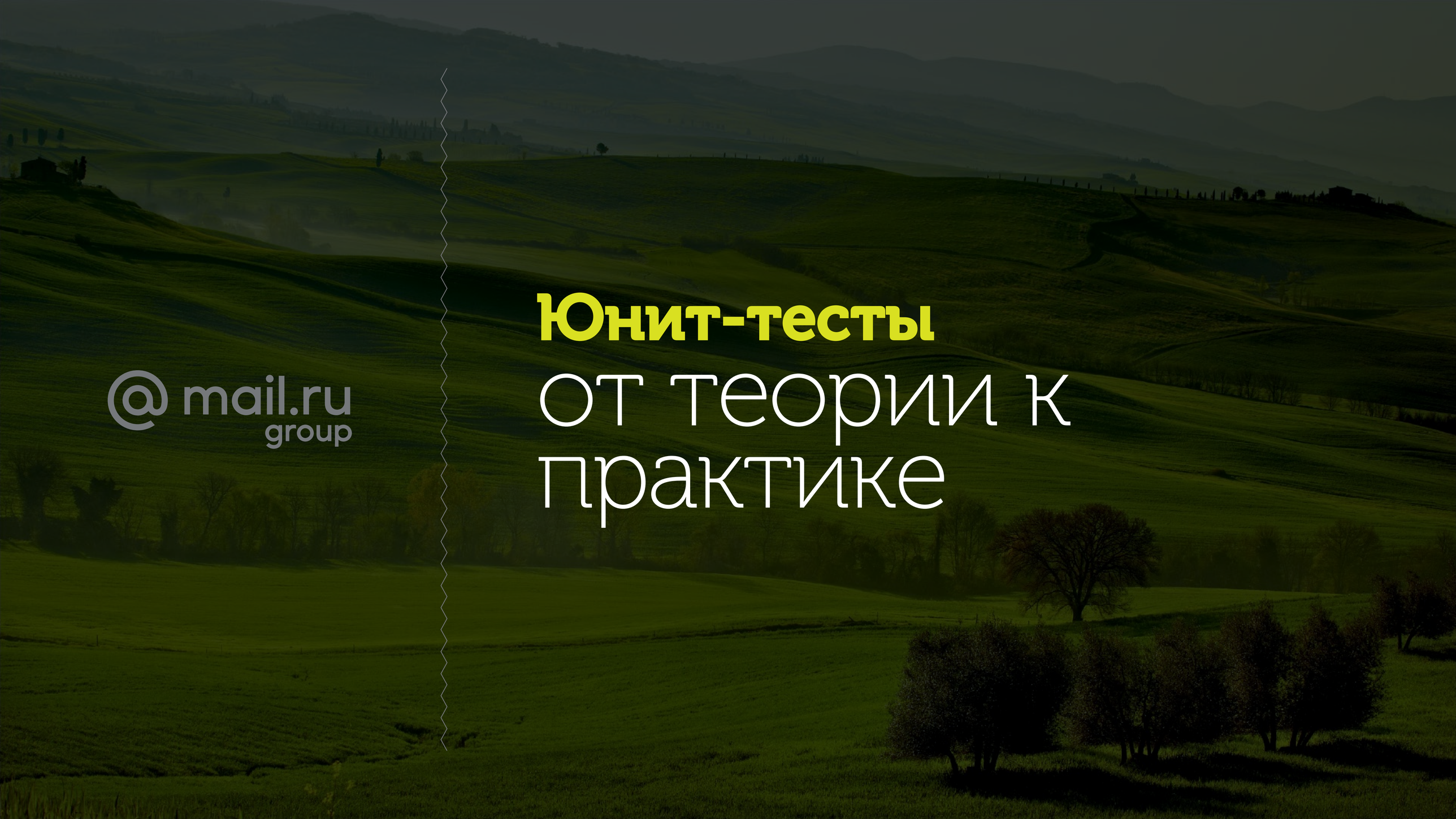
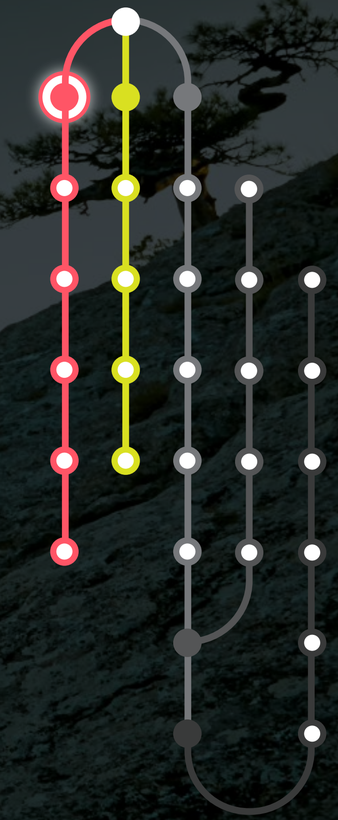
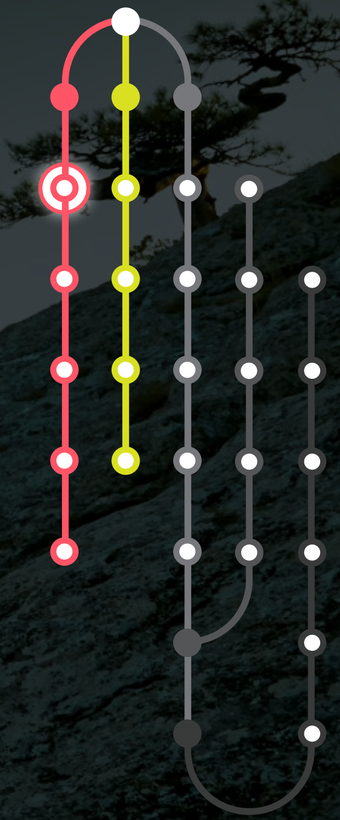

