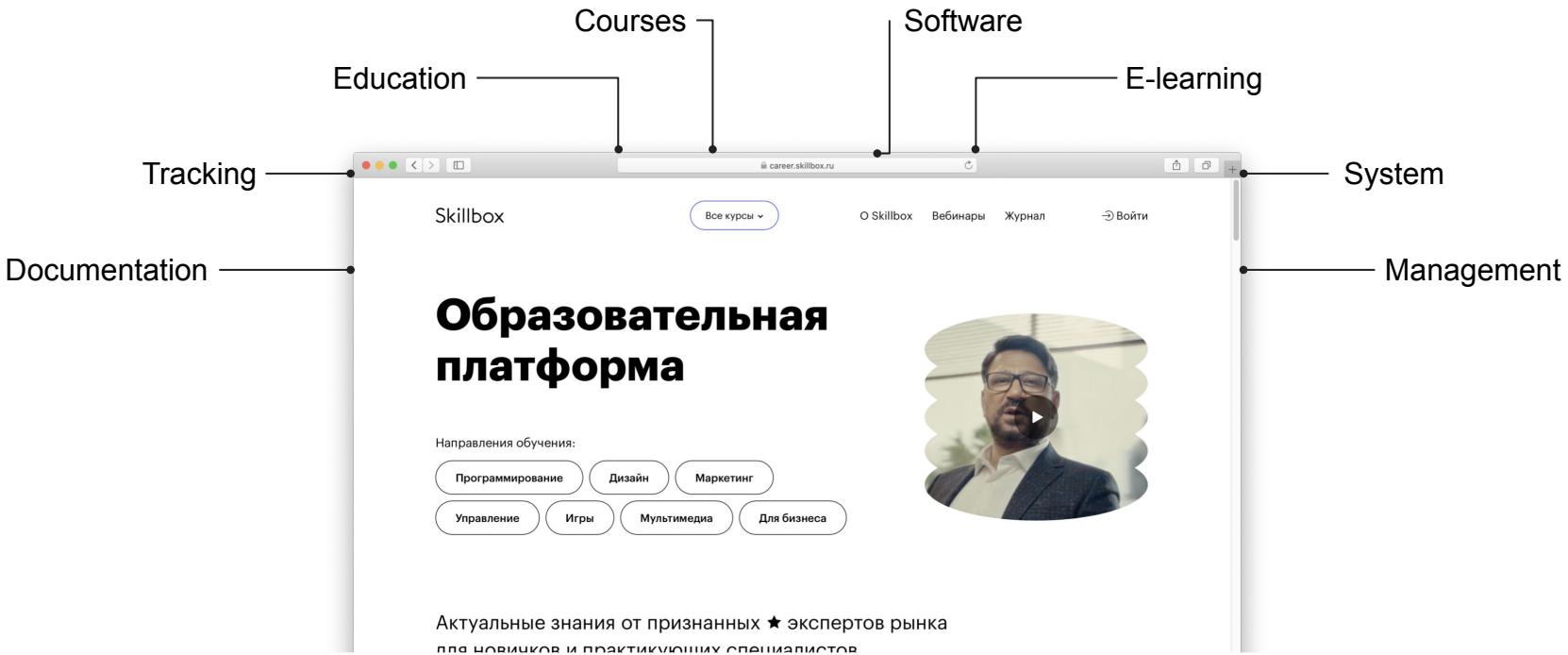
Слоеный фреймворк автотестирования

Архитектура, примеры на практике и подводные камни

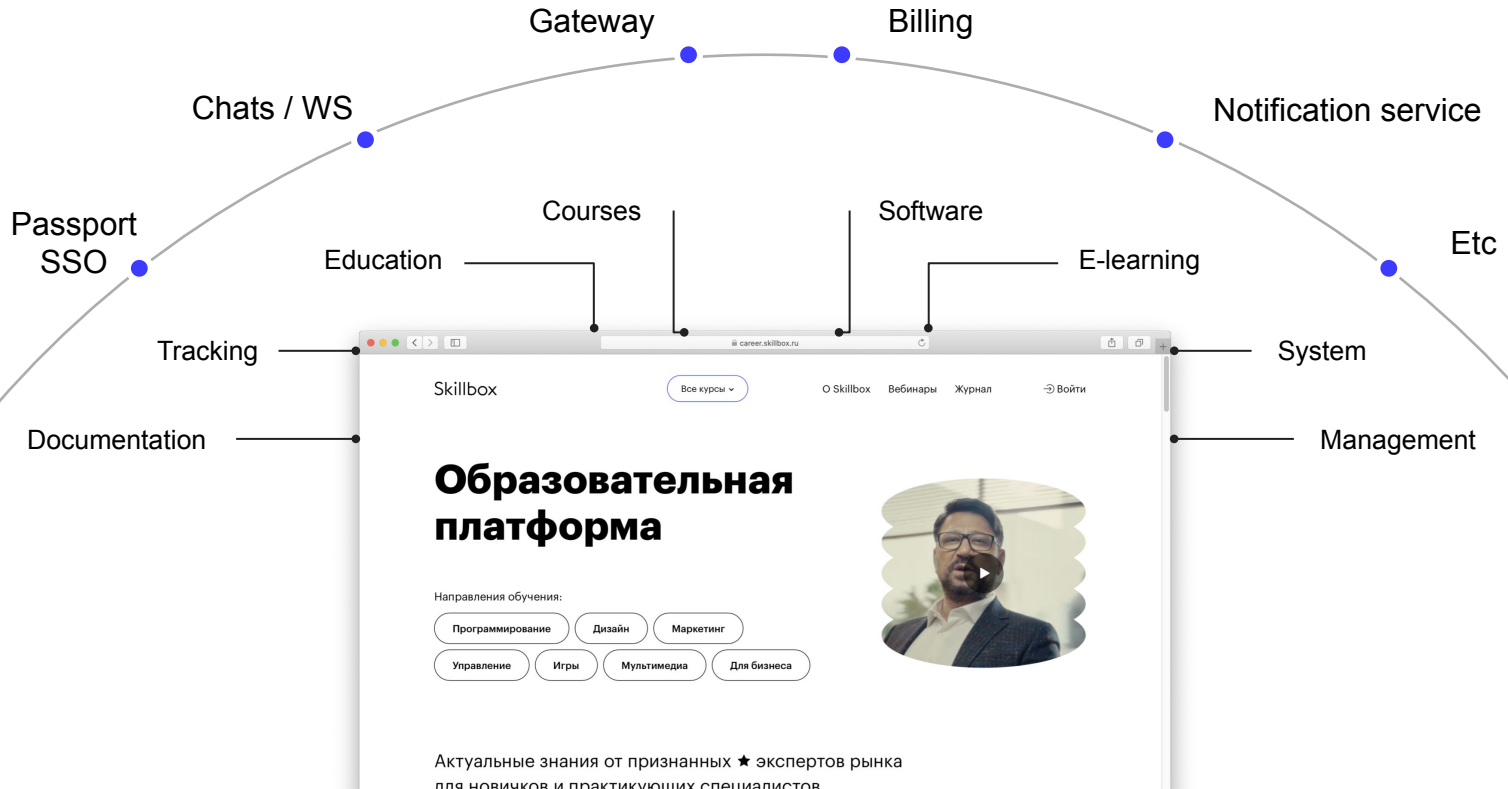
Skillbox

Роман Помелов,
QA Automation Lead

Платформа

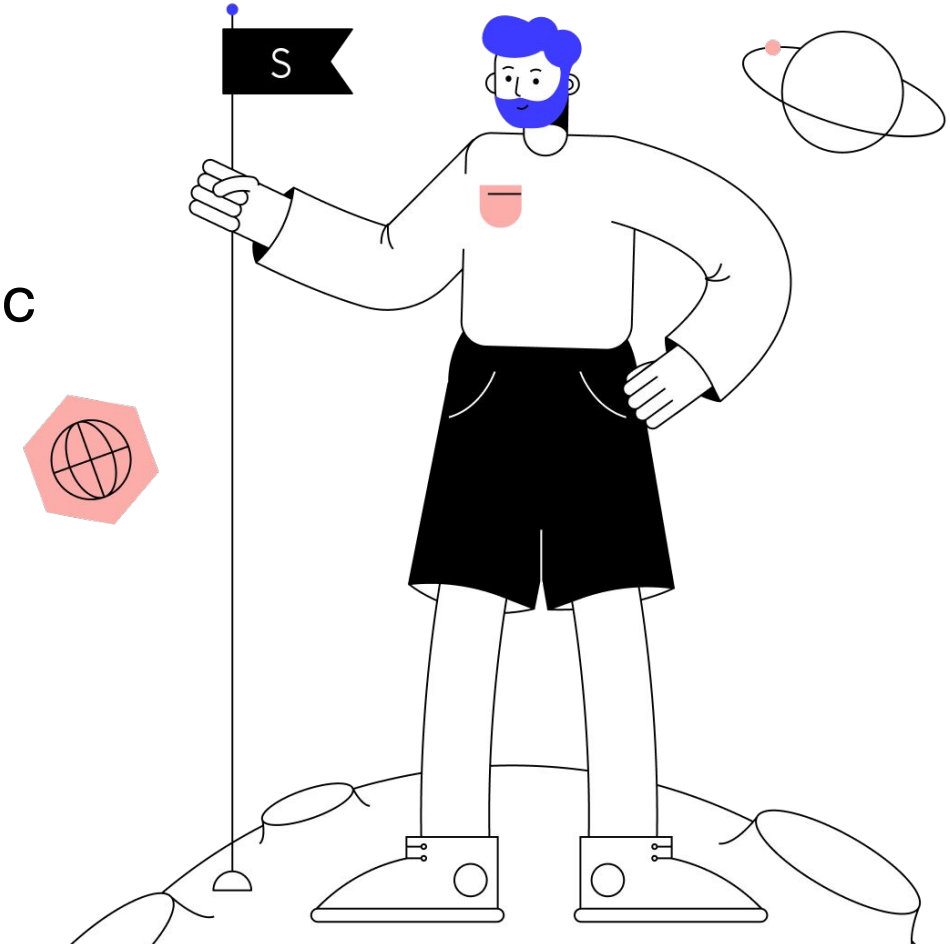

Courses
Software
Education
E-learning
Tracking
System
Documentation
Management
career.skillbox.ru
Skillbox
Все курсы
О Skillbox
Вебинары
Журнал
Войти
Образовательная платформа
Направления обучения:
Программирование
Дизайн
Маркетинг
Управление
Игры
Мультимедиа
Для бизнеса
Актуальные знания от признанных ★ экспертов рынка

Платформа


Gateway
Billing
Chats / WS
Notification service
Passport
SSO
Courses
Software
Education
E-learning
Etc
Tracking
System
Documentation
Management
career.skillbox.ru
Skillbox
Все курсы
О Skillbox
Вебинары
Журнал
Войти
Образовательная платформа
Направления обучения:
Программирование
Дизайн
Маркетинг
Управление
Игры
Мультимедиа
Для бизнеса
Актуальные знания от признанных ★ экспертов рынка

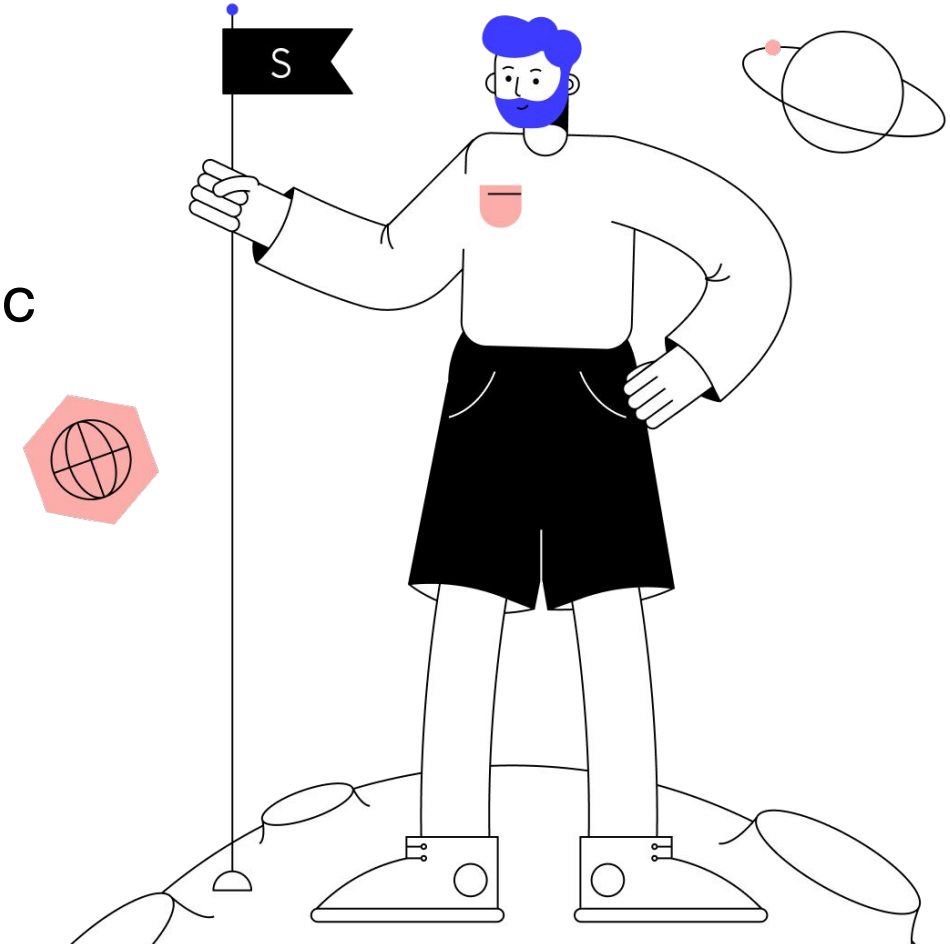
План-ураган

1. Тех.долг растет
2. ISTQB Arch
3. Препарируем тест-кейс
4. Задачки из жизни
5. Итоги



План-ураган

1. **Тех.долг растет**
2. ISTQB Arch
3. Препарируем тест-кейс
4. Задачки из жизни
5. Итоги



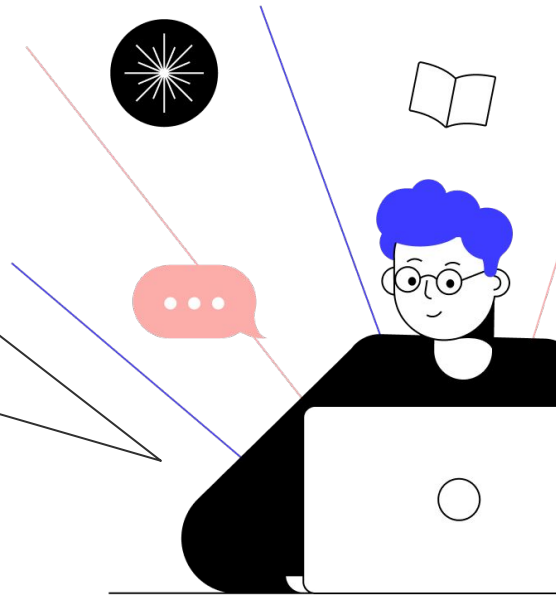
Я оказался на новом проекте

Как я размышлял:

1. UI тесты → Selenium
2. API тесты → API client over requests
3. База данных → ORM SQLAlchemy
4. ELK/Graylog → пока не до этого
5. Test Runner → pytest

Потом пришел QA Lead

Когда тесты-то будут?



Потом пришел QA Lead

Ладно, сейчас напишу по-быстрому, добавлю TODO, а потом сделаю красиво…

Кто оставлял #TODO и никогда к нему не возвращался?

Как понять, что тех.долг растёт?

Как понять, что тех.долг растёт?

Как понять, что тех.долг растёт?

- Новые люди пугаются
- Долгий онбординг в проект

...

Приходит сеньор и переделывает по-своему, потом никто не разбирается

Как понять, что тех.долг растёт?

Я не понимаю, как работает код

- не могу отследить логику
- не вижу / не знаю паттерн

Если я не понимаю,
значит это “плохой код”?

```
330 +
331 +     :param context: war
332 +     :param step: war
333 +     """
334 +     client_response = getattr(context.userdata.api_context.client,
  '_last_response', None)
```

Roman Pomelov @roman.pomelov · 2 days ago Owner

Дополнительный интерфейс для `_last_response` внутри клиента это 👌

Если в сценарии использовалось несколько клиентов, нужно взять последний из `_container`.

