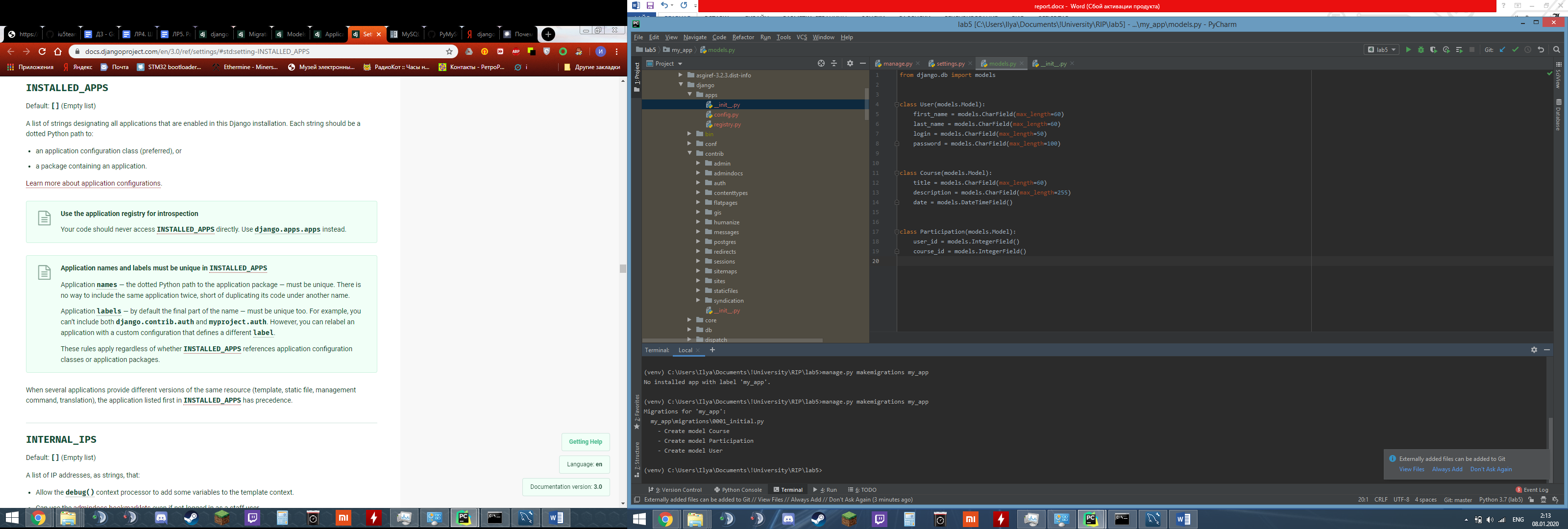


Django ORM

Создание моделей **models.py:**

from django.db import models  
  
  
class User(models.Model):  
 first\_name = models.CharField(max\_length=60)  
 last\_name = models.CharField(max\_length=60)  
 login = models.CharField(max\_length=50)  
 password = models.CharField(max\_length=100)  
  
  
class Course(models.Model):  
 title = models.CharField(max\_length=60)  
 description = models.CharField(max\_length=255)  
 date = models.DateTimeField()  
  
  
class Participation(models.Model):  
 user\_id = models.IntegerField()  
 course\_id = models.IntegerField()

Выполнение команды **manage.py makemigrations my\_app**



Выполнение команды **manage.py migrate:**

(venv) C:\Users\Ilya\Documents\!University\RIP\lab5>manage.py migrate

System check identified some issues:

WARNINGS:

?: (mysql.W002) MySQL Strict Mode is not set for database connection 'default'

HINT: MySQL's Strict Mode fixes many data integrity problems in MySQL, such as data truncation upon insertion, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/en/3.0/ref/databases/#mysql-

sql-mode

Operations to perform:

Apply all migrations: admin, auth, contenttypes, my\_app, 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

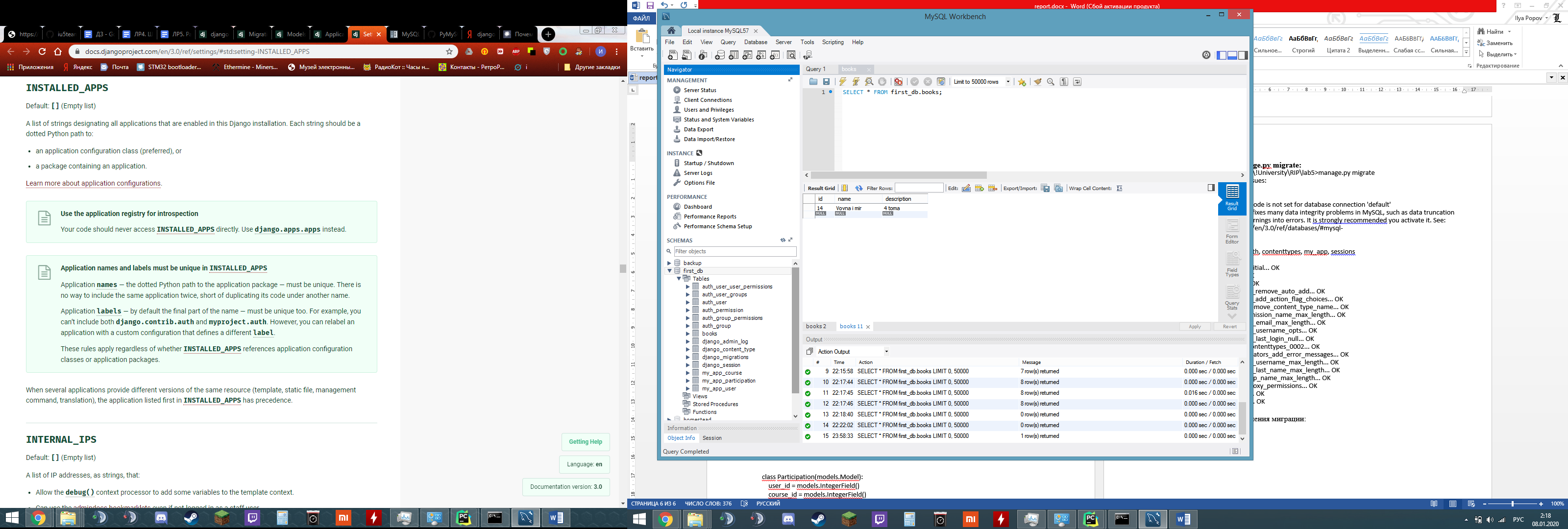
Applying auth.0010\_alter\_group\_name\_max\_length... OK

Applying auth.0011\_update\_proxy\_permissions... OK

Applying my\_app.0001\_initial... OK

Applying sessions.0001\_initial... OK

Состояние БД после выполнения миграции:



Список объектов предметной области на отдельной страничке, реализованный с помощью **Class Based View**

