
# Тесты во фронтике

Пара идей о том, как можно тестировать приложения frontik

Шапошников А.А.

# Мотивация и цель

”TDD во фронтике”

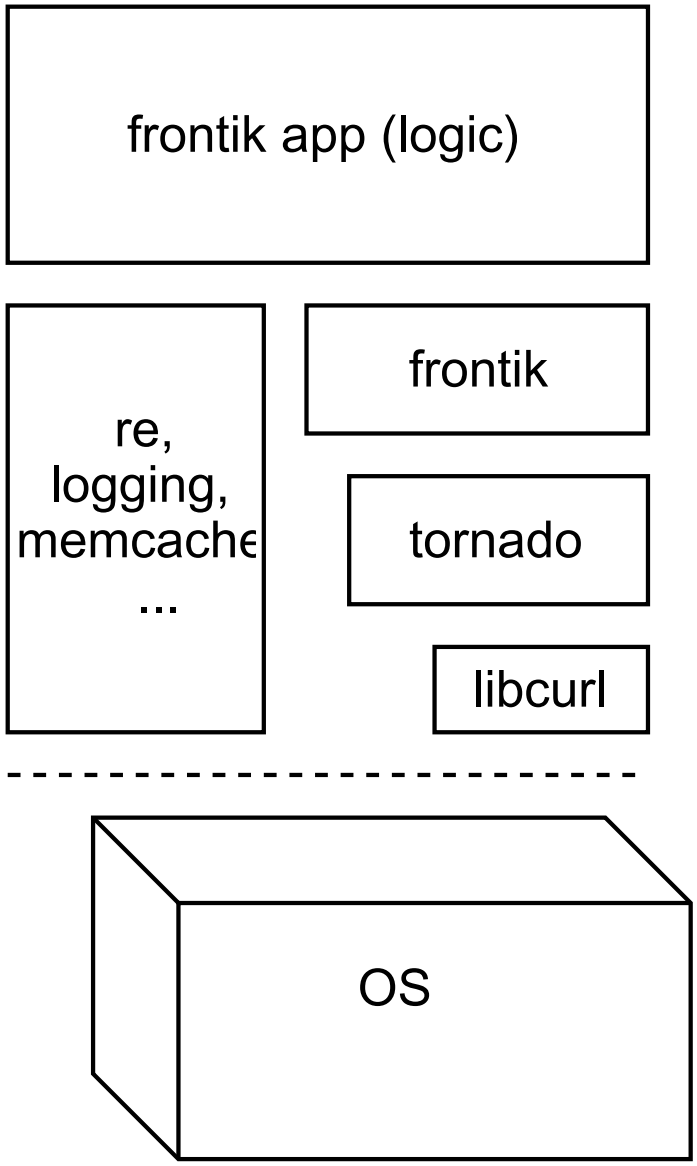
Обеспечить слой logic системой, позволяющей отлавливать баги связанные с изменением контрактов с внешними сервисами

В целом дружить с нашей экосистемой

# Требования к тестам

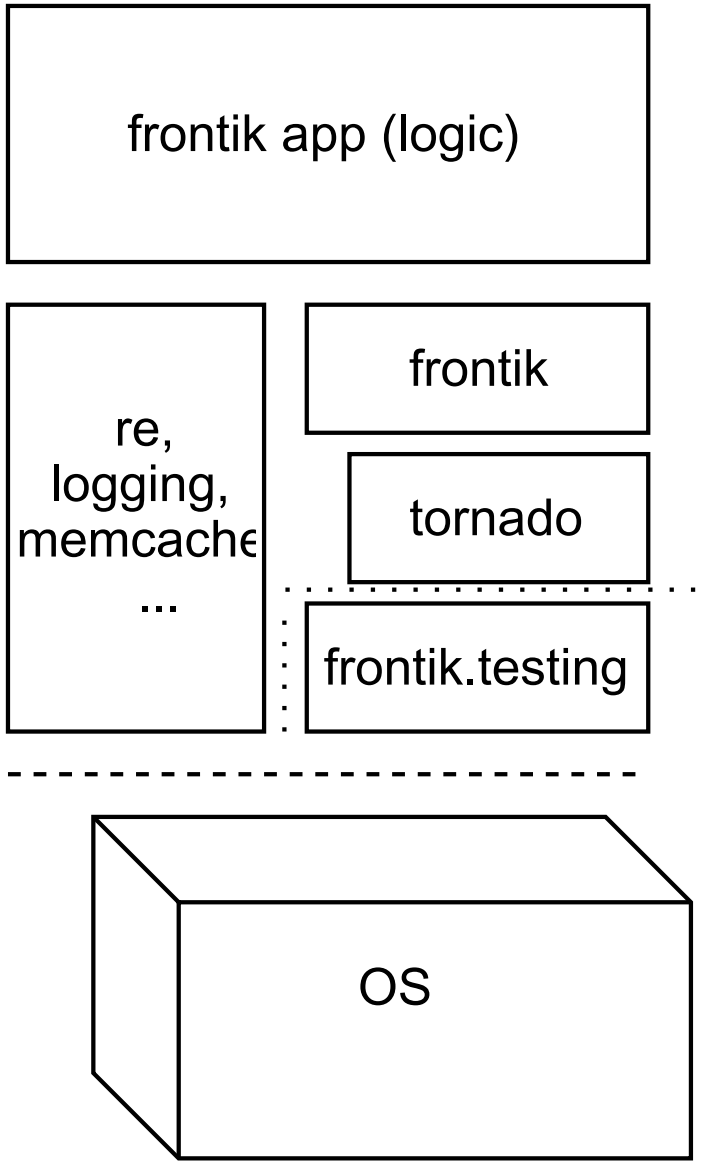
1. Наглядность и читаемость, совместимость с unittest инфраструктурой python

