# 4. Реализация

## 4.1 Тестирование

### 4.1.1 Модульные тесты

### Файл test.py, который содержит тесты, создается в Django в ходе создания проекта. В данном файле можно создавать unit- и doc- тесты. В данной курсовой работе были выбраны unit-тесты для тестирования атрибутов и методов, которые их изменяют. Проще говоря unit-тесты служат для непосредственной проверки заполненных полей, а не поэтапной работы программы.

Для создания unit-теста используется класс TestCase, от которого наследуются все , созданные разработчиком, классы с тестами.

### Проверка метода резервирования стола

def test\_dinnerwagon\_is\_reserved(self):  
 city\_test = City(name=**'Tomsk'**)  
 est\_test = Establishment(name=**'Vaflia project'**, city=city\_test, email=**'kaka@mail.ru'**)  
 estbranch = EstablishmentBranch(establishment=est\_test, address=**"City Tomsk, Vershinina str, 39a"**,  
 order\_phone\_number=**"99224343"**, help\_phone\_number=**"4324356"**)  
 esthall = BranchHall(type=1, branch=estbranch, )  
 dinerwagon\_test = DinnerWagon(hall=esthall, seats=20, is\_reserved=**False**)  
 dinerwagon\_test.reserve()  
 self.assertTrue(dinerwagon\_test.is\_reserved)

Данный метод проверяет идентичность поля is\_reserved. Если метод резервирования стола работает, то is\_reserved будет возвращать «True»

* 1. Проверка поля с названием блюда

**def** test\_dish\_class\_name(self):  
 dish = Dish(  
 name=**'Гречка'**,  
 price=70.50,  
 category=Dish.DISH\_TYPE\_GARNISH,  
 )  
 self.assertEqual(dish.\_\_str\_\_(), **'Гречка'**, **"Имя класса неверно"**)

В данном методе, удостоверяемся, что в ходе заполнения базы данных, поле name объекта Dish было передано верно.

3) Проверка поля с отменой заказа и проверкой его состояния

**def** test\_order\_decline(self):  
 order\_test = Order(client\_phone=8432424,  
 type=Order.TYPE\_DINNER\_WAGON,  
 state=Order.STATE\_DONE,  
 order\_date=date(2014, 12, 12),  
 execute\_datetime=datetime(2014, 12, 12, 18),  
 dinner\_wagon=DinnerWagon(is\_reserved=1, seats=2))  
 order\_test.decline()  
 self.assertEqual(order\_test.state, Order.STATE\_CANCELED)  
 self.assertEqual(order\_test.dinner\_wagon.is\_reserved, **False**)

Этот тест проверяет поле State у объекта Order и также проверяет занятость столика и сравнивает их с контрольными данными.

### Интеграционные тесты

### Интеграционные тесты в Django создаются с помощью класса WebTest, который является наследником TestCase, который нужен для модульного тестирования. Основное отличие TestCase и WebTest это переменная self.testapp из DjangoTestApp. Она позволяет получить доступ к API WebTest.

Важно понимать, что интеграционные тесты — это не замена юнит-тестам, а только дополнение к ним, и что 100% покрытие никак не гарантирует отсутствия ошибок. Юнит-тесты — точные, они говорят, что именно поломалось, они крайне полезны при рефакторинге и в сложных местах проекта.   
Чтобы показать различие: в unit-тесте для формы регистрации мы бы создали объект класса EmailRegistrationForm, передавали бы в него разные словари с данными и смотрели бы, какие вызываются исключения. Юнит-тесты максимально приближены к листингу, тестируют отдельный его метод или атрибут, и позволяют проверить, что все части системы по отдельности работаю правильно. Интеграционные тесты помогают проверять, что и вместе они работают тоже правильно[1].

### 4.1.3 Построение и выполнение тестов

Выполнение тестов осуществляется с помощью утилиты manage.py test. По умолчанию эта утилита проверит все тесты каждого приложения. Чтобы обратится к какому-то определенному приложению и проверить только его тесты, следует использовать python manage.py test <application\_name>.

Самое интересное в тестах Django - это то, что они не используют настоящую базу данных. Для тестов Django создает в оперативной памяти базу данных специально ля тестов. И поля в ней заполняются теми данными, которые были использованный в тестах

Запустим тесты в приложении wanna\_eat

C:\Kursovoy>python manage.py test

Creating test database for alias 'default'...

..............

----------------------------------------------------------------------

Ran 14 tests in 0.032s

OK

Destroying test database for alias 'default'..Затраченное время: 00:00:03.75

Из запуска тестов видно, что никаких ошибок или несовпадений не было выявлено.

**4.1.4 Запуск приложения**

Запуск приложения осуществляется на сервере разработки. Этот сервер служит непосредственно разработки приложения. Ему не нужно много ресурсов, в отличие от Apache Tomcat, например.

Стандартно сервер запускается на 127.0.0.1:8000. Это может вызвать небольшие проблемы, так как, например, Skype, приложение для видео коммуникации, работает именно на 8000 порту. В целях разрешения возможных конфликтов следует перенести сервер, например, на 127.0.0.1:8080. Для этого используем команду python manage runserver 8080. Приложение будет расположено именно на этом порту.

[1] http://habrahabr.ru/post/91471/-