@ mail.ru
group
Юнит-тесты
от теории к
практике


Что мы хотим
Принципы
Реализация

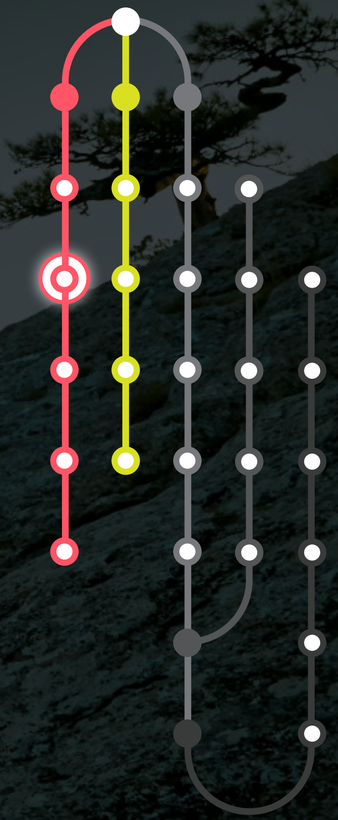
Что мы хотим



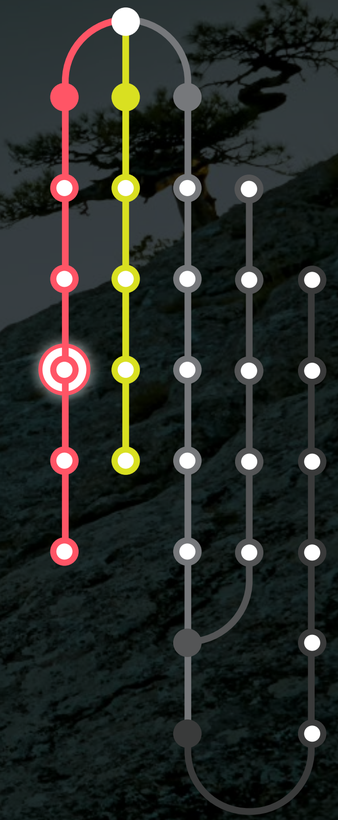
Регрессия



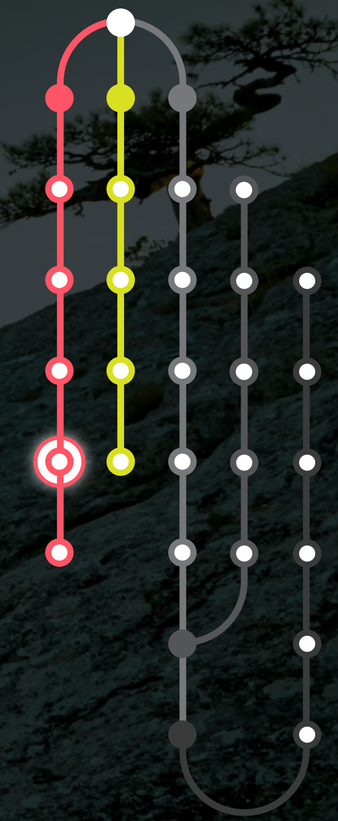
Влияние на архитектуру



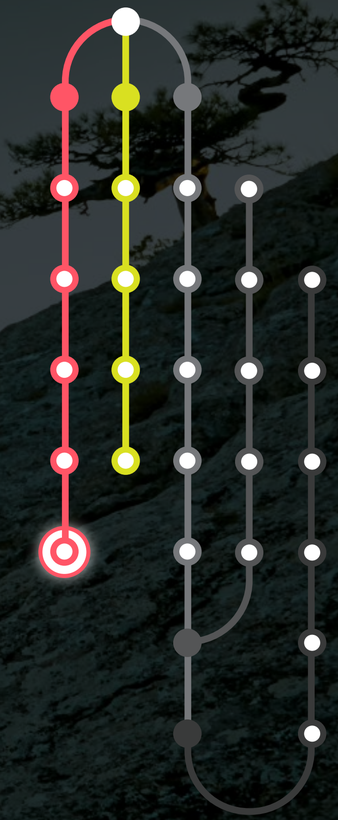
Понимание



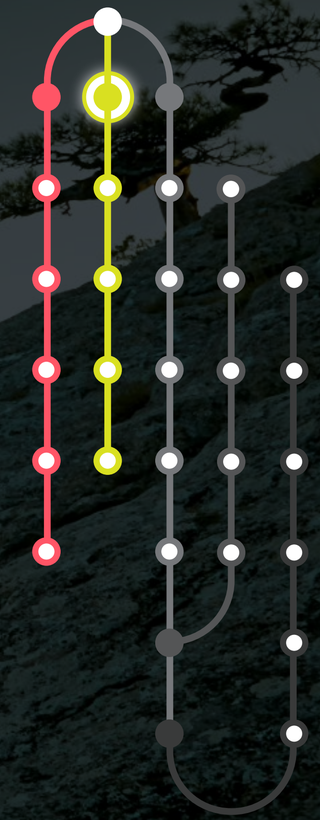
Отладка



Комфорт

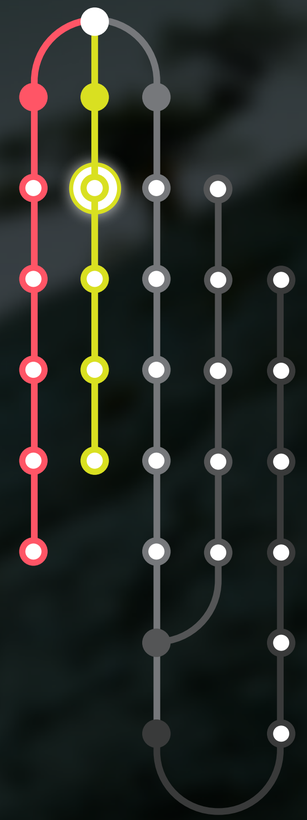


Принципы



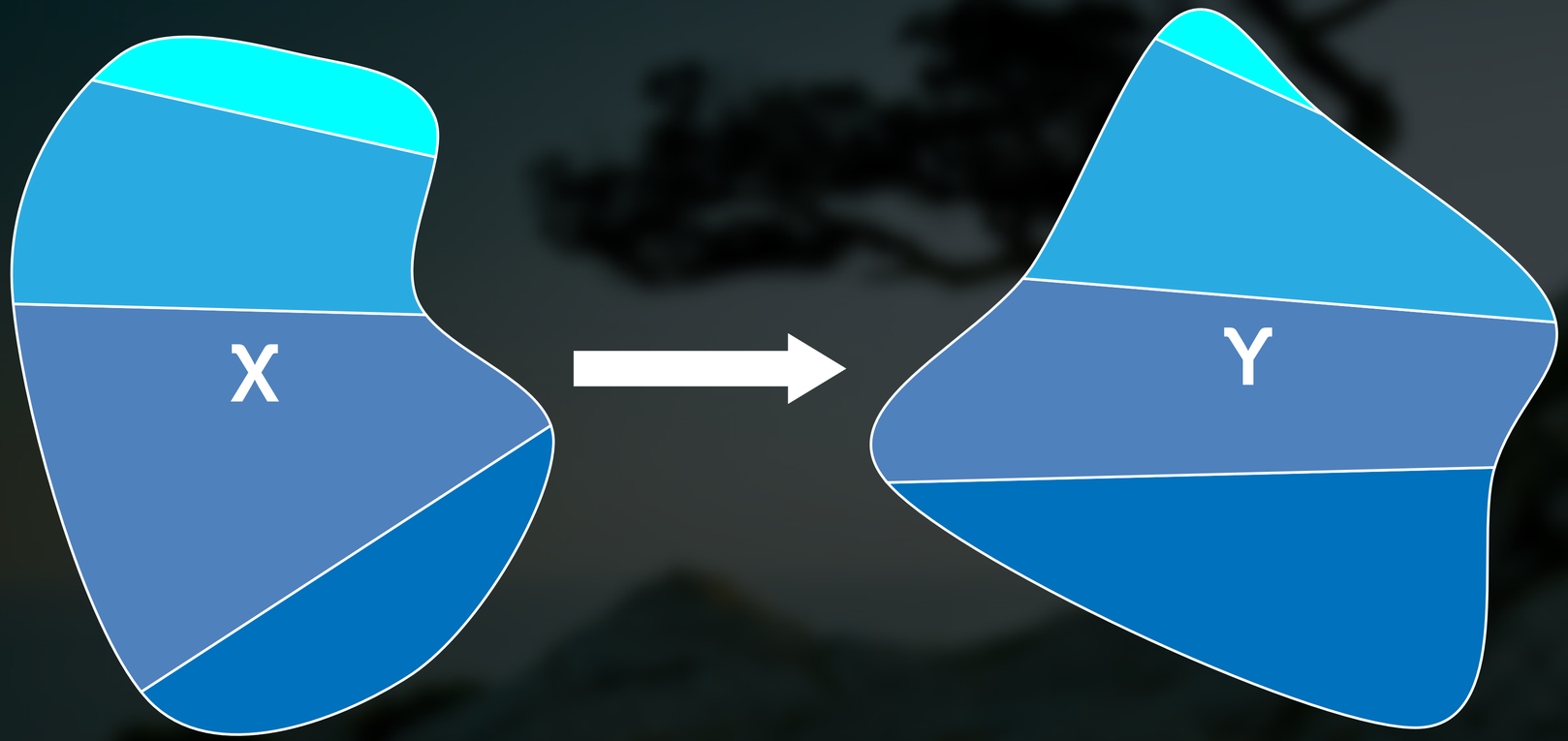
Test the interface, not the implementation?

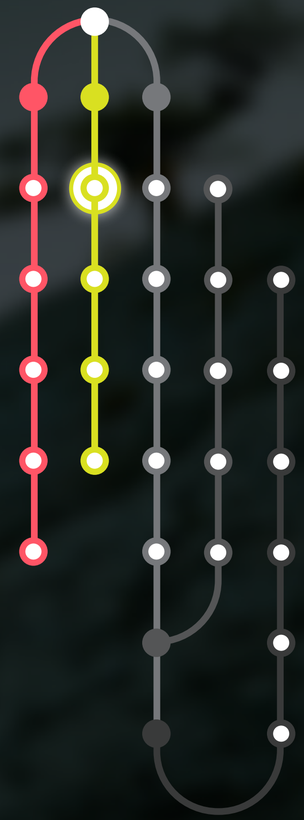
- f: X → Y



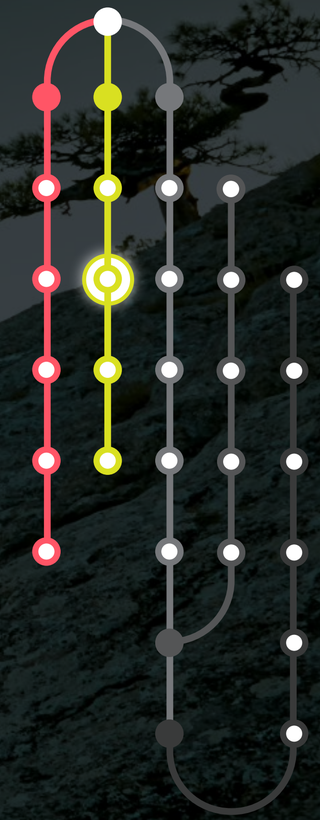
Test the interface, not the implementation?

- f: X → Y

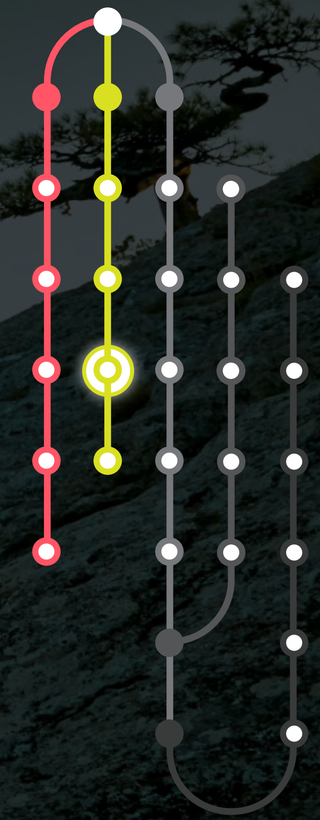


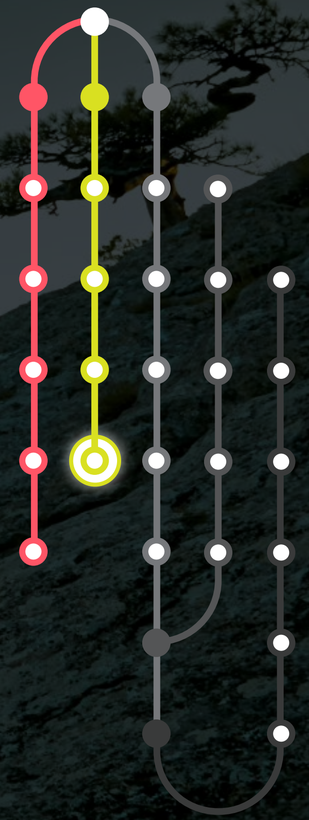


TDD?
Test first?



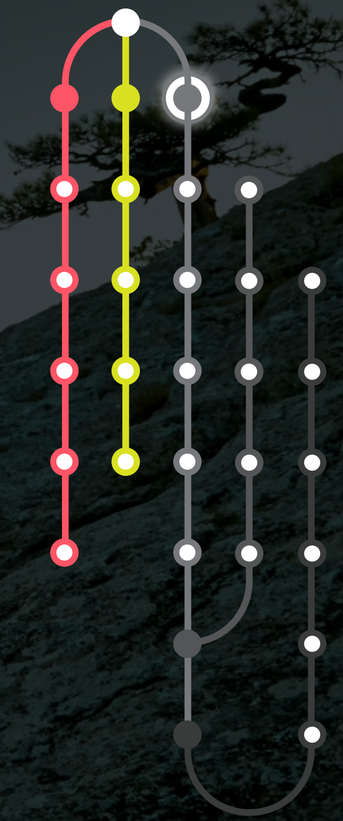
Unit testing ≠ testing





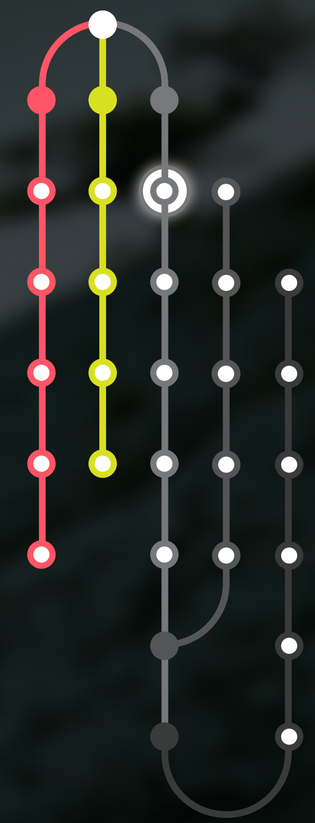
фото|@tudamru

Реализация

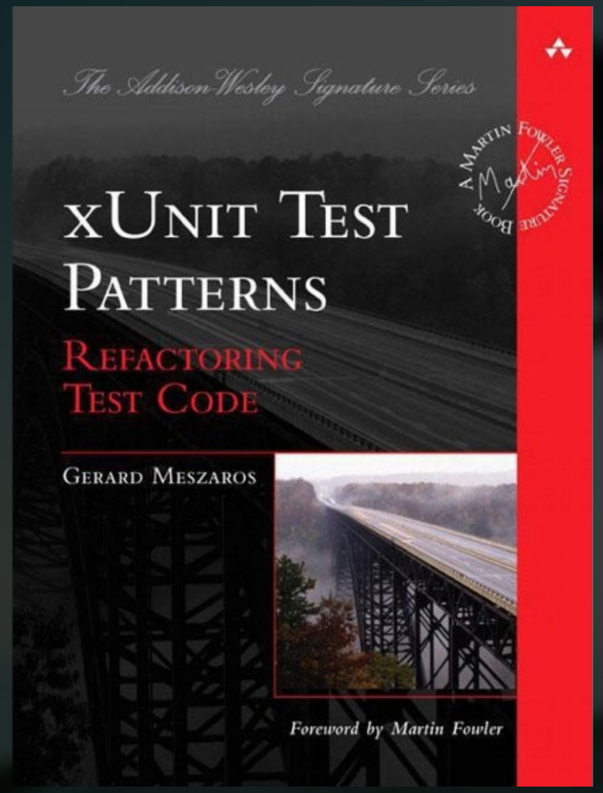


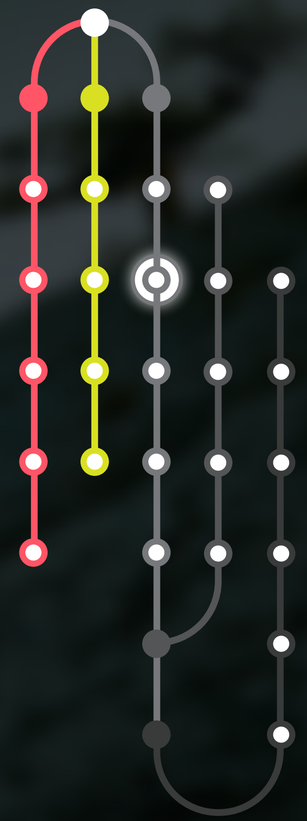
xUnit

- SUnit
- JUnit
- CppUnit
- RUnit → RSpec
- unittest
- Test::Class



Meszaros, Gerard (2007)
xUnit Test Patterns





Именование

- Verification → VerificationTestCase
- def verify
  - def test_verify__success
  - def test_verify__error
- def _filter_data
  - def test__filter_data

Custom asserts

```
def assertDirExists(self, dir_path):
    self.assertTrue(os.path.isdir(dir_path))

def assertNotFileExists(self, file_path):
    self.assertFalse(os.path.exists(file_path))

def assertFileContent(self, content, path):
    with open(path, 'rb') as fh:
        self.assertEqual(content, fh.read())
```

“test”

verify

validate

assert

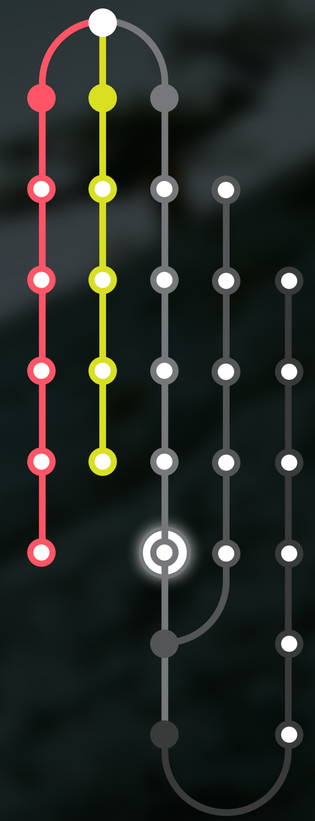
check

prove

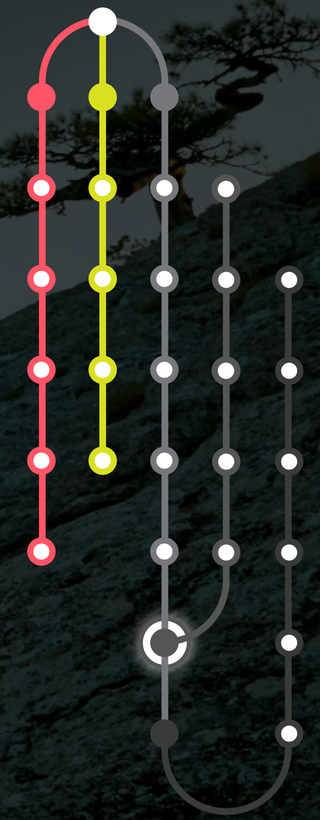
proof

ensure

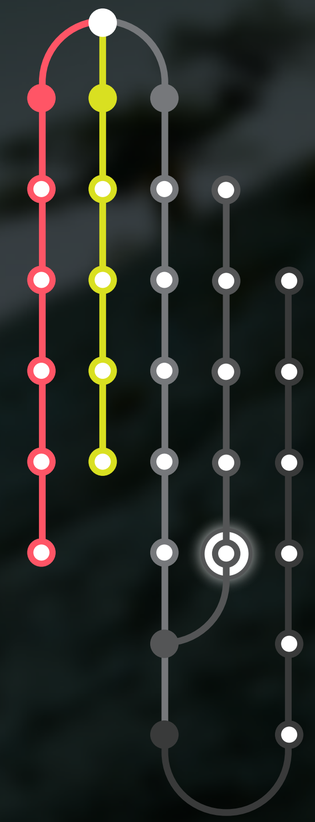
exam



Внешние зависимости



- Память
- Файловая система
- База данных
- ...



Dependency injection


```
def create(dt=None):
    if dt is None:
        dt = datetime.now()
```

Dependency injection

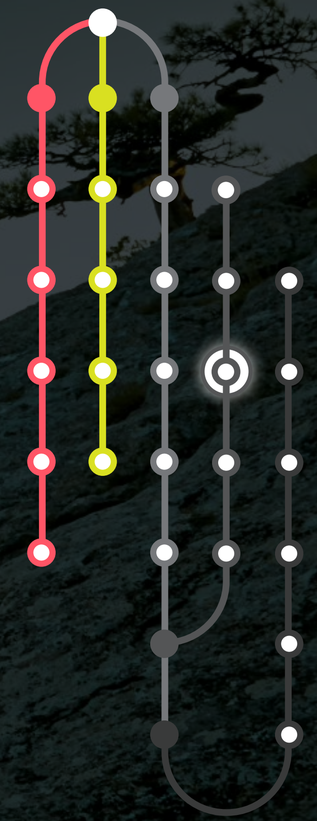
```
def create(dt=None):
    if dt is None:
        dt = datetime.now()

def download(requests_lib=None):
    if requests_lib is None:
        requests_lib = requests
```

Dependency injection

```
def create(dt=None):
    if dt is None:
        dt = datetime.now()

def download(requests_lib=None):
    if requests_lib is None:
        requests_lib = requests

class Downloader:
    def __init__(self, config, logger):
        self._config = config
        self._logger = logger
```

Monkey patching

```
@patch('uuid.uuid4')
def test__post(self, patched_uuid):
    patched_uuid.return_value = 'ID'
    self.assertEqual('ID2', uuid.uuid4())
```

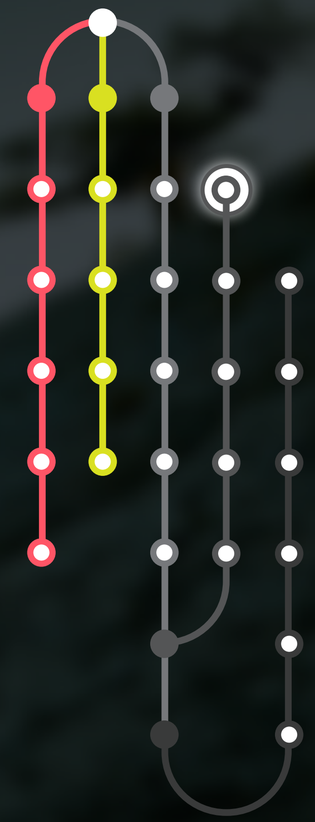


Test doubles

- Dummy
- Stub
- Spy
- Mock

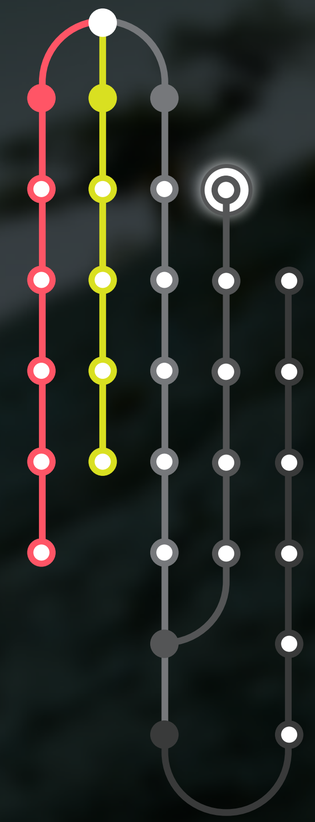
База данных

- Поддельная база



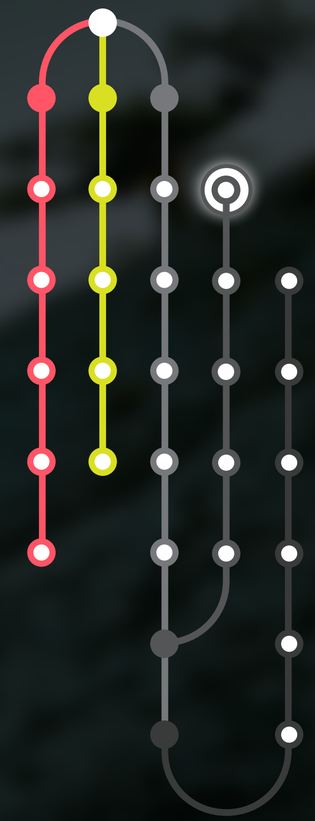
База данных

- Поддельная база
- SQLite

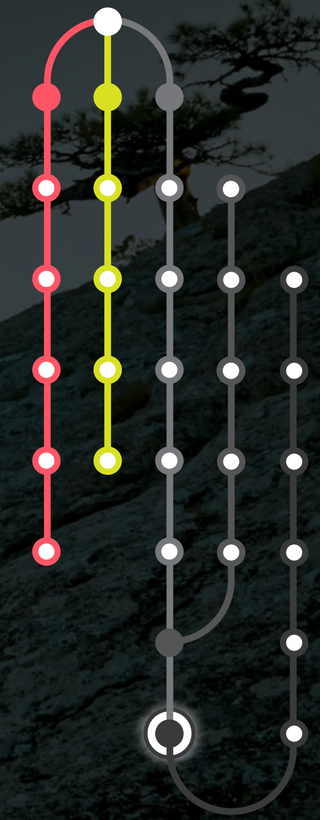


База данных

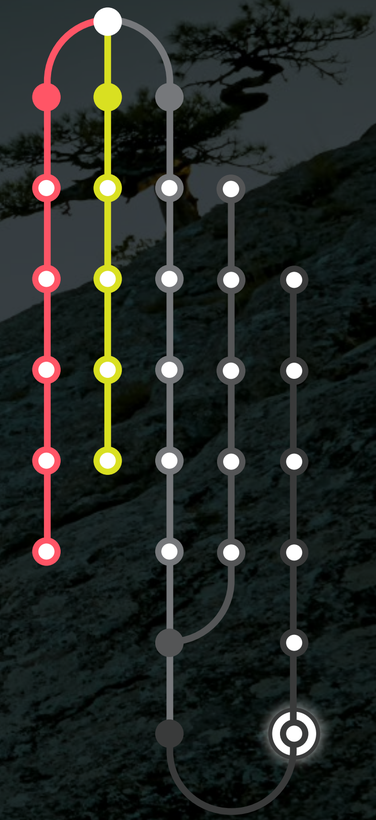
- Поддельная база
- SQLite
- Настоящая база



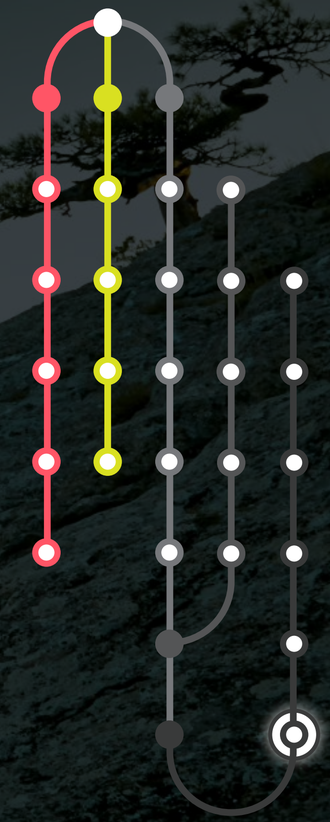
Данные



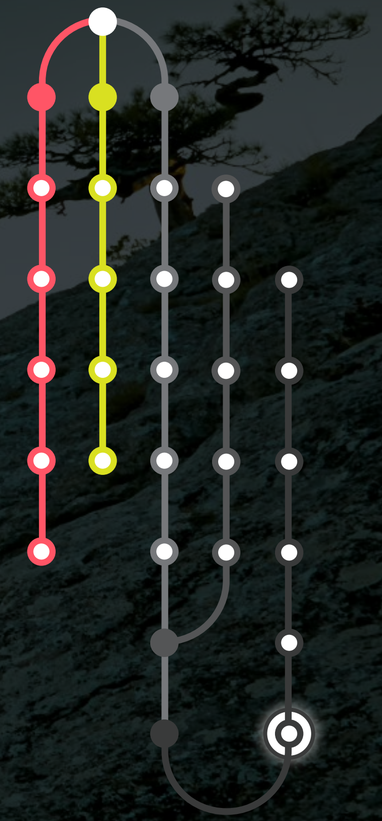
- Копия реальной базы



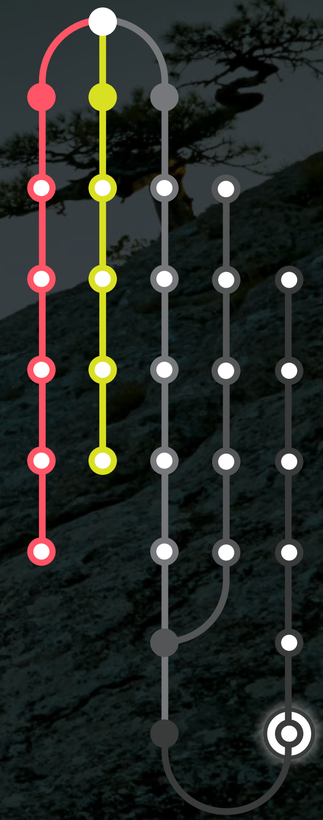
- Копия реальной базы
- Слепок реальной базы



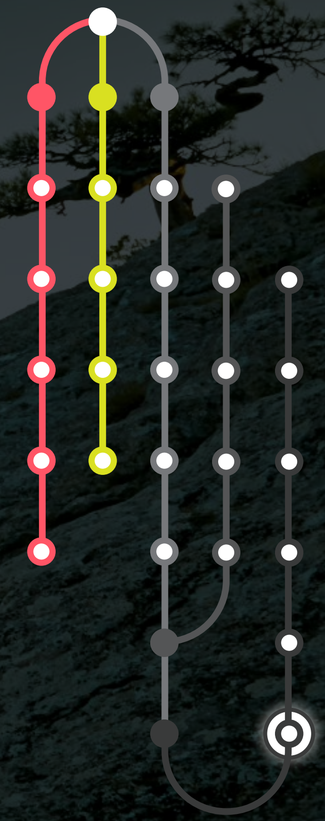
- Копия реальной базы
- Слепок реальной базы
- Вручную подготовленные данные



- Копия реальной базы
- Слепок реальной базы
- Вручную подготовленные данные
- Пустая база

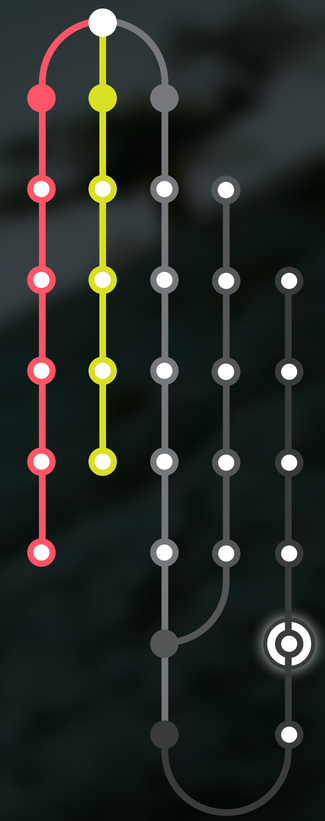


- Копия реальной базы
- Слепок реальной базы
- Вручную подготовленные данные
- Пустая база
- Фабрики

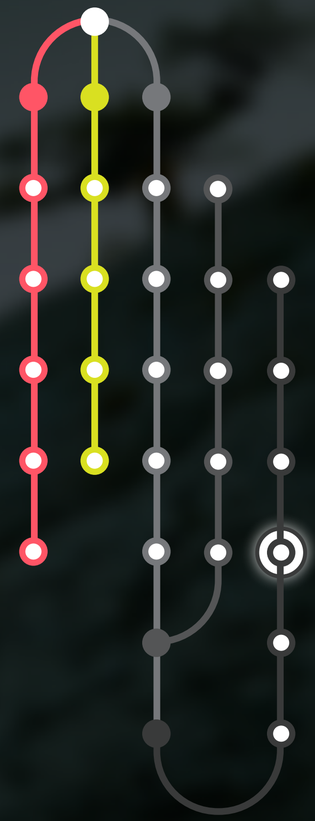


Фабрики

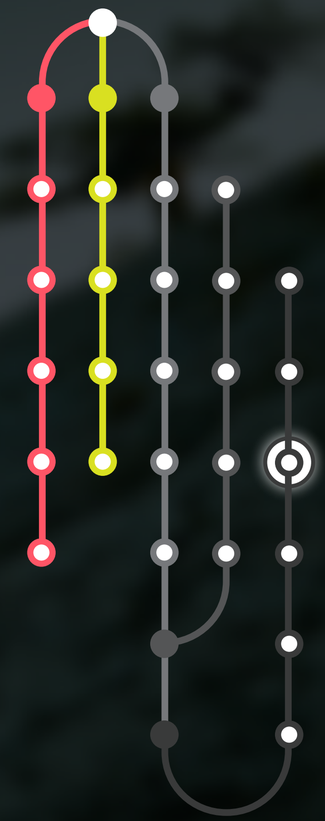
```
class UserFactory(...):
    class Meta:
        model = models.User

    first_name = factory.Sequence(
        lambda n: 'User #{}'.format(n)
    )
    group = factory.SubFactory(GroupFactory)

user = UserFactory()
```



Тестирование приватного метода?

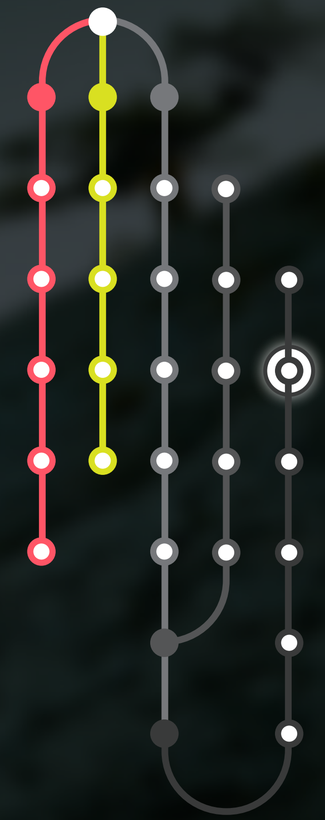


```
def count_german_letters(self):
    return len([
        x for x in self._string
        if self._is_german_letter(x)
    ])

def delete_german_letters(self):
    return ''.join(
        x for x in self._string
        if not self._is_german_letter(x)
    )

def _is_german_letter(self, c):
    return c in string.ascii_lowercase or c in 'ÄäÖöÜüßß'
```



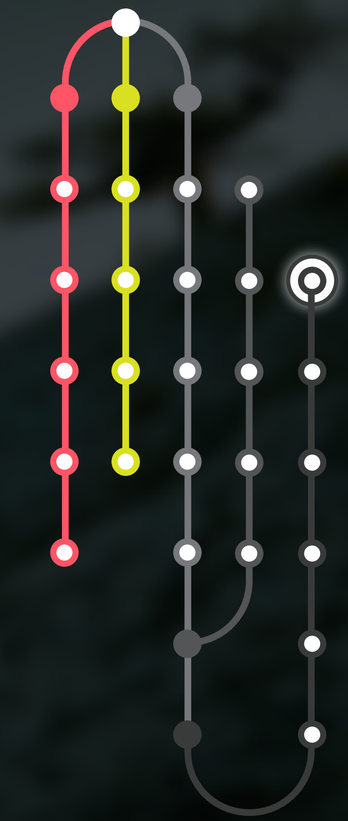
Глобальный патчинг

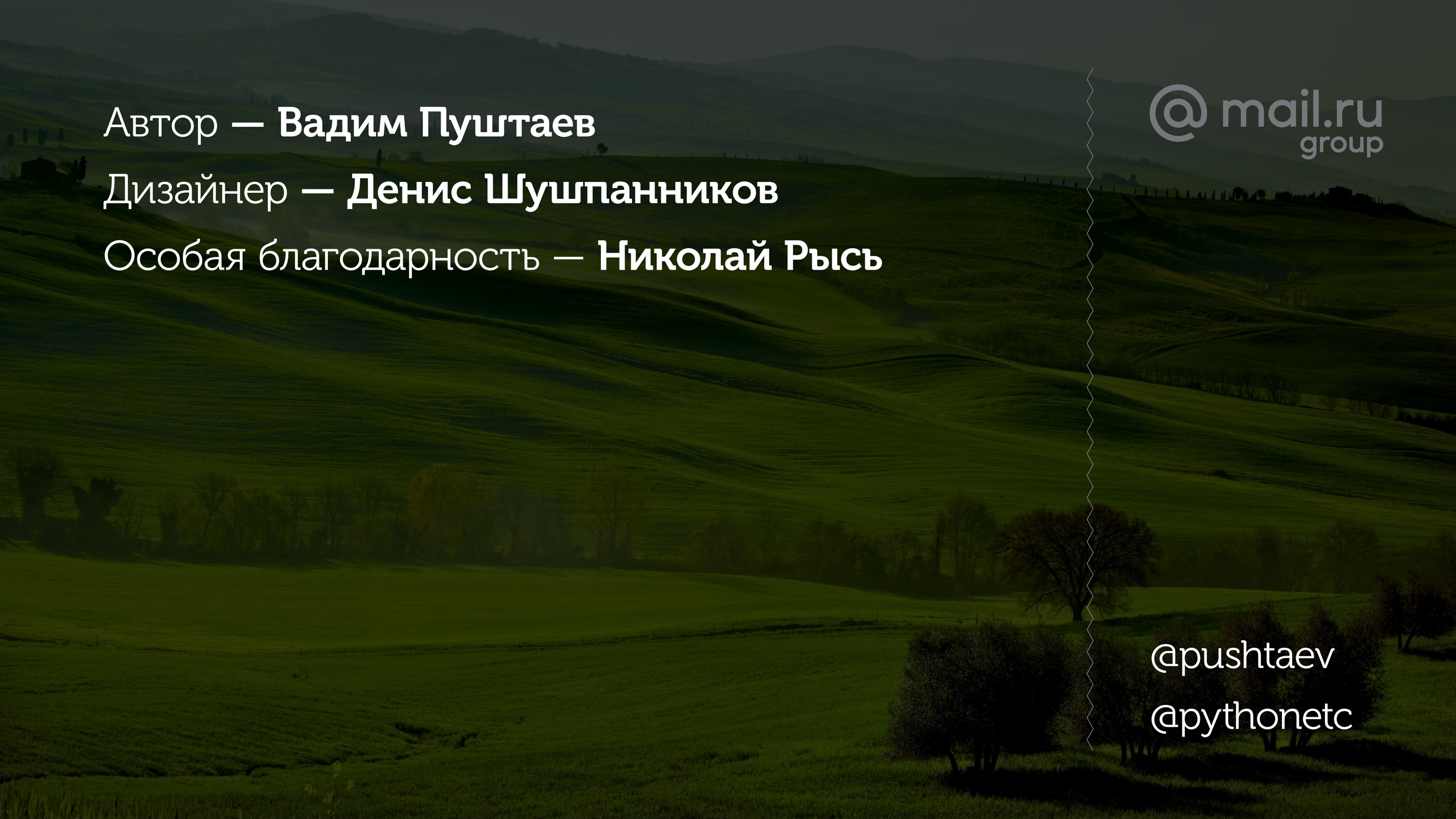
```
def setUp(self):
    super(BaseTestCase, self).setUp()

    self._patchers = {
        'etcd': patch('common.utils.etcd.Etcd.write'),
        'celery': patch('celery.current_app.send_task'),
    }

    self._patchers['etcd'].return_value = None
    self._patchers['celery'].return_value = True

    for patcher in self._patchers.values():
        patcher.start()
```



Таблицы результатов

```
import pytest
@pytest.mark.parametrize("test_input,expected", [
    ("3+5", 8),
    ("2+4", 6),
    ("6*9", 42),
])
def test_eval(test_input, expected):
    assert eval(test_input) == expected
```





Автор — **Вадим Пуштаев**

Дизайнер — **Денис Шушпанников**

Особая благодарность — **Николай Рысь**

@mail.ru group

@pushtaev

@pythonetc