2. Не должны зависеть от способа их запуска и чего-то требовать от окружения (т.е. unit тесты)


frontik app (logic)
re,
logging,
memcache
...
frontik
tornado
libcurl
OS

# Тестовый фреймворк

- Приложения frontik ходят в бэкэнды, затем опционально накладывают xsl
- Такая узкая специализация позволяет встроиться и перехватывать запросы в сетевом стеке и приложение изолировано


frontik app (logic)
re,
logging,
memcache
...
frontik
tornado
frontik.testing
OS

# Ключевые решения в ходе разработки

– Встроен во frontik. Использует настоящий экземпляр frontik с подмененным http-client

– **Нет** tornado/ioloop, callback’и вызываются через тестовый фреймворк

– **Нет** HTTP и сокетов, http-client напрямую ходит к мокам сервисов

– собственно наложение xsl можно тестировать, но зачем?

# Код под тестом

```
1  def vacancies_by_manager(handler, employer_manager_id):
2      def vacancy_list_callback(xml, response):
3          if response.code == 403:
4              raise HTTPError(403)
5          if xml is None:
6              raise HTTPError(503)
7          vacancy_ids = xml.xpath('//vacancy/@id')
8          if vacancy_ids:
9              self.doc.put(Element('some'), highlight =
10                 handler.config.vacancies_prolongate_high
11     url = "{host}employerManager/"+
12     "{employer_manager_id}/vacancies".format(
13         host = handler.config.serviceHost,
14         employer_manager_id = employer_manager_id)
15     handler.get_url(url, data =
16                     {'userId' : handler.session.user_id}
17                     callback = vacancy_list_callback)
```

# Тесты

```
1  class TestEmployerVacanciesPage(TestCase):
2      def test_vacancies_by_manager(self,):
3          env = EmptyEnvironment()
4          set_stub(env, manager_vacancies,
5              'serviceHost/ManagerVacancies.xml')
6          doc = env.configure(
7              vacancies_prolongate_highlight = 3
8          ).call(
9              vacancies_by_manager,
10             employer_manager_id='3',
11         ).get_doc().root_node # or get_result()
12         em_id_actual = doc.xpath('//some/xPath/[0]')
13         self.assertEqual(em_id_actual, '3')
```

# Тесты (продолжение)

`set_stub` - из моков, устанавливает в окружении mock

`manager_vacancies` - пара (сервис, ресурс) т.е. url, на который будет поставлен mock

`'serviceHost/ManagerVacancies.xml'` - файл с ответом

`vacancies_by_manager` - это метод под тестом

`EmptyEnvironment` - импортирован из frontik.testing, создает окружение с моками внешних сервисов