А еще есть `ResponseContext().response`, там как раз лежит последний ответ. Кажется надежнее использовать его, чем в этом месте лезть напрямую в клиента.

https://gitlab.skillbox.pro/qa/sb_atf/-/blob/dev/sb_atf/core/backend/context/response_context.py#L116

Edited by Roman Pomelov 2 days ago

Collapse replies

Артем Нетрибейчук @artem.netribeychuk · 2 days ago Author Maintainer

Я этим не горжусь, фрейм вынудил меня пойти на уговоры с совестью 😔

Все фасады и context-ы ломаются на вот этих строчках

фреймворк заставил меня

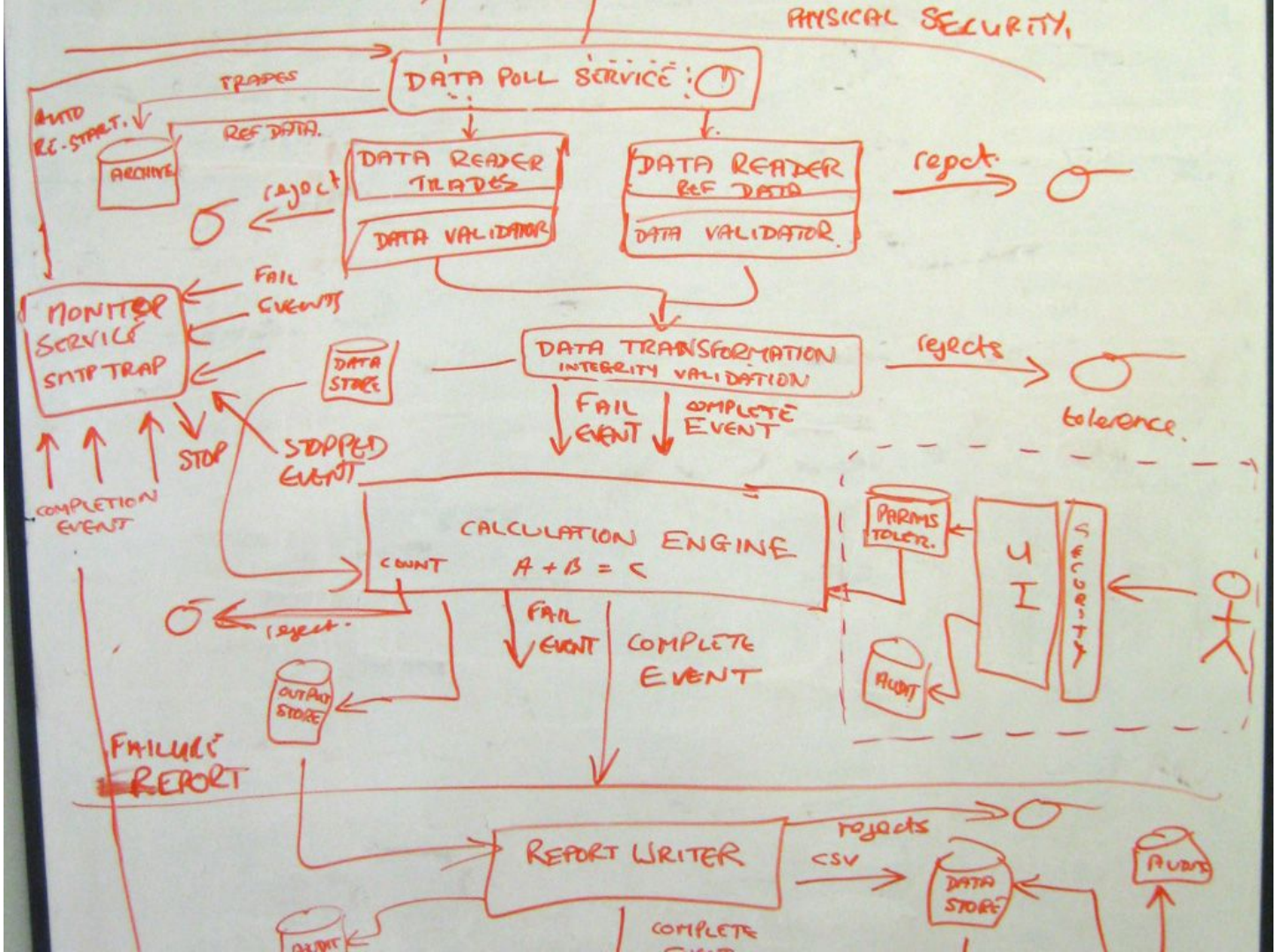
пойти на уговоры с совестью 😑

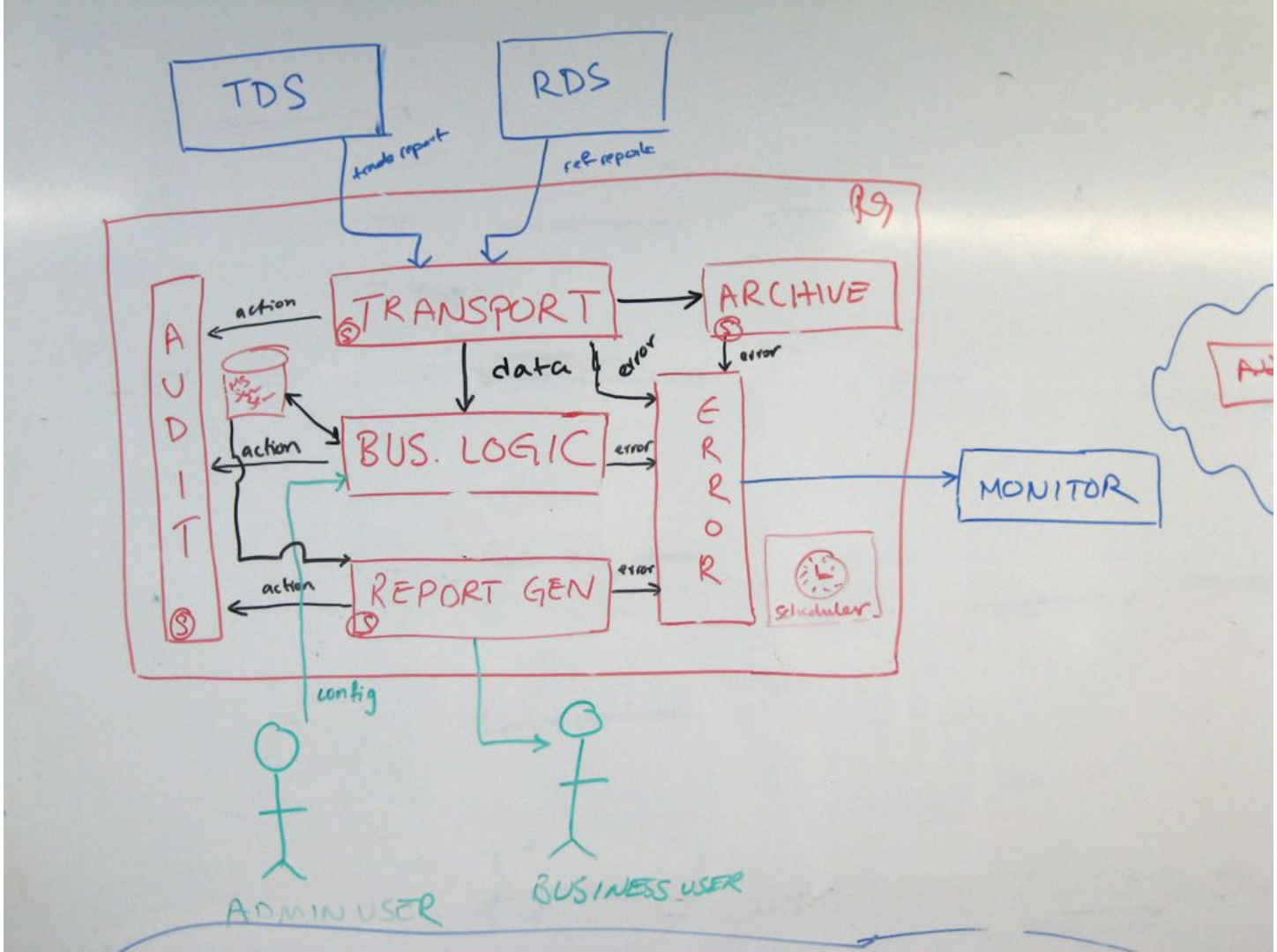
Архитектура

Архитектура в строительстве

Строгий набор документов

- геология
- поэтажный план
- инженерные коммуникации


PHYSICAL SECURITY
DATA POLL SERVICE
TRADES
REF DATA
AUTO RE-START
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
DATA READER REF DATA
DATA VALIDATOR
reject
reject
MONITOR SERVICE SNMP TRAP
FAIL EVENTS
DATA STORE
DATA TRANSFORMATION INTEGRITY VALIDATION
rejects
FAIL EVENT
COMPLETE EVENT
tolerance
STOP
STOPPED EVENT
COMPLETION EVENT
CALCULATION ENGINE
COUNT
A + B = C
PARMS TOLER.
UI
SECURITY
AUDIT
reject
FAIL EVENT
COMPLETE EVENT
OUTPUT STORE
FAILURE REPORT
REPORT WRITER
rejects
CSV
DATA STORE
AUDIT
COMPLETE


TDS
RDS
trade report
ref repoole
TRANSPORT
ARCHIVE
action
data
error
AUDIT
BUS. LOGIC
ERROR
MONITOR
REPORT GEN
scheduler
config
ADMIN USER
BUSINESS USER

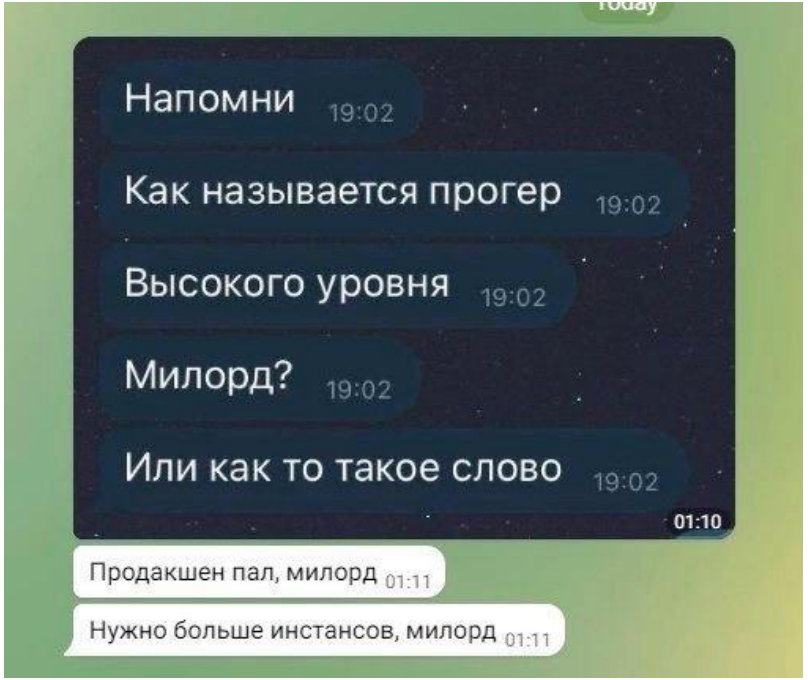
Архитектура в IT

Кто во что горазд, хотя есть нотации

Цена ошибки в ПО воспринимается не так остро

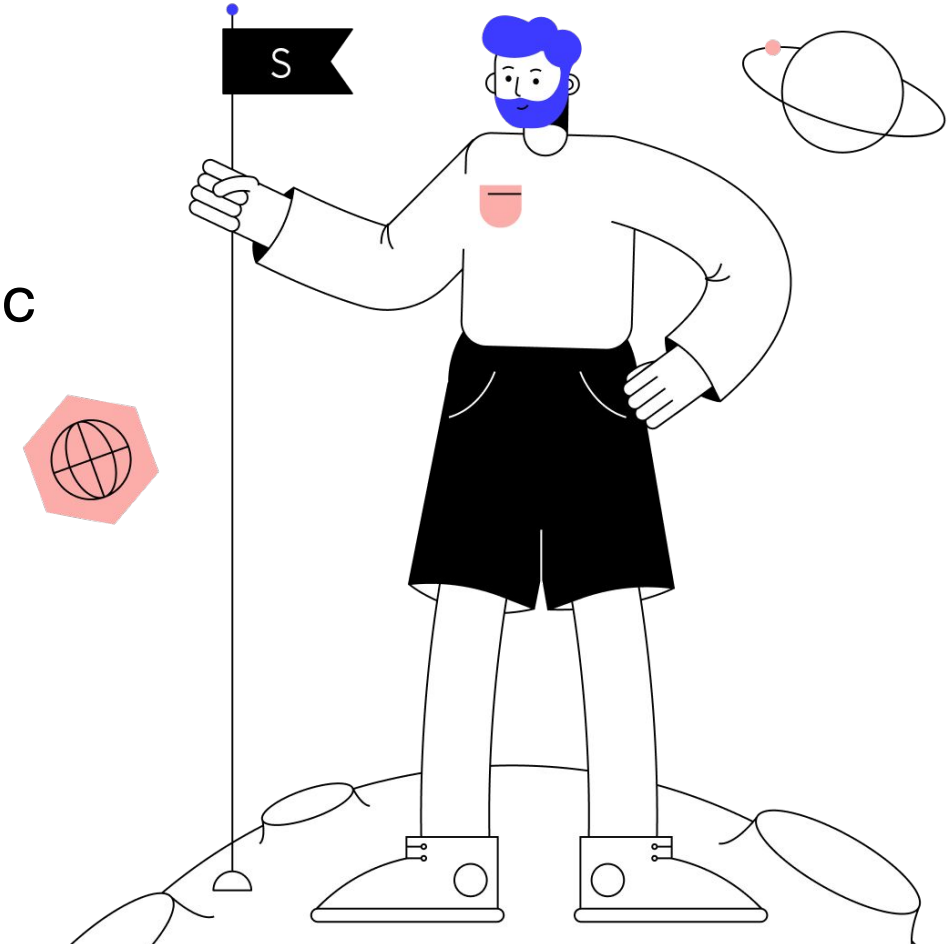
Архитектура в IT

Может вообще проектировать не обязательно?



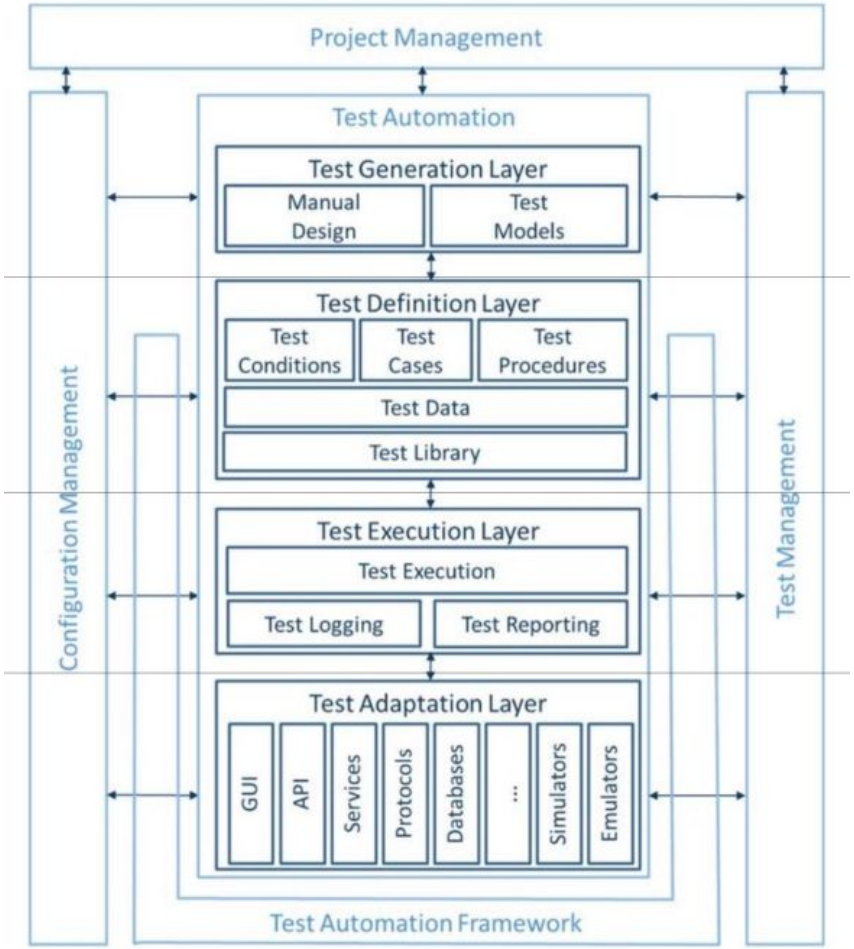
План-ураган

1. Тех.долг растет
2. **ISTQB Arch**
3. Препарируем тест-кейс
4. Задачки из жизни
5. Итоги



ISTQB gTAA

Generic Test Automation Arch



Пора подумать про архитектуру

С чего начинается фреймворк?

Берем знакомые технологии и применяем на новом проекте

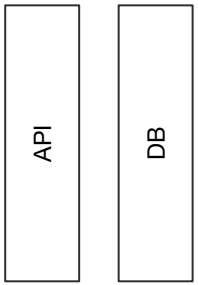
Пора подумать про архитектуру

Что у нас есть:

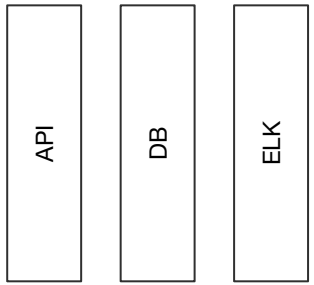
1. UI тесты → Selenium
2. API тесты → API client over requests
3. База данных → SQLAlchemy
4. ELK/Graylog → пока не до этого
5. Test Runner → pytest

Test Adaptation Layer

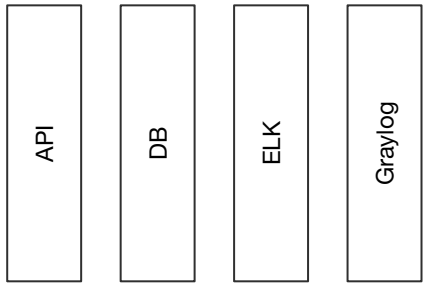
Test Adaptation Layer



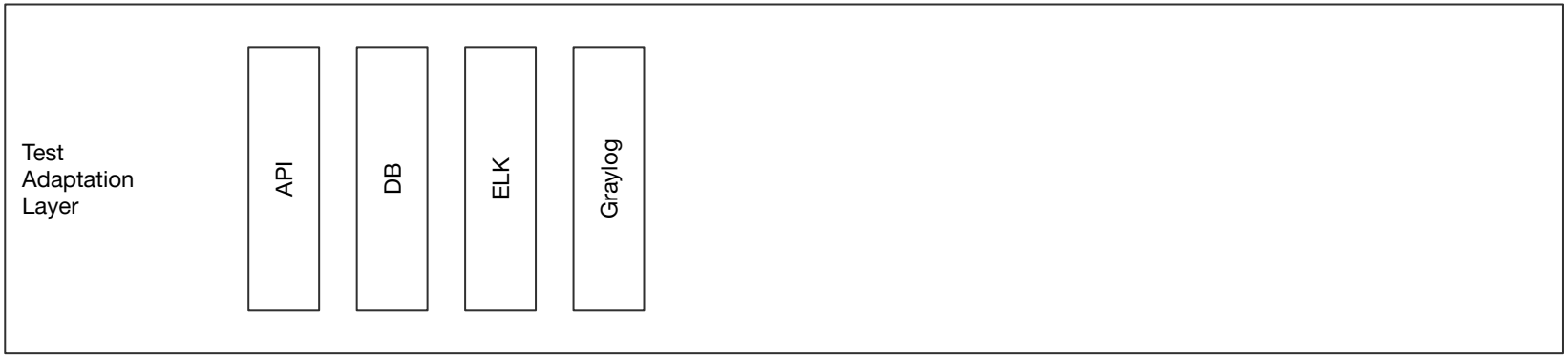
Test Adaptation Layer



Test Adaptation Layer



Test Adaptation Layer

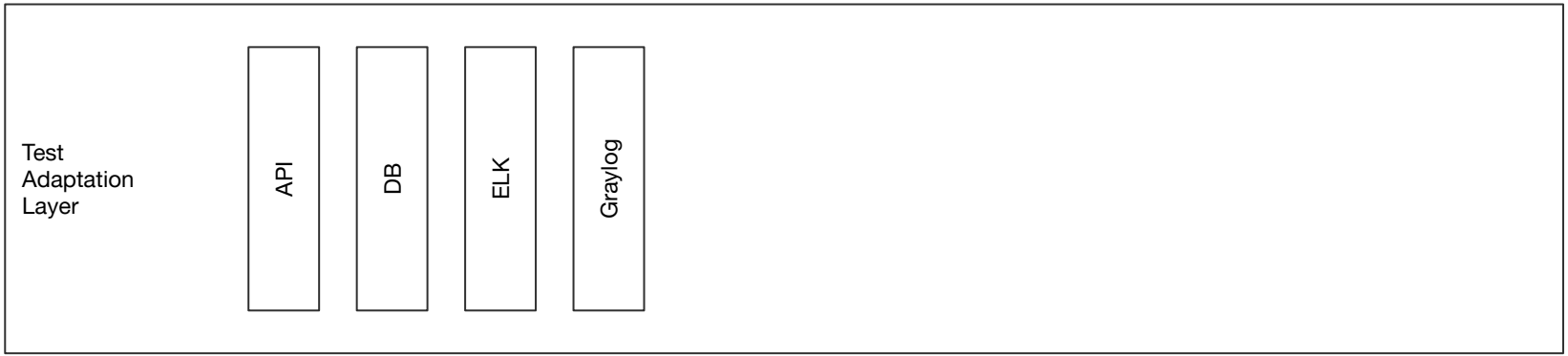


Test Adaptation Layer

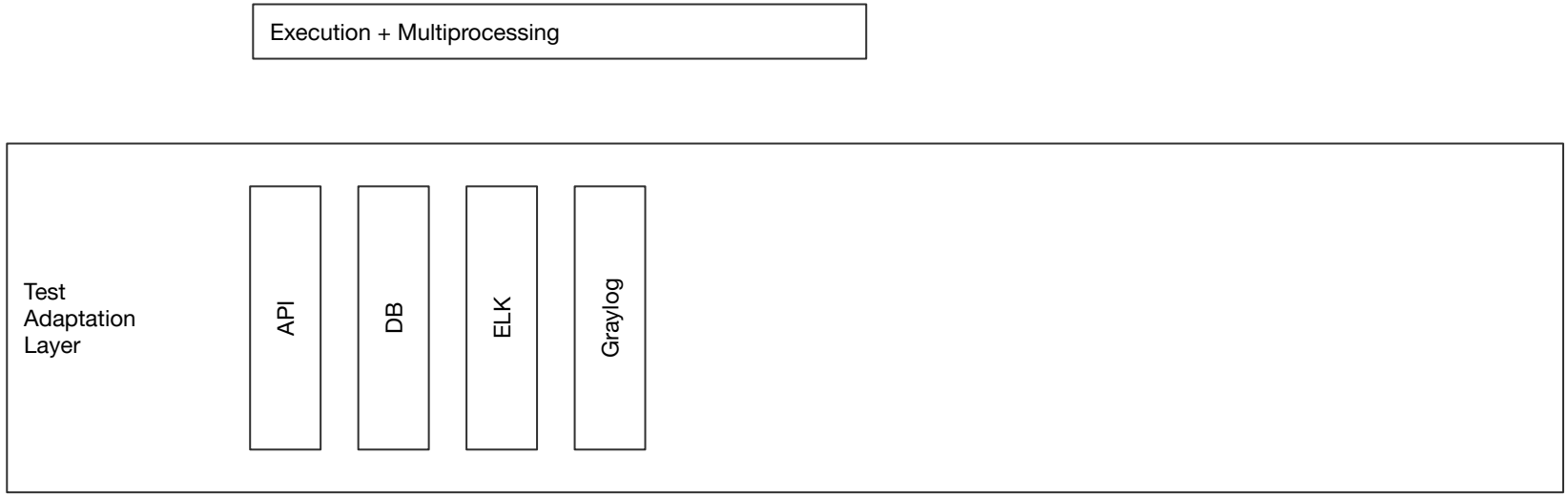
Набор клиентов и адаптеров

- подготовка окружения
- **управление объектом тестирования**
- мониторинг

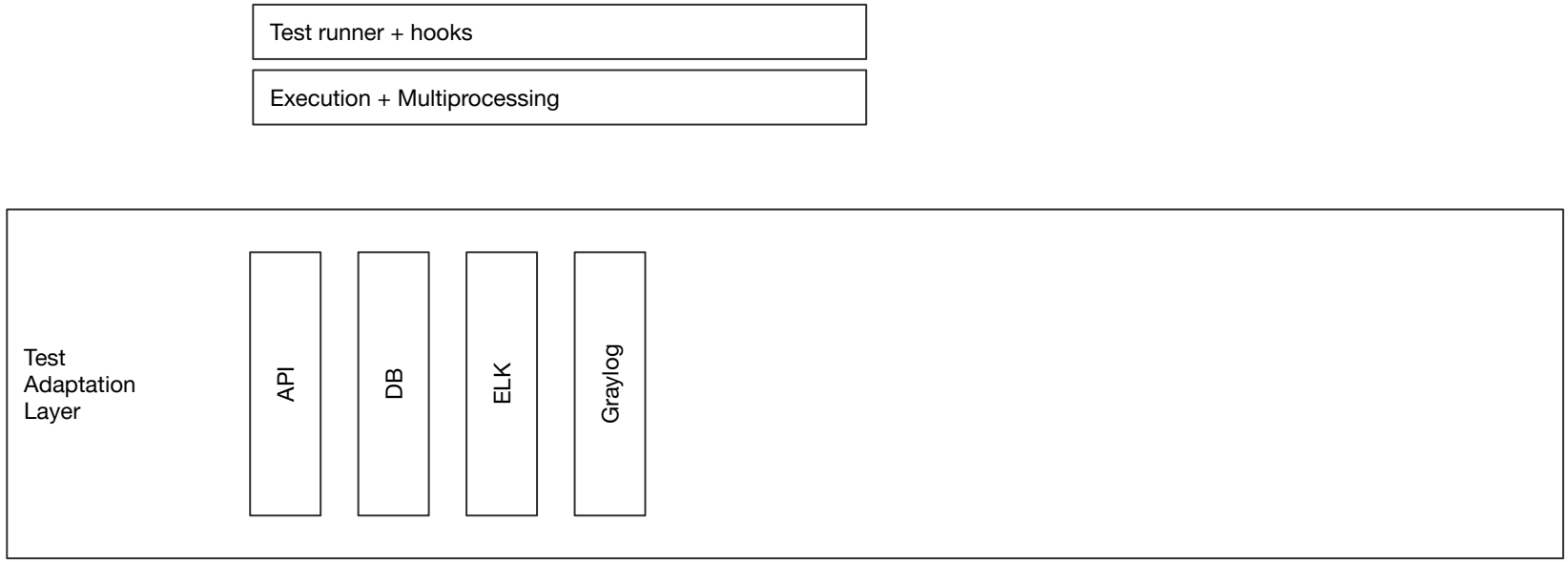
Test Execution Layer



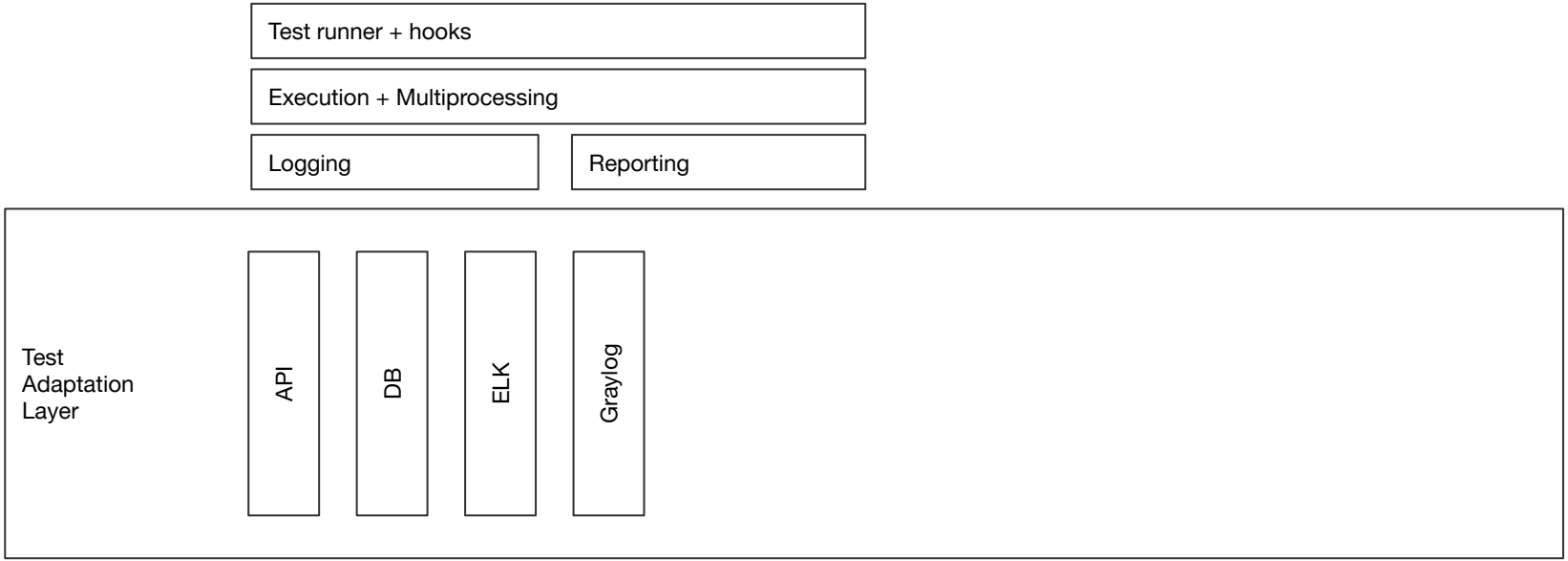
Test Execution Layer



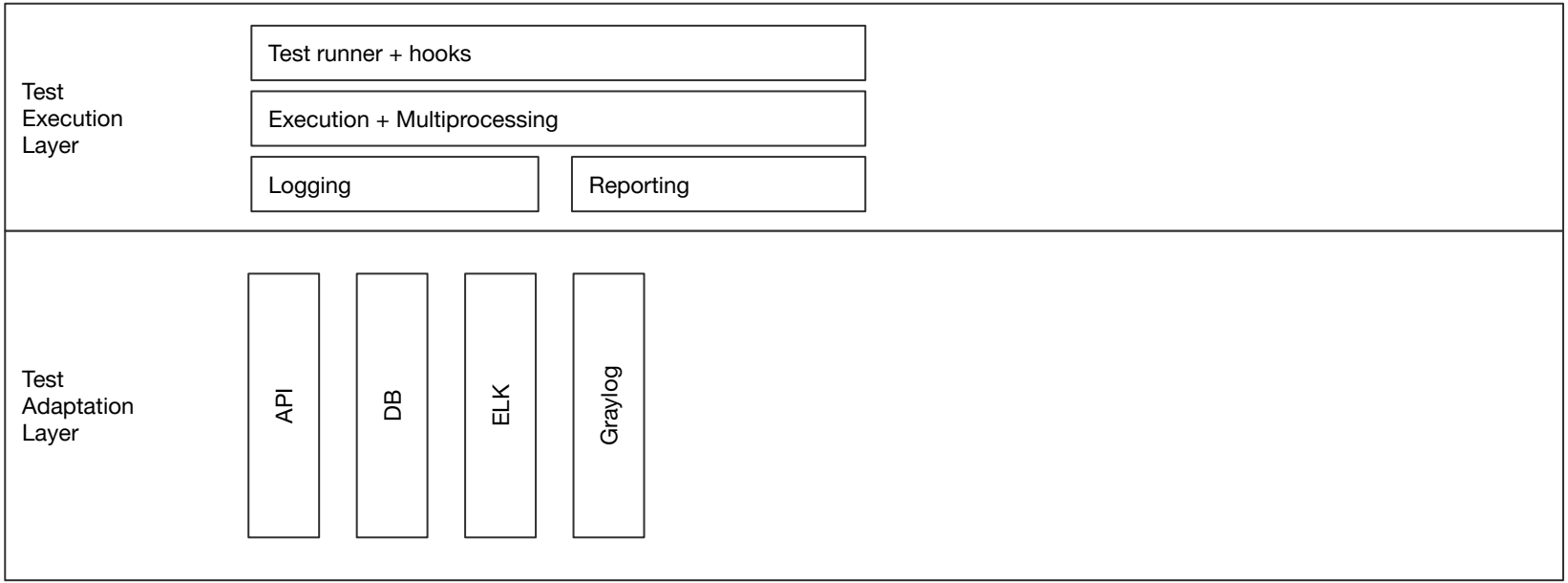
Test Execution Layer



Test Execution Layer



Test Execution Layer

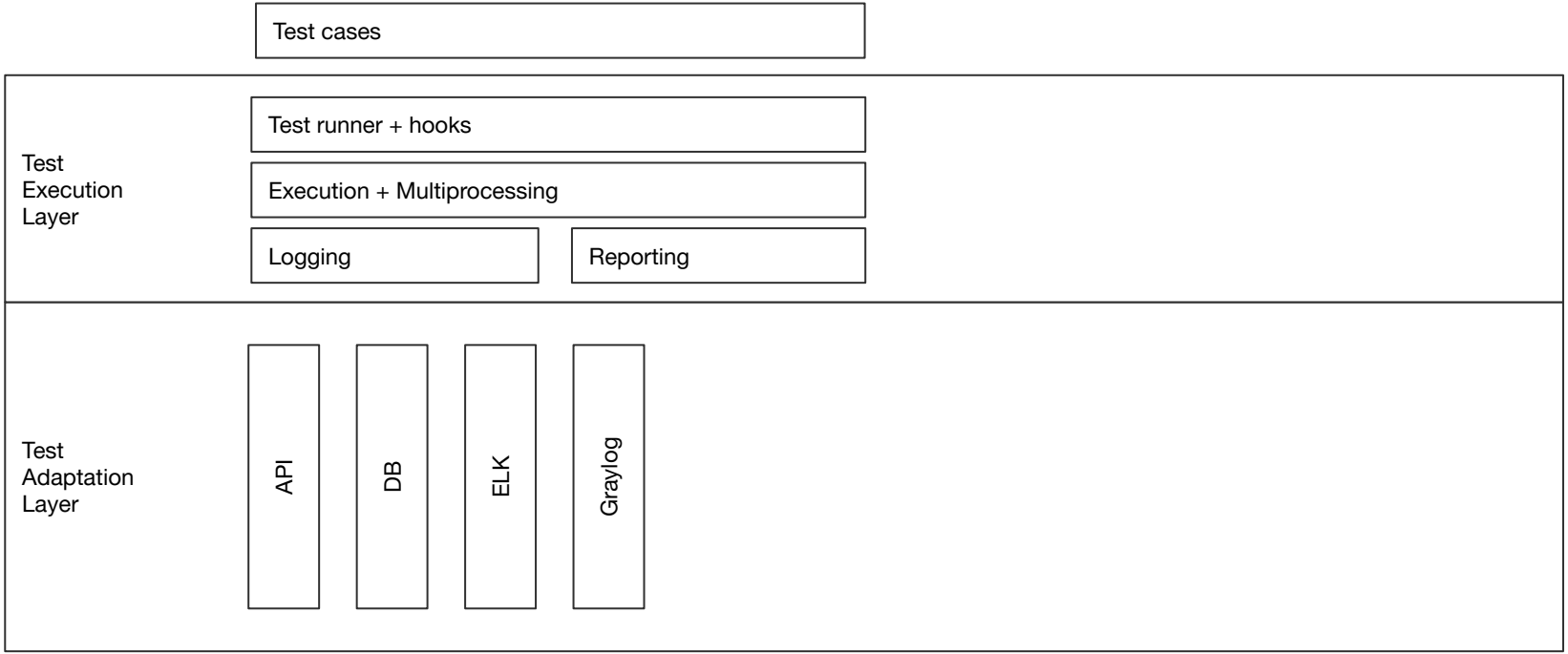


Test Execution Layer

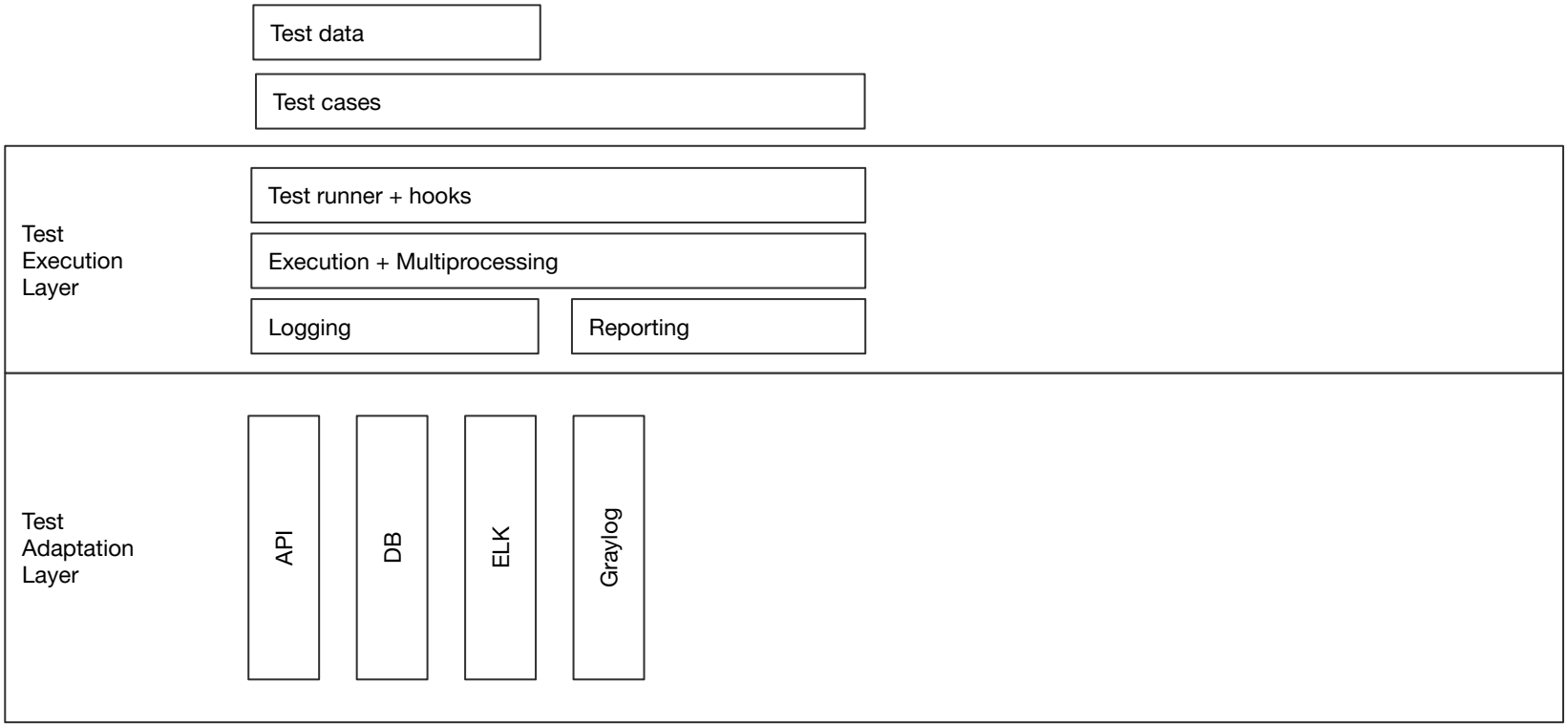
Запуск и управление потоком тестов

- test suites
- smart multithreading
- logging + reporting
- rerun

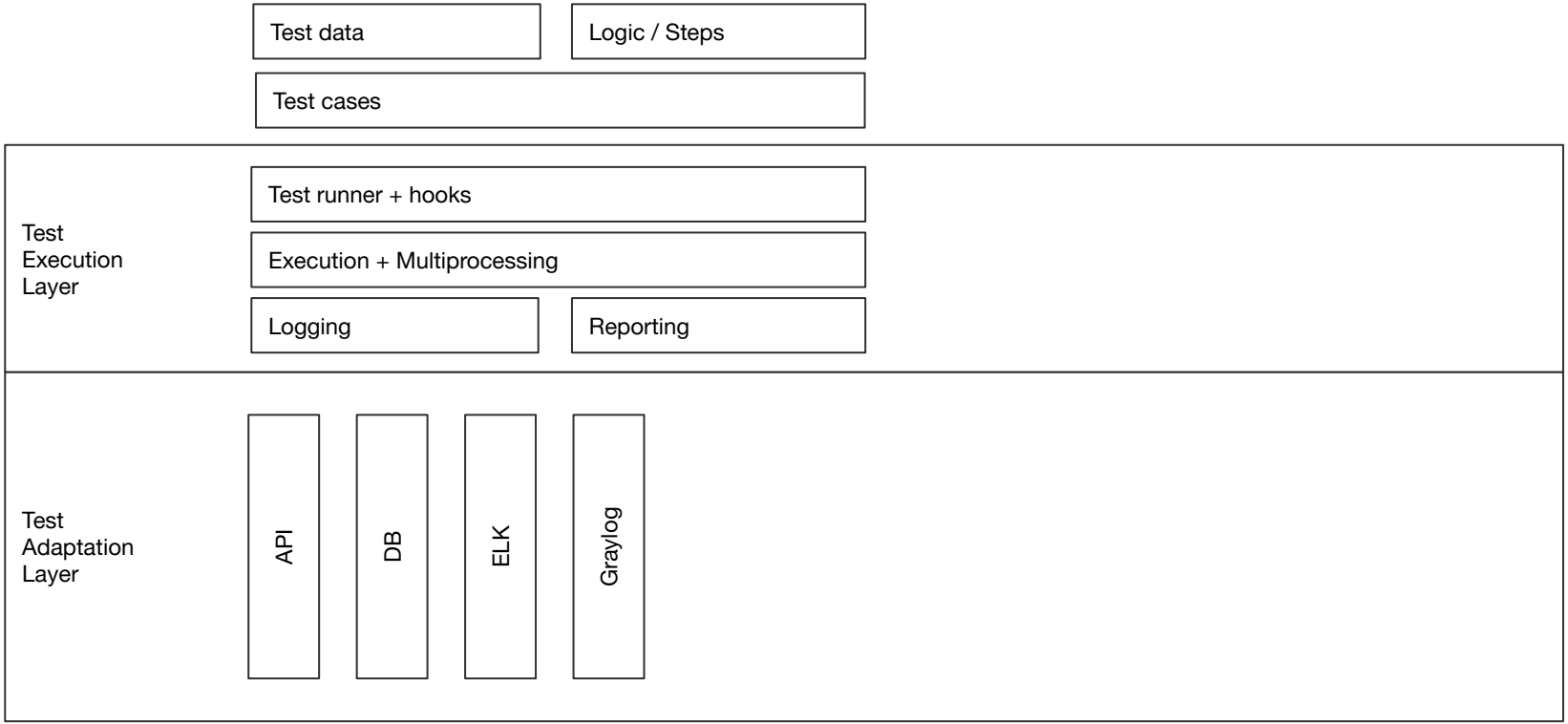
Test Definition Layer



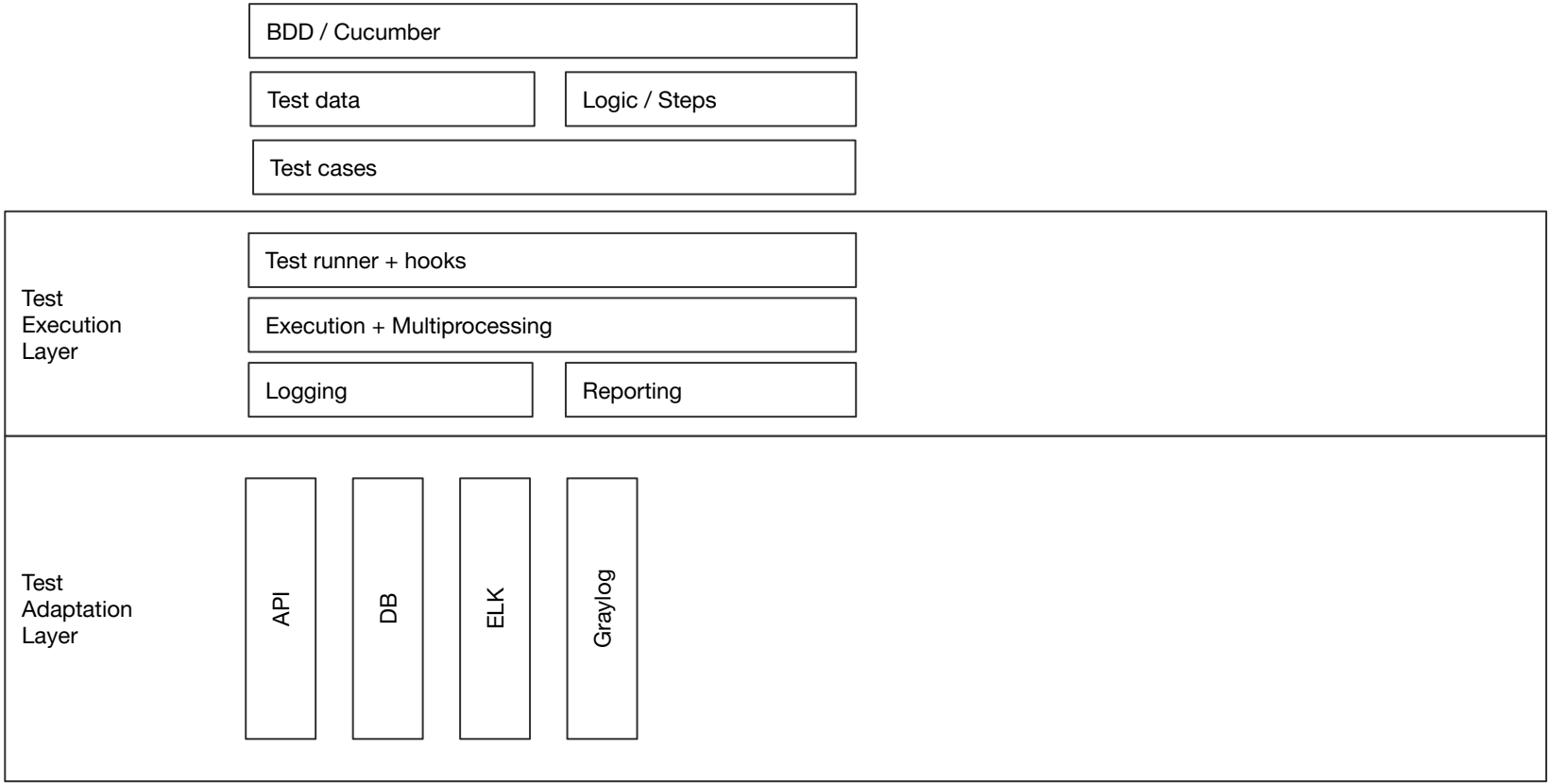
Test Definition Layer



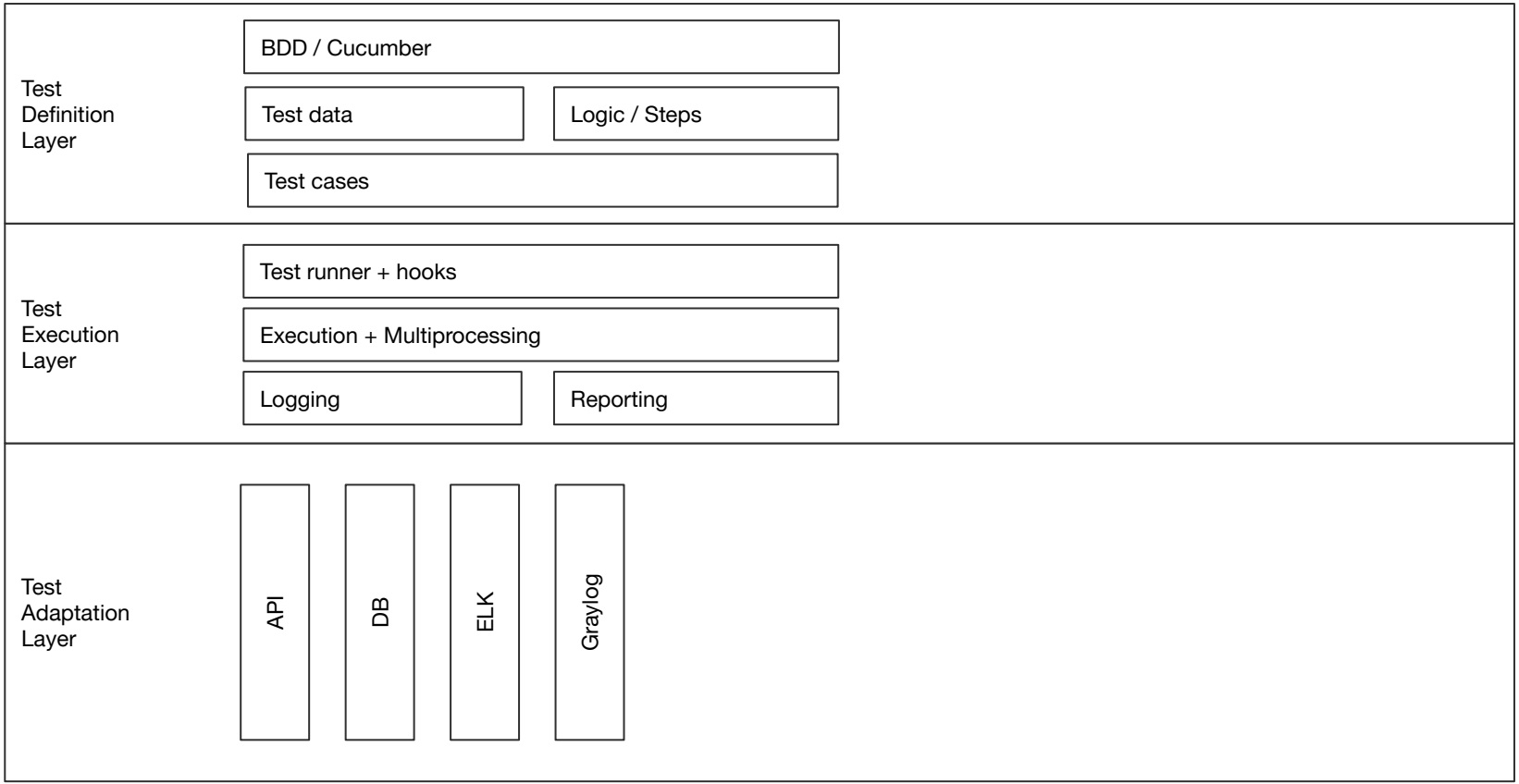
Test Definition Layer



Test Definition Layer



Test Definition Layer

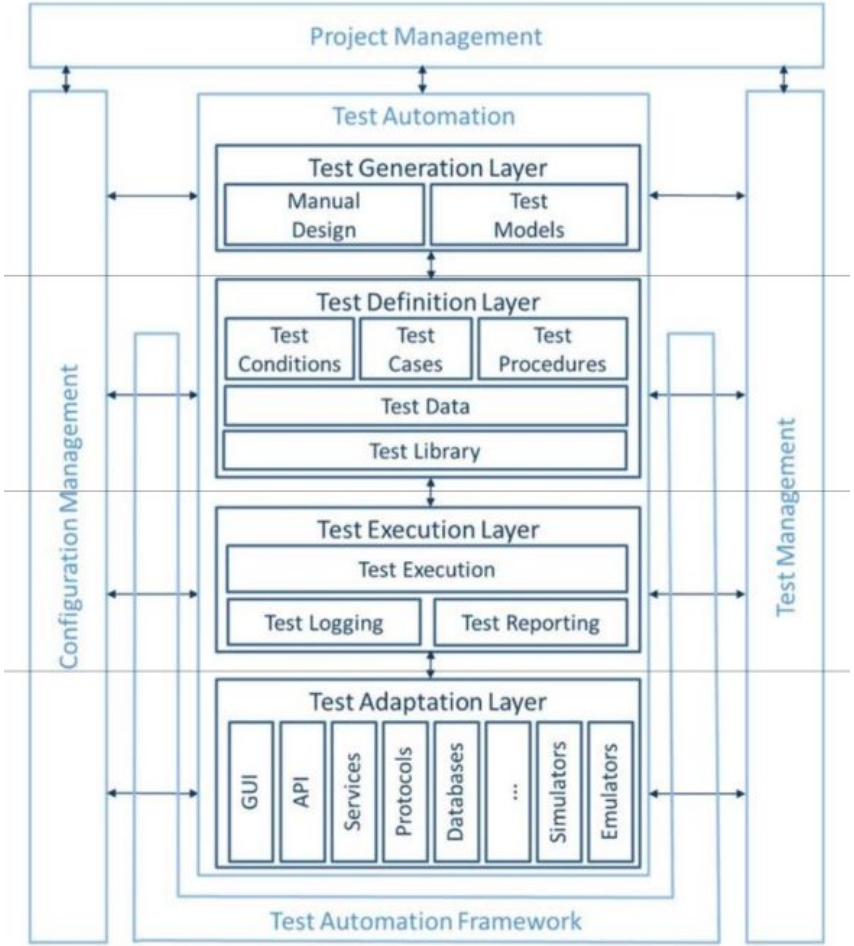


Test Definition Layer

- написание тестов
- организация тестов
- теги
- метки

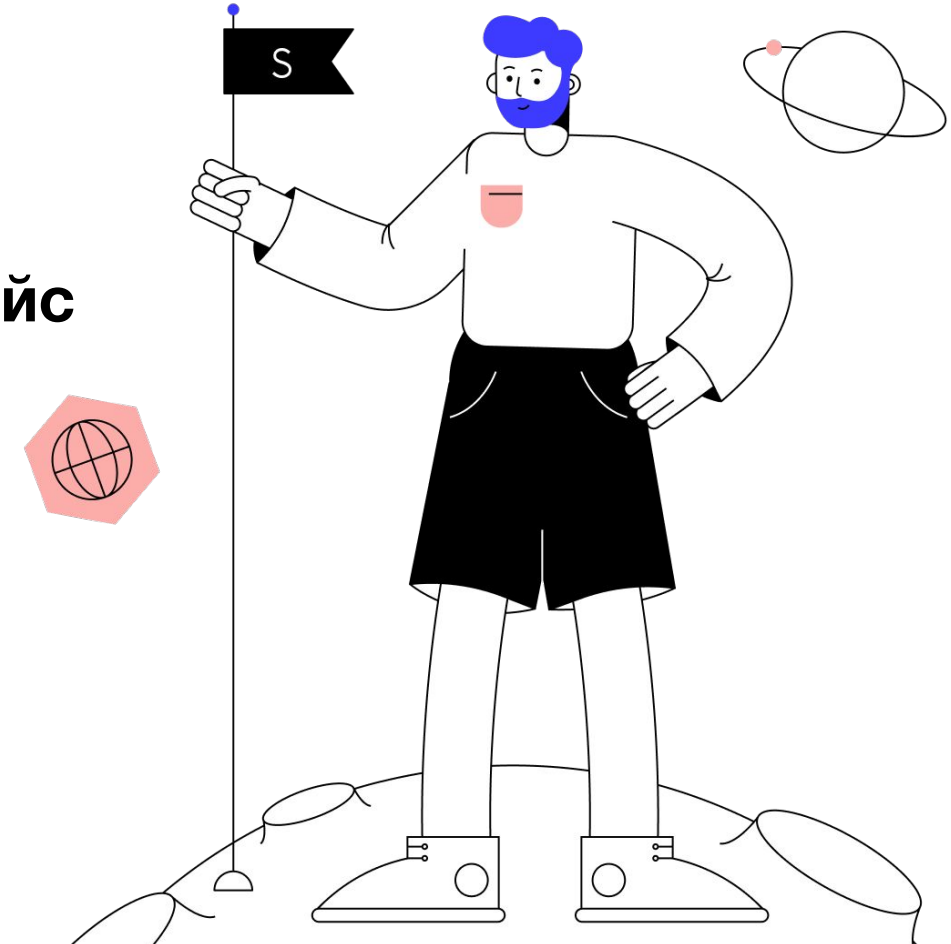
ISTQB gTAA

Уже не так страшно, не правда ли?



План-ураган

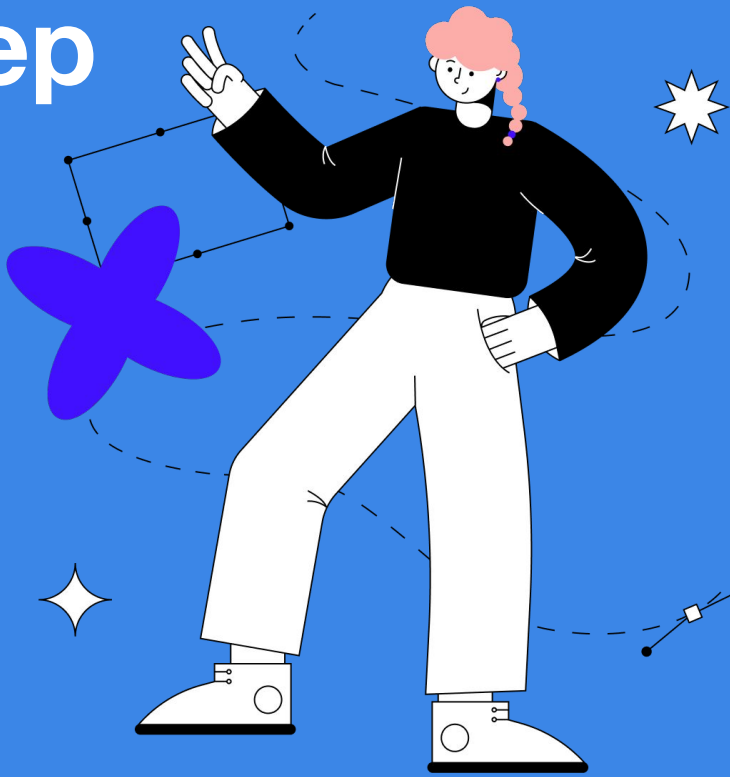
1. Тех.долг растет
2. ISTQB Arch
3. **Препарируем тест-кейс**
4. Задачки из жизни
5. Итоги



Хватит теории, давай истории

Жизненный пример

API+UI тест



Закрыть доступ к курсу

1. Создать студента (API)
2. Создать курс (API)
3. Выдать доступ к курсу (API)
4. Закрыть курс (UI)
5. Assert (API)

```
Feature: Остановка доступа к профессии

    Background:
        Given API LMS: создать профессию
            | var_name       | value                          |
            | name           | ${prof_name}                   |
            | slug           | ${slug}                        |
            | status         | published                      |
            | specialization | QA PRESETUP DEFAULT            |

        Given API LMS: создать пользователя
            | key        | value         |
            | email      | ${stud_email} |
            | first_name | ${stud_fname} |
            | last_name  | ${stud_lname} |
            | password   | ${stud_pass}  |

        Given API LMS: студенту "${stud_email}" выдан доступ к профессии "${prof_name}"
```

```
25
26      Scenario: Приостановка доступа к профессии
27
28          # UI actions
29
30          Given пользователь с ролью support открывает страницу Выдача доступов. Профессии
31          When пользователь вводит "${stud_email}" в поле "Поле поиска"
32          Then таблица "Таблица доступов" обновилась
33
34          Then открыта страница "Выдача доступов. Профессии. Создание доступа"
35          When пользователь нажимает на элемент "Приостановить"
36          When пользователь нажимает на элемент "Модальное окно > Подтвердить"
37          Then на странице не виден элемент "Модальное окно"
38
39          # API assert
40
41          Then API LMS: аккаунту не выдан доступ к профессии
42              | key              | value         |
43              | email            | ${stud_email} |
44              | student_password | ${stud_pass}  |
45              | profession_name  | ${prof_name}  |
```

```python
@staticmethod
@given("API LMS: создать профессию")
def creating_profession_lms(context):
    """Шаг создает профессию.

                    Given API LMS: создать профессию
                    | var_name  | value            |
                    | name      | Force профессия  |
                    | slug      | profession_slug  |
                    | teardown  | false            |
                    | force     | true             |

    """

    prof_data = parse_table_params(context.table, python_value=True)

    name = prof_data.pop("name")
    teardown = prof_data.pop("teardown", True)

    api_client = admin_user(context)
    prof_api = StaffProfessionsLogic(api_client)

    if specialization := prof_data.pop("specialization", None):
        specialization_id = LMSSpecializationsLogic(api_client).get_specialization_id(specialization)
        prof_data.update({"specialization_id": specialization_id})

    prof_info = prof_api.create_profession(name=name, **prof_data)
```

```python
@staticmethod
@given("API LMS: создать профессию")
def creating_profession_lms(context):
    """Шаг создает профессию.

                    Given API LMS: создать профессию
                    | var_name  | value            |
                    | name      | Force профессия  |
                    | slug      | profession_slug  |
                    | teardown  | false            |
                    | force     | true             |

    """

    prof_data = parse_table_params(context.table, python_value=True)

    name = prof_data.pop("name")
    teardown = prof_data.pop("teardown", True)

    api_client = admin_user(context)
    prof_api = StaffProfessionsLogic(api_client)

    if specialization := prof_data.pop("specialization", None):
        specialization_id = LMSSpecializationsLogic(api_client).get_specialization_id(specialization)
        prof_data.update({"specialization_id": specialization_id})

    prof_info = prof_api.create_profession(name=name, **prof_data)
```

```python
@staticmethod
@given("API LMS: создать профессию")
def creating_profession_lms(context):
    """Шаг создает профессию.

                    Given API LMS: создать профессию
                    | var_name  | value            |
                    | name      | Force профессия  |
                    | slug      | profession_slug  |
                    | teardown  | false            |
                    | force     | true             |

    """

    prof_data = parse_table_params(context.table, python_value=True)

    name = prof_data.pop("name")
    teardown = prof_data.pop("teardown", True)

    api_client = admin_user(context)
    prof_api = StaffProfessionsLogic(api_client)

    if specialization := prof_data.pop("specialization", None):
        specialization_id = LMSSpecializationsLogic(api_client).get_specialization_id(specialization)
        prof_data.update({"specialization_id": specialization_id})

    prof_info = prof_api.create_profession(name=name, **prof_data)
```

```python
    @staticmethod
    @given("API LMS: создать профессию")
    def creating_profession_lms(context):
        """Шаг создает профессию.

                        Given API LMS: создать профессию
                        | var_name  | value            |
                        | name      | Force профессия  |
                        | slug      | profession_slug  |
                        | teardown  | false            |
                        | force     | true             |

        """

        prof_data = parse_table_params(context.table, python_value=True)

        name = prof_data.pop("name")
        teardown = prof_data.pop("teardown", True)

        api_client = admin_user(context)
        prof_api = StaffProfessionsLogic(api_client)

        if specialization := prof_data.pop("specialization", None):
            specialization_id = LMSSpecializationsLogic(api_client).get_specialization_id(specialization)
            prof_data.update({"specialization_id": specialization_id})

        prof_info = prof_api.create_profession(name=name, **prof_data)
```

Тестовая логика =
новый **backend route**

```
9 usages
class StaffProfessionsLogic:
    """API методы работы с профессиями.

    /api/v3/staff/professions/
    """

    def __init__(self, client: LmsClient):
        self.api = StaffProfessionsApi(client)

    2 usages
    def create_profession(self, **payload) -> dict:
        """Создать профессию.

        :param payload: параметры профессии
        """

        response = self.api.post_profession(**payload)
        check_that(
            lambda: response.ok,
            ProfessionException,
            message: f"Не удалось создать профессию. {response.status_code}, {response.text}",
        )

        return response.json()
```

```
1 usage
class StaffProfessionsApi(LMSApi):
    """API профессий.

    /api/v3/staff/professions/
    """

    1 usage
    def post_profession(self, **payload) -> Response:
        """Создать профессию.

        :param payload: параметры профессии
        """

        return self.client.post( *args: "/api/v3/staff/professions/", json=payload)
```

```
1 usage
class StaffProfessionsApi(LMSApi):
    """API профессий.

    /api/v3/staff/professions/
    """

    1 usage
    def post_profession(self, **payload) -> Response:
        """Создать профессию.

        :param payload: параметры профессии
        """

        return self.client.post( *args: "/api/v3/staff/professions/", json=payload)
```

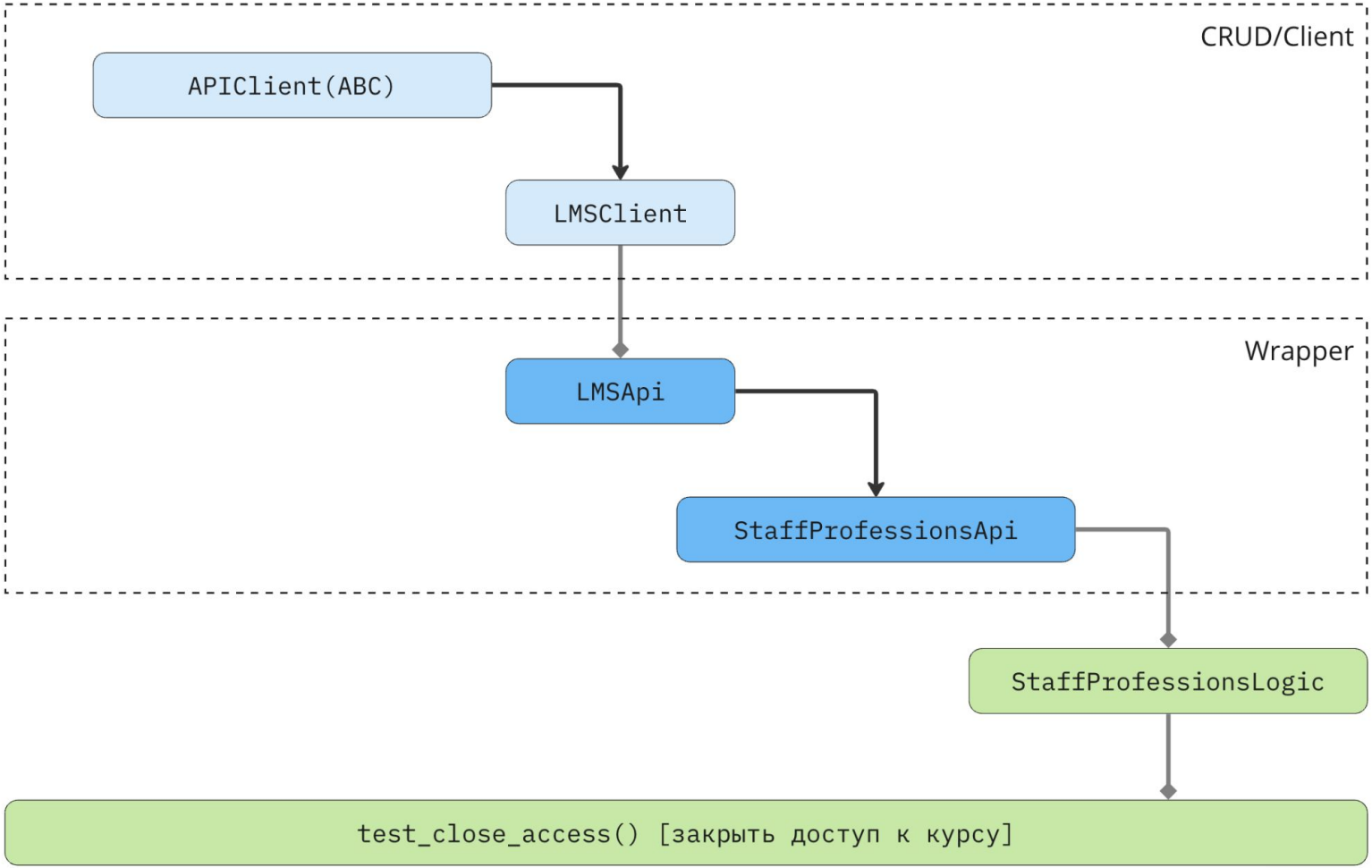
```python
class LMSApi:
    """API методы LMS."""

    def __init__(self, client: LmsClient):
        self.client = client

    def basic_auth(self, basic_login, basic_password):...

    def auth_user(self, email, password):...

    def upload_files(self, files_str: str, endpoint: str):...

    1 usage
    def get_front_versions(self):...

    1 usage
    def get_back_versions(self):...

    1 usage
    def get_lms_front_version(self):
        """Метод получает версию фронта LMS на текущем стенде."""
```

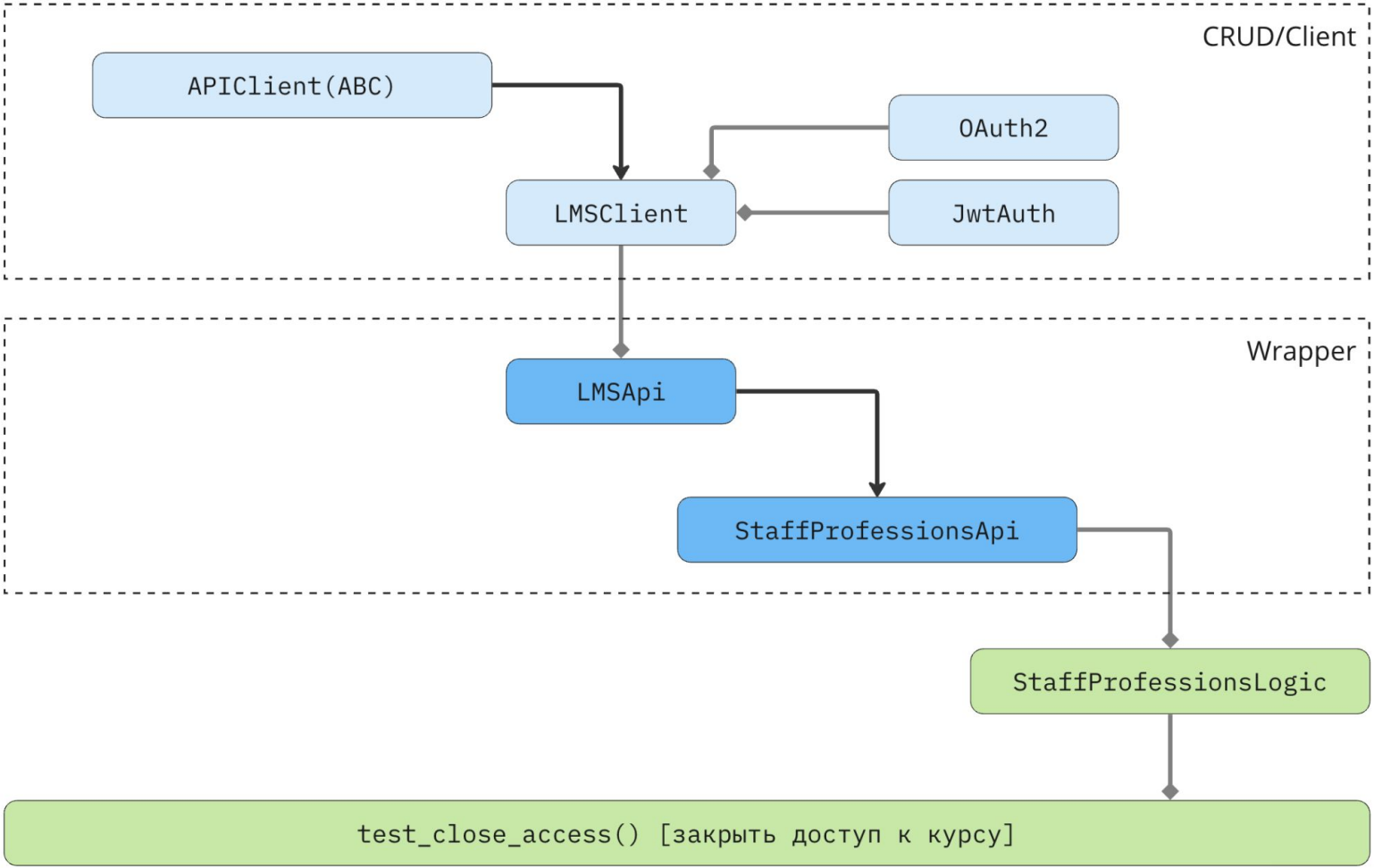
Держитесь,
еще чуть-чуть…

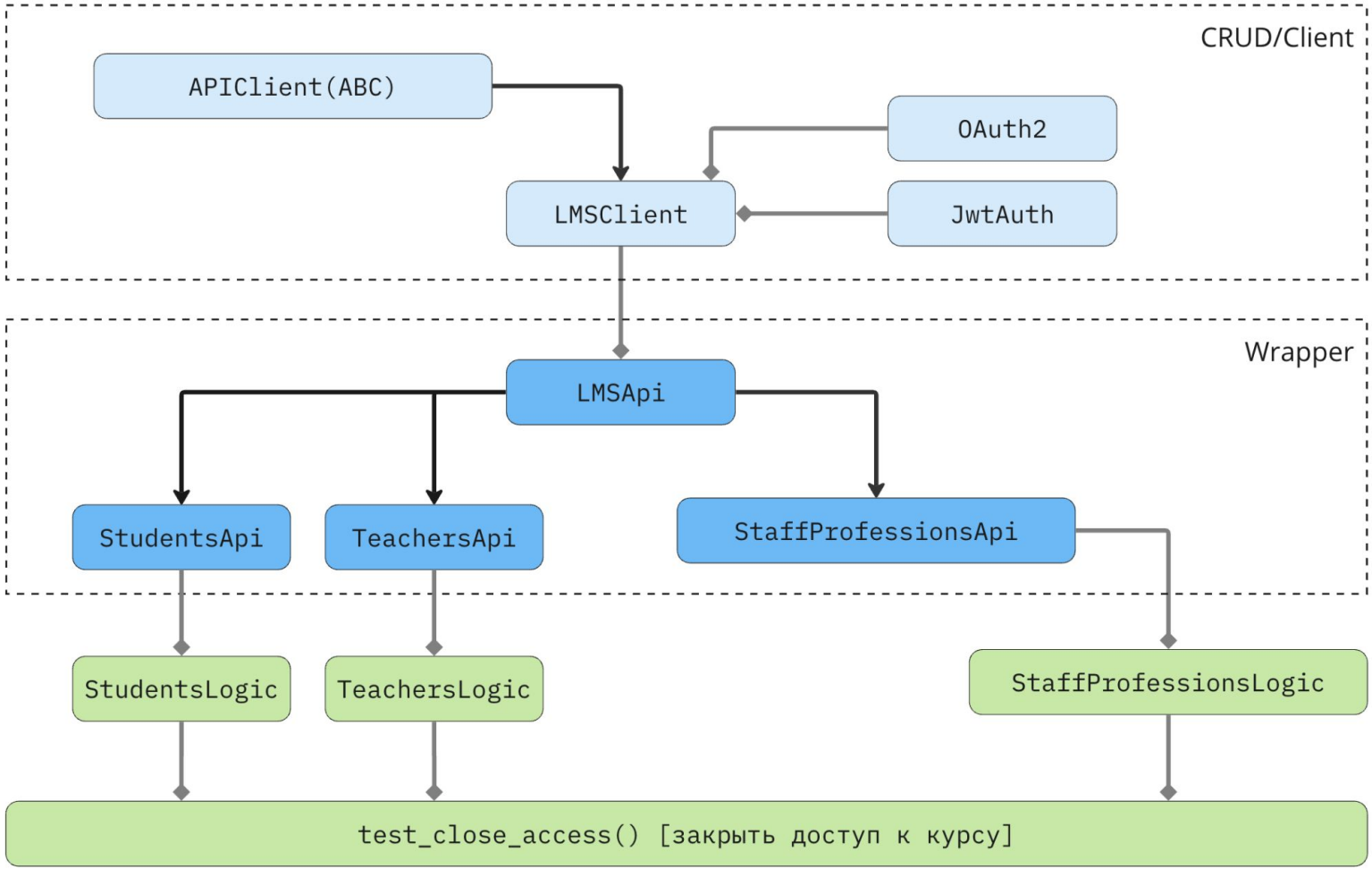
```python
class LmsClient(ApiClient):
    """Клиент для отправки запросов к Skillbox LMS API."""

    def __init__(self, base_url: str):
        super().__init__(base_url)
        self.session = SessionWithBaseUrl(base_url)
        self._auth = LmsJwtAuth()

    def auth(self, email: str, password: str) -> None:
        """Авторизуем пользователя по учетным данным.

        :param email: email
        :param password: пароль
        """

        self.clear_session()
        self._auth.set_email(email)
        self._auth.set_password(password)
        self._auth.set_session(self.session)
```

```python
# конечная, поезд дальше не идет

class ApiClient(ABC):
    _base_url: ...
    _session: ...
    _default_headers: ...
    _last_response: ...

    def __init__(self, base_url: ...):
        self._base_url = base_url
        self._session = Session()
        self.set_default_headers()

    def send(self, method: ..., endpoint: ..., *args, **kwargs) -> ...:...

    def get(self, *args, **kwargs):...

    def post(self, *args, **kwargs):...
```

UML
Class Diagram

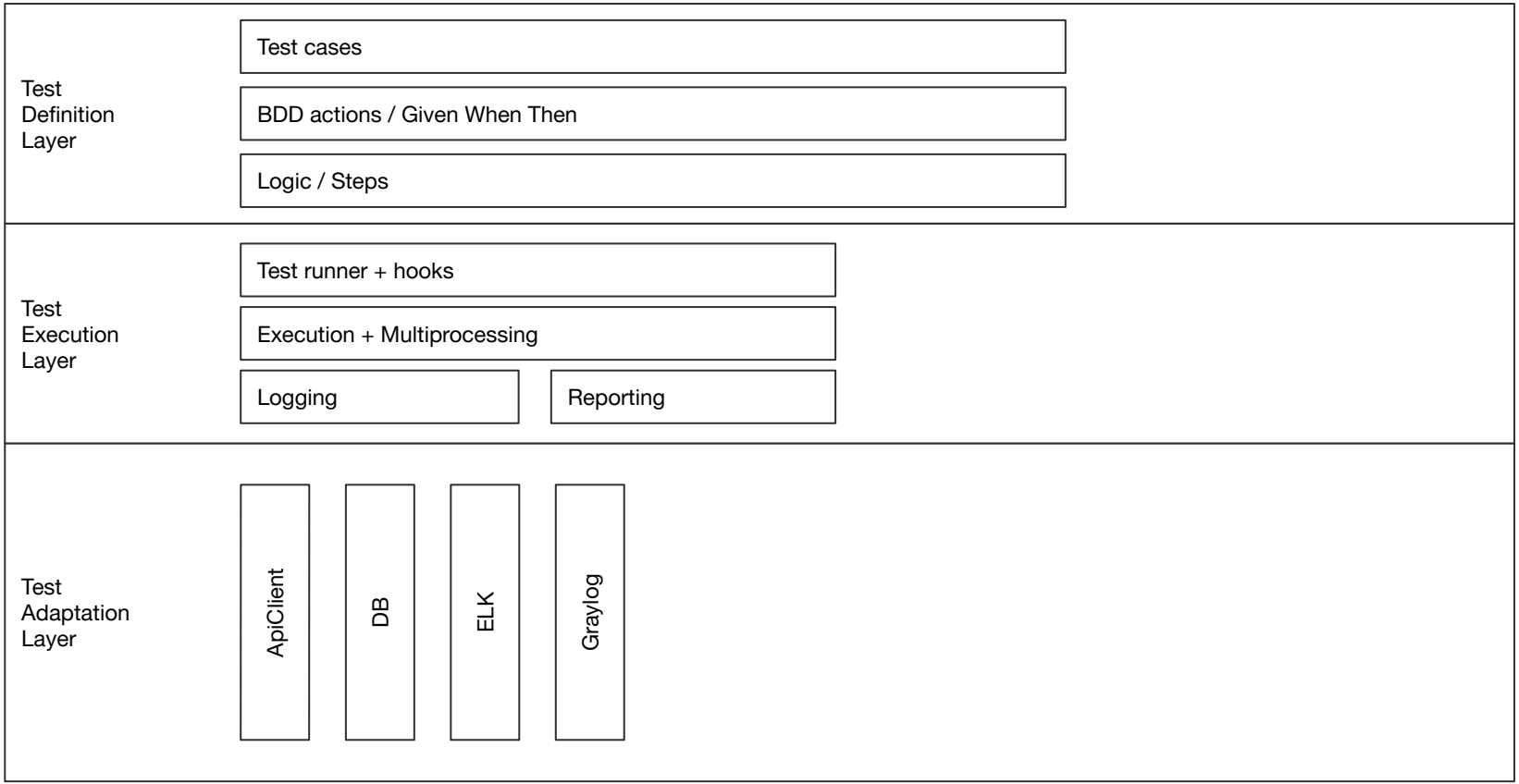

CRUD/Client
APIClient(ABC)
LMSClient
Wrapper
LMSApi
StaffProfessionsApi
StaffProfessionsLogic
test_close_access() [закрыть доступ к курсу]


CRUD/Client
APIClient(ABC)
OAuth2
LMSClient
JwtAuth
Wrapper
LMSApi
StaffProfessionsApi
StaffProfessionsLogic
test_close_access() [закрыть доступ к курсу]


CRUD/Client
APIClient(ABC)
OAuth2
LMSClient
JwtAuth
Wrapper
LMSApi
StudentsApi
TeachersApi
StaffProfessionsApi
StudentsLogic
TeachersLogic
StaffProfessionsLogic
test_close_access() [закрыть доступ к курсу]

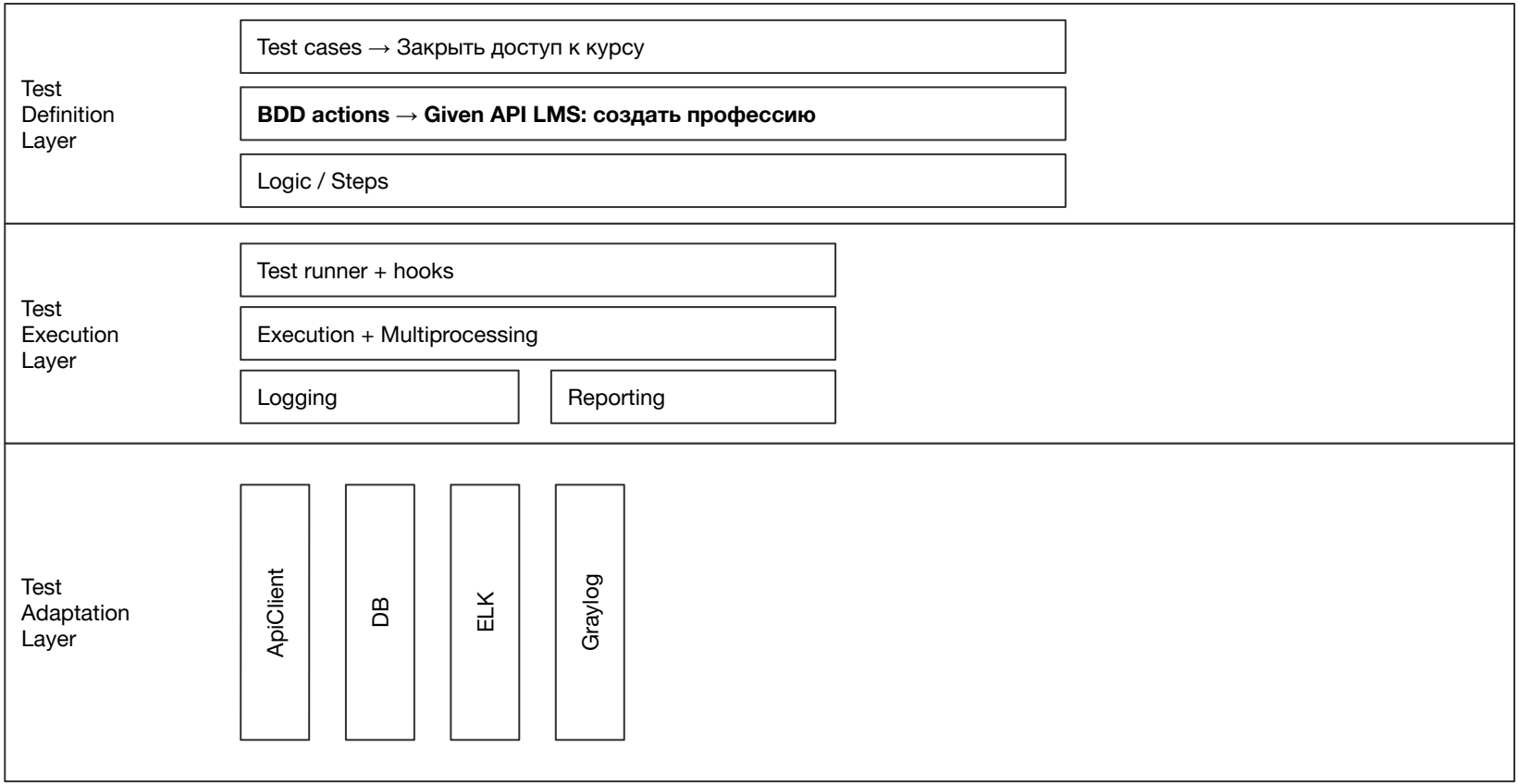
Как это ложится на схему ISTQB

Исходная схема

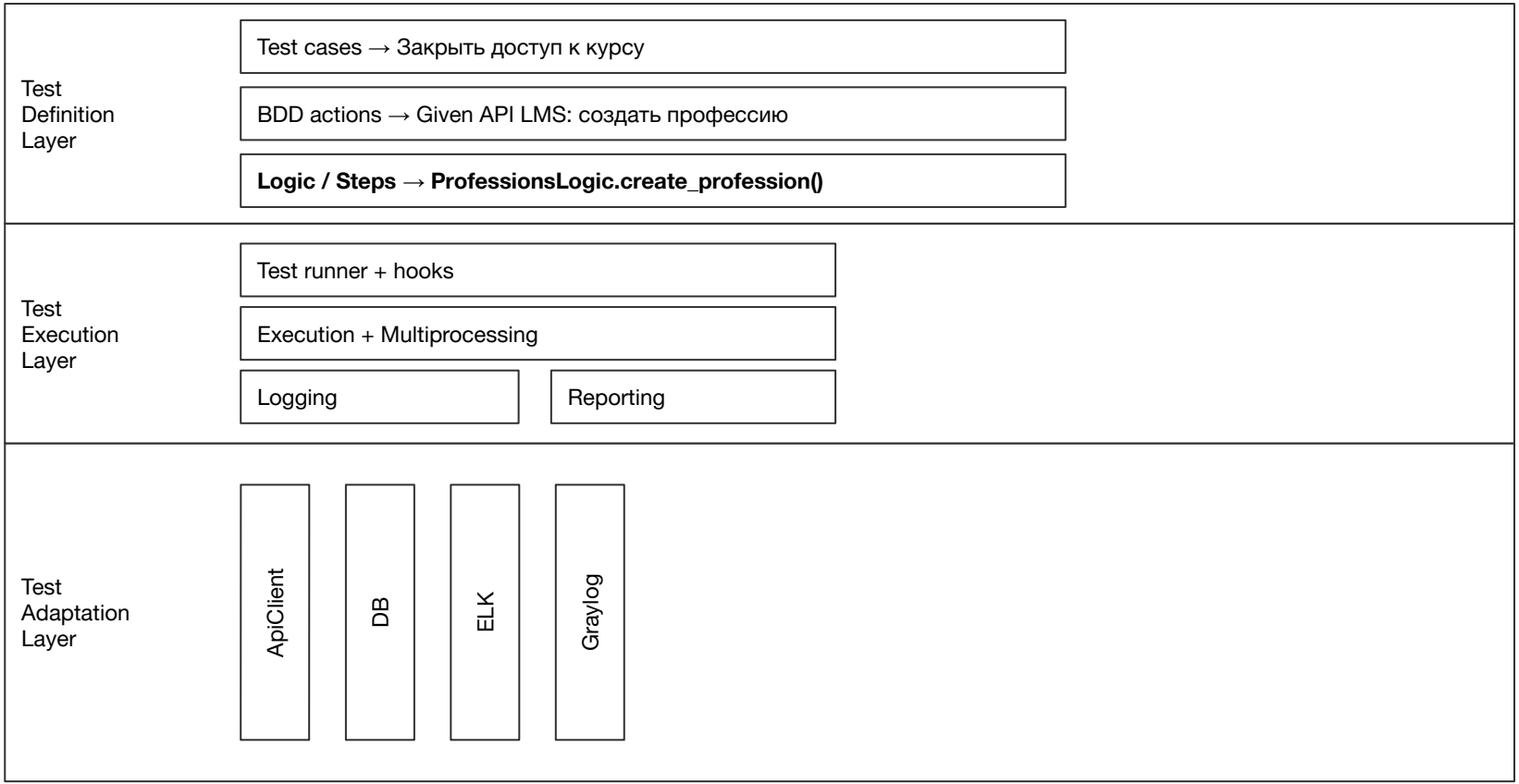


Исходная схема

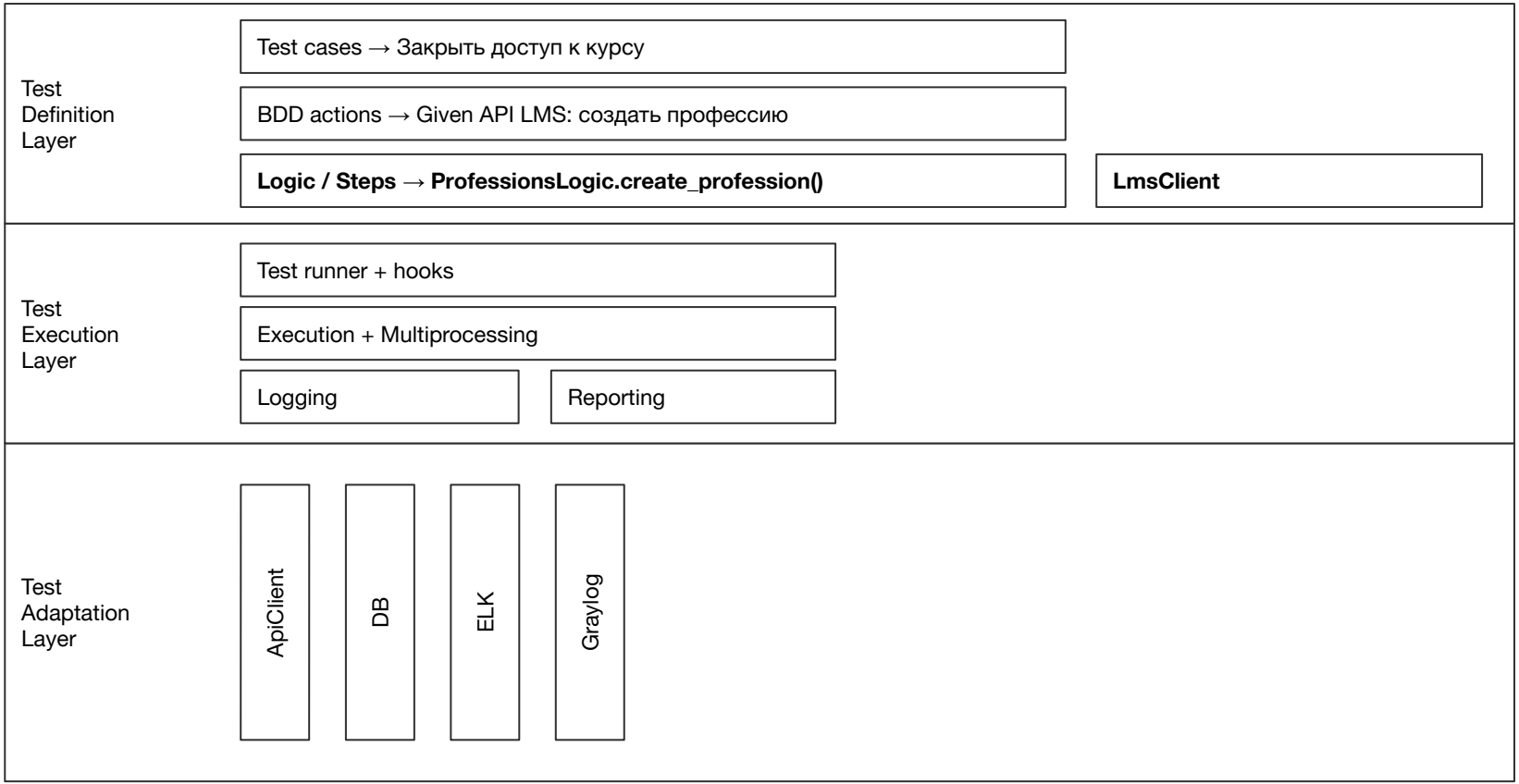
Исходная схема



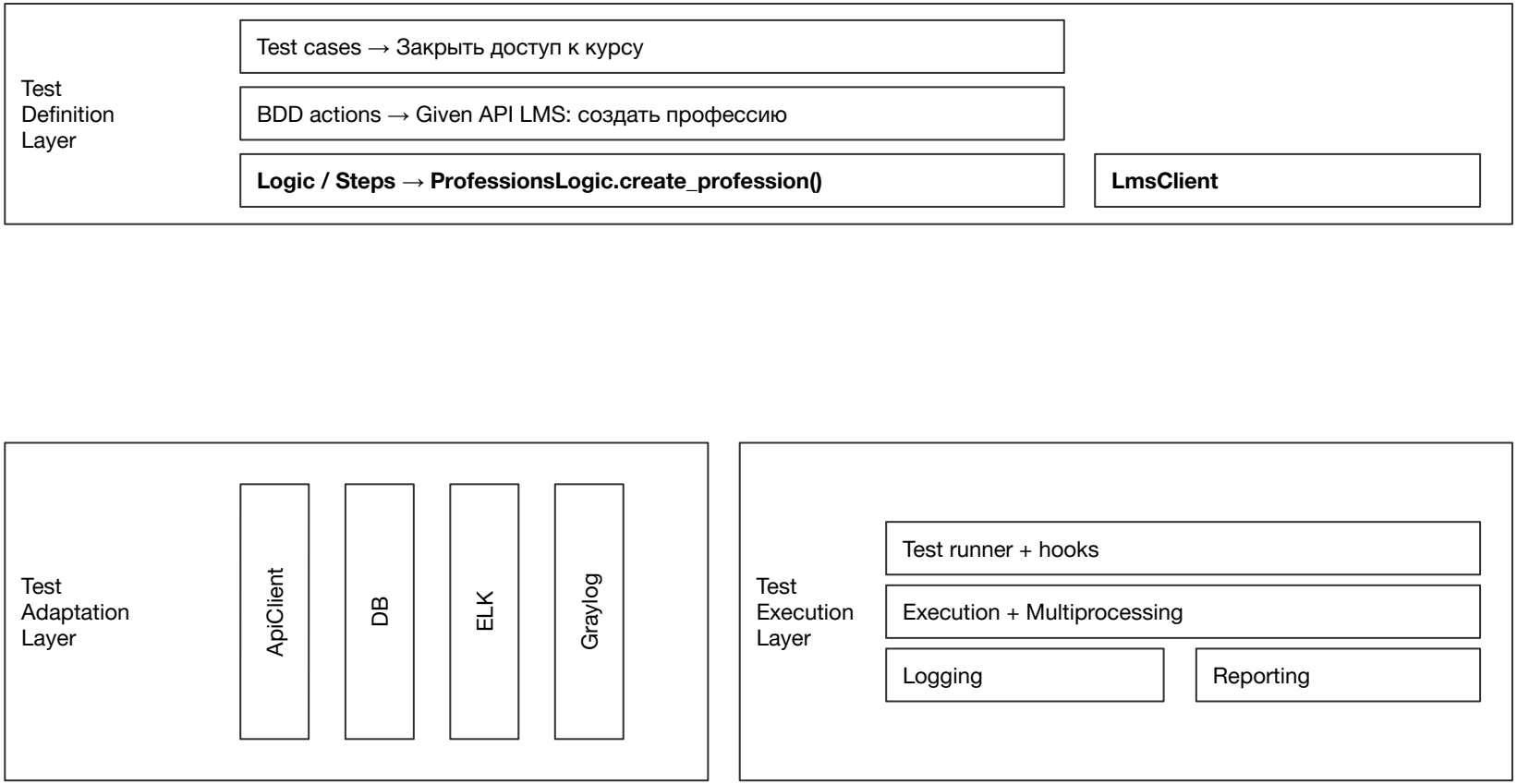
Исходная схема



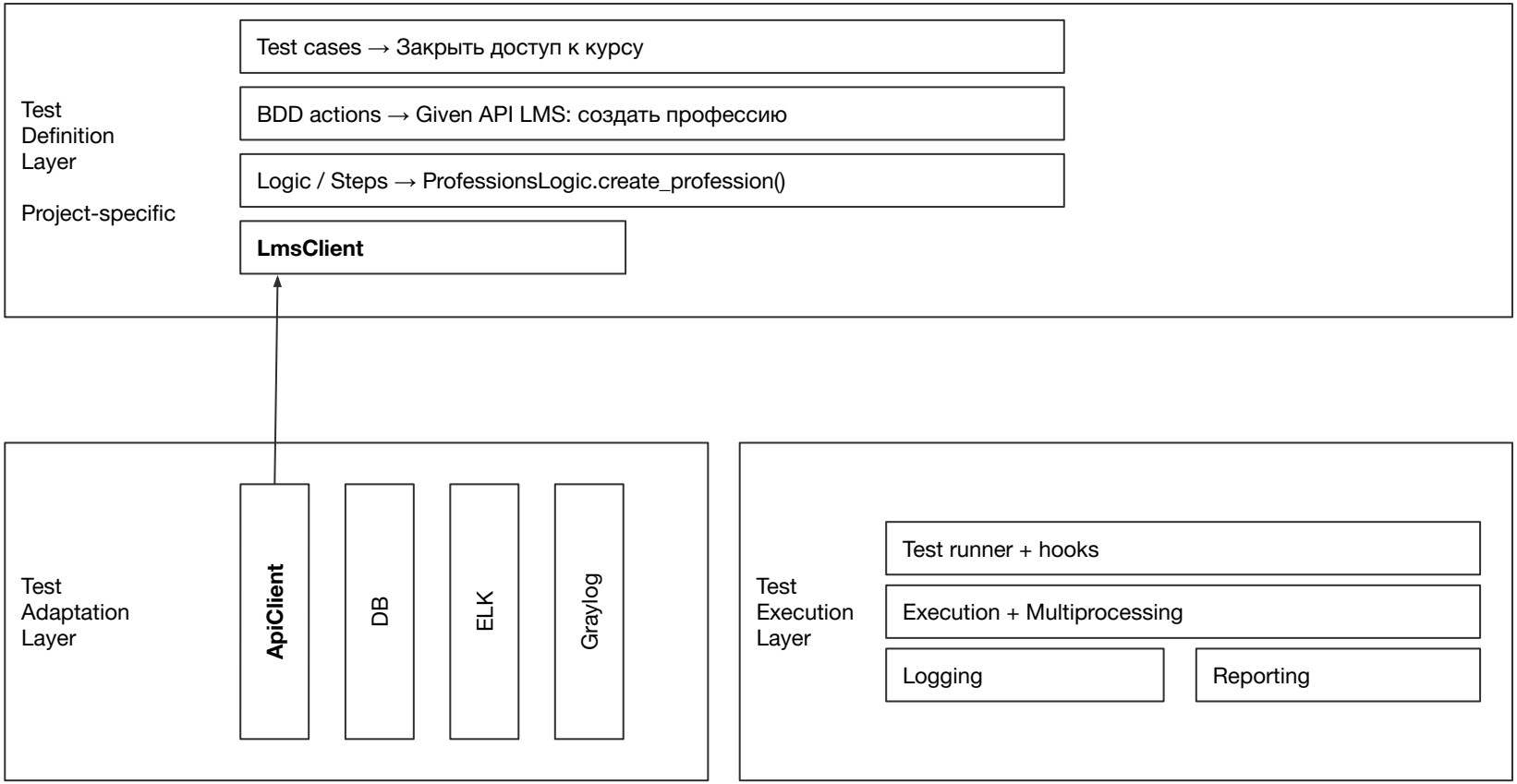
Исходная схема



Исходная схема



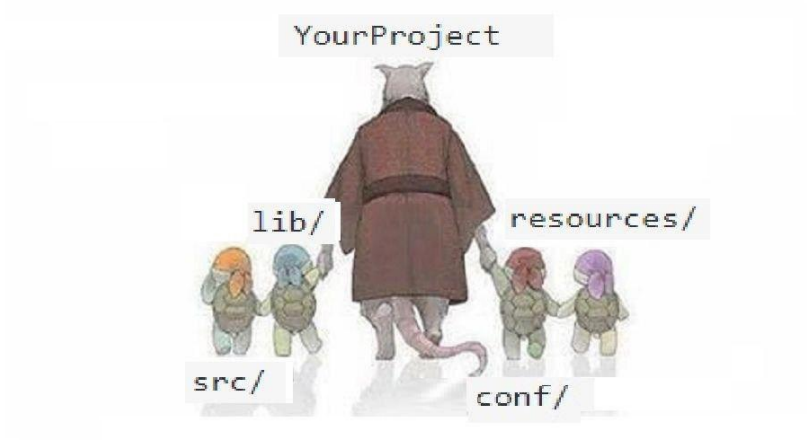
Исходная схема



Структура проекта

Мысль

Структура проекта
не будет точь-в-точь
повторять архитектурную
схему


YourProject
lib/
resources/
src/
conf/


шизолегаси код
неиспользуемые библиотеки
конфиги без документации
рандомные файлы
YourProject

Структура проекта

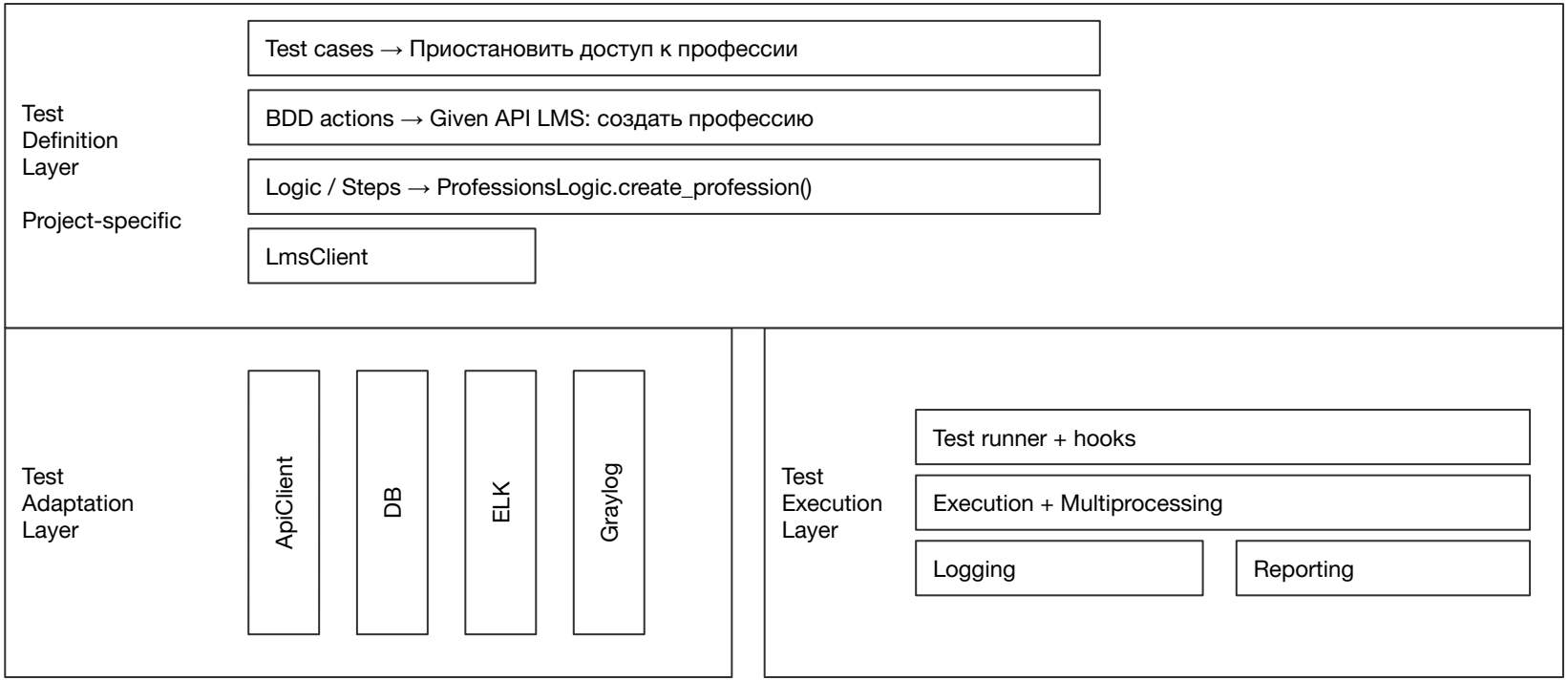
```
├── core
│   ├── backend
│   │   ├── client.py → ApiClient(ABC)
│   │   ├── database_client.py
│   ├── exceptions.py
```

Структура проекта

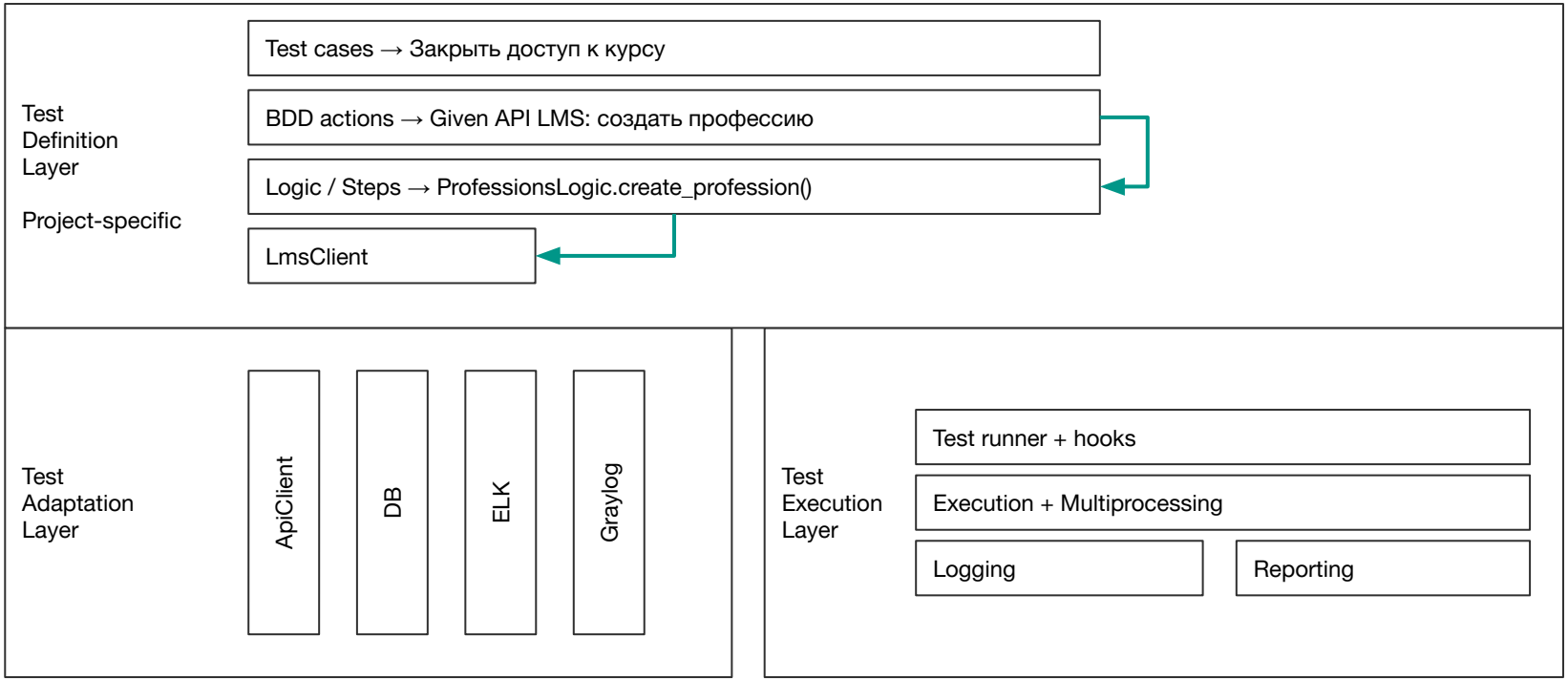
```
├── lms
│   ├── client.py → LMSClient(ApiClient)
│   ├── exceptions.py
│   ├── api
│   │   ├── common_api.py → LMSApi
│   │   ├── professions_api.py → ProfessionsApi(LMSApi)
│   │   ├── students_api.py
│   │   ├── teachers_api.py
│   ├── steps
│       ├── generic_steps.py
│       ├── profession_steps.py → ProfessionsLogic
│       ├── registration_steps.py
│       ├── students
│       │   ├── lessons_steps.py
│       │   └── p2p_tasks_steps.py
│       ├── teachers
│       │   └── user_homeworks_steps.py
```

Связи между уровнями / слоями

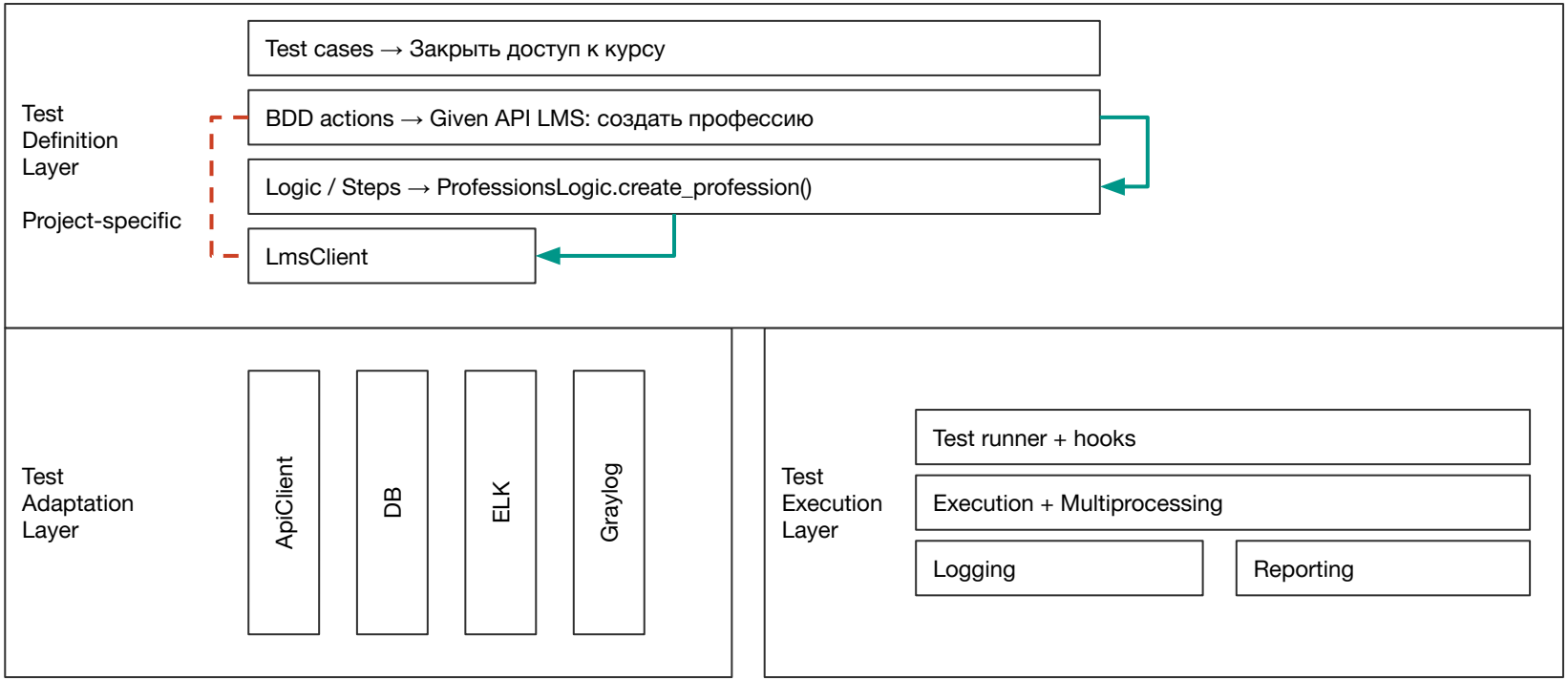
Связи между уровнями



Связи между уровнями

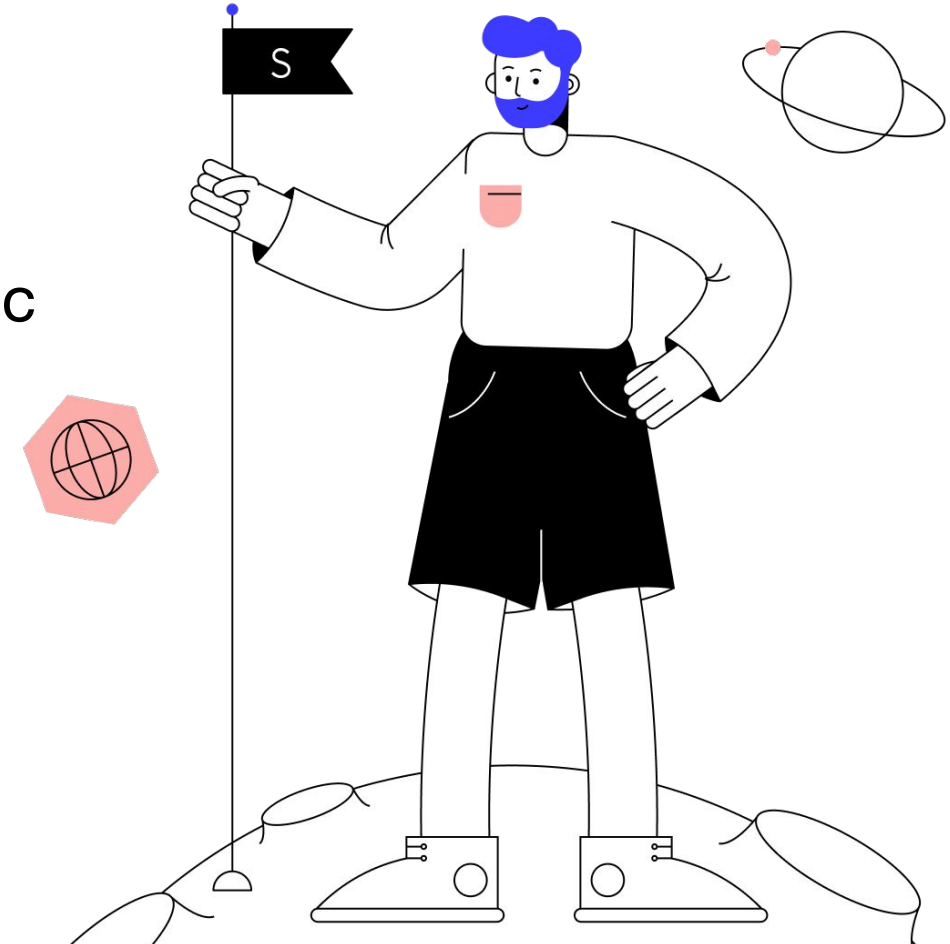


Связи между уровнями



План-ураган

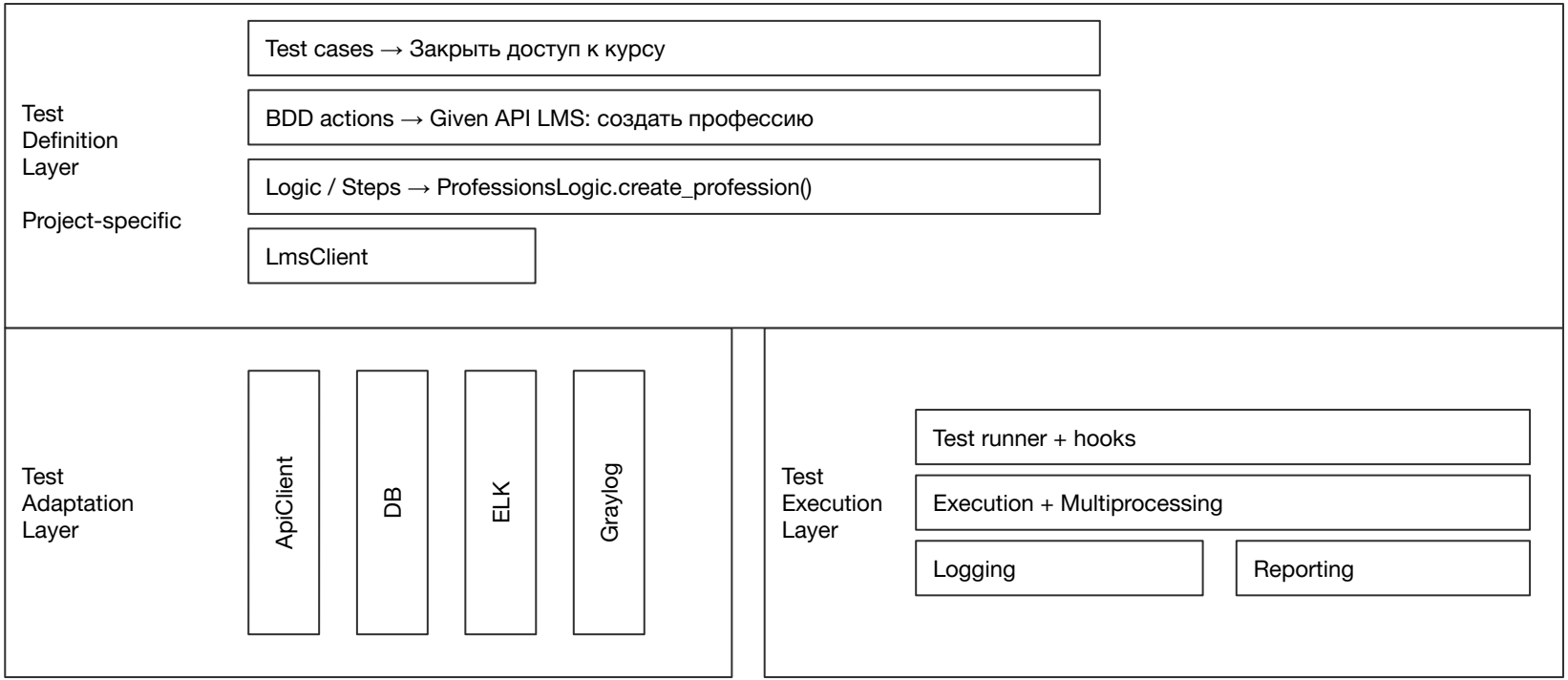
1. Тех.долг растет
2. ISTQB Arch
3. Препарируем тест-кейс
4. **Задачки из жизни**
5. Итоги