`call` - позволяет вызвать тестируюемую функцию в подготовленном окружении

caveat: не поддерживается более одного вызова call в одном тесте

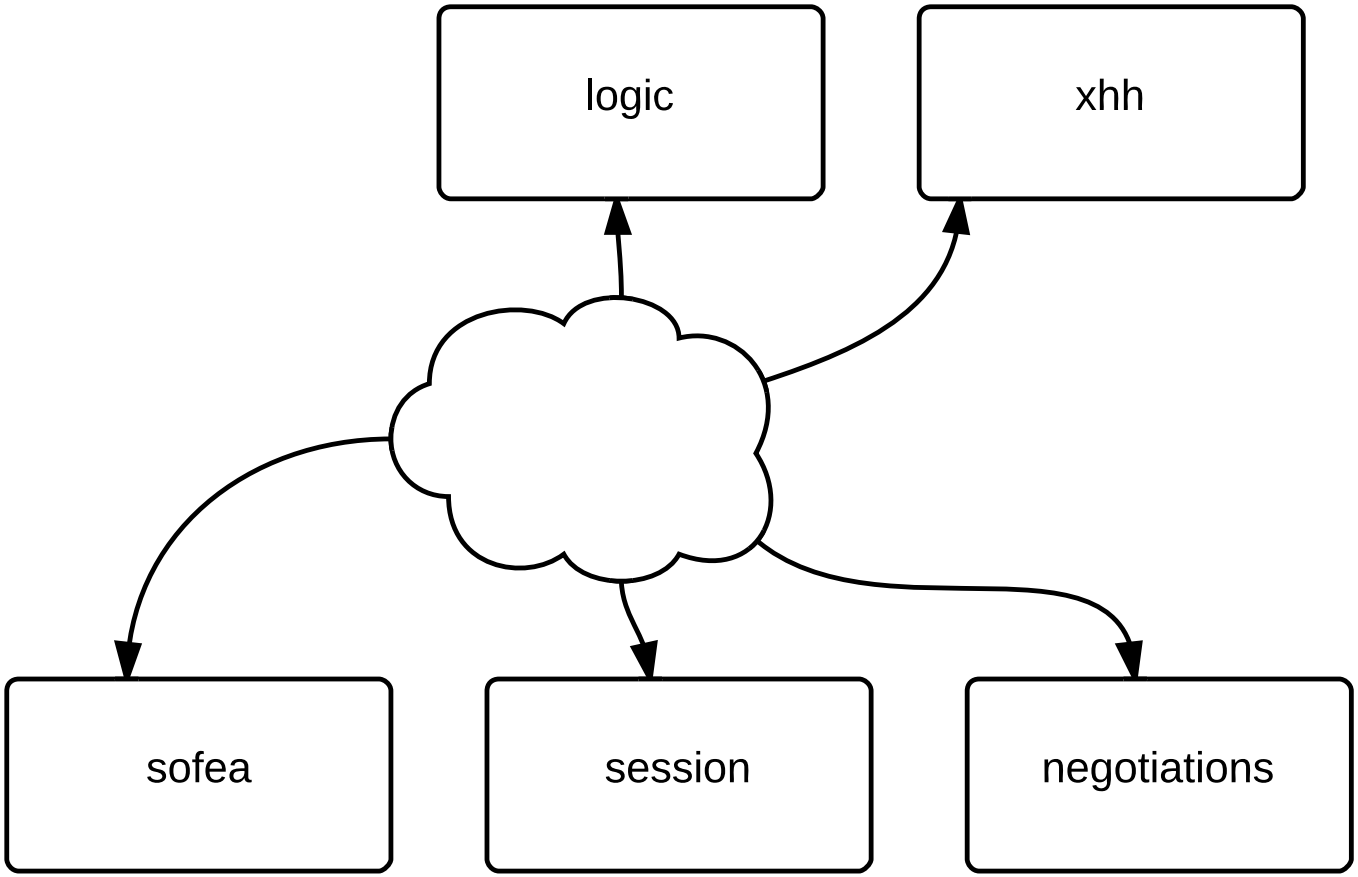
# Вызов тестирования интеграционного слоя

Почти все тесты ходят за одними и теми же данными (i.e. вакансия, сессия), и в то же время не всем подойдет один и тот же ответ

Ответы генерируют внешние сервисы, которые лежат в другом репозитории и обновляются в случайный момент времени

Если контракт одного HTTP ресурса поменялся с одной стороны, тесты (в идеальном мире) показывают, что код, существенно его использующий на другой стороне, сломался

Как ориентироваться в моках?


logic
xhh
sofea
session
negotiations

# Репозиторий моков

Напротив, развязан с frontik’ом и даже по возможности с питоном

Как оказалось, одного xml мало, мы храним рядом с ним ещё и необходимые header’ы

И ещё отдельно пару сервис-ресурс (aka host-url)

# Куда можно двигаться дальше

1. Документация
2. Более внятные сообщения об ошибках
3. Поддержка более широкого интерфейса, например IO loop
4. Версионность моков
5. ”Встроенная” поддержка protobuf
6. Автоматизированный прогон тестов всех репозиториев после изменений в моках
7. Больше сахара, чтобы тесты не выглядели так страшно
8. Автоматическое определение race conditions
9. ...

# Резюме

Справились с проблемой TDD во фронтике

Создали репозиторий для хранения того, что у нас отвечают какие сервисы

Есть надежда, что ломаться теперь будет в разработке и при сборке, а не в проде

Тесты выглядят и читаются сносно

# Предлагаю критиковать

Что не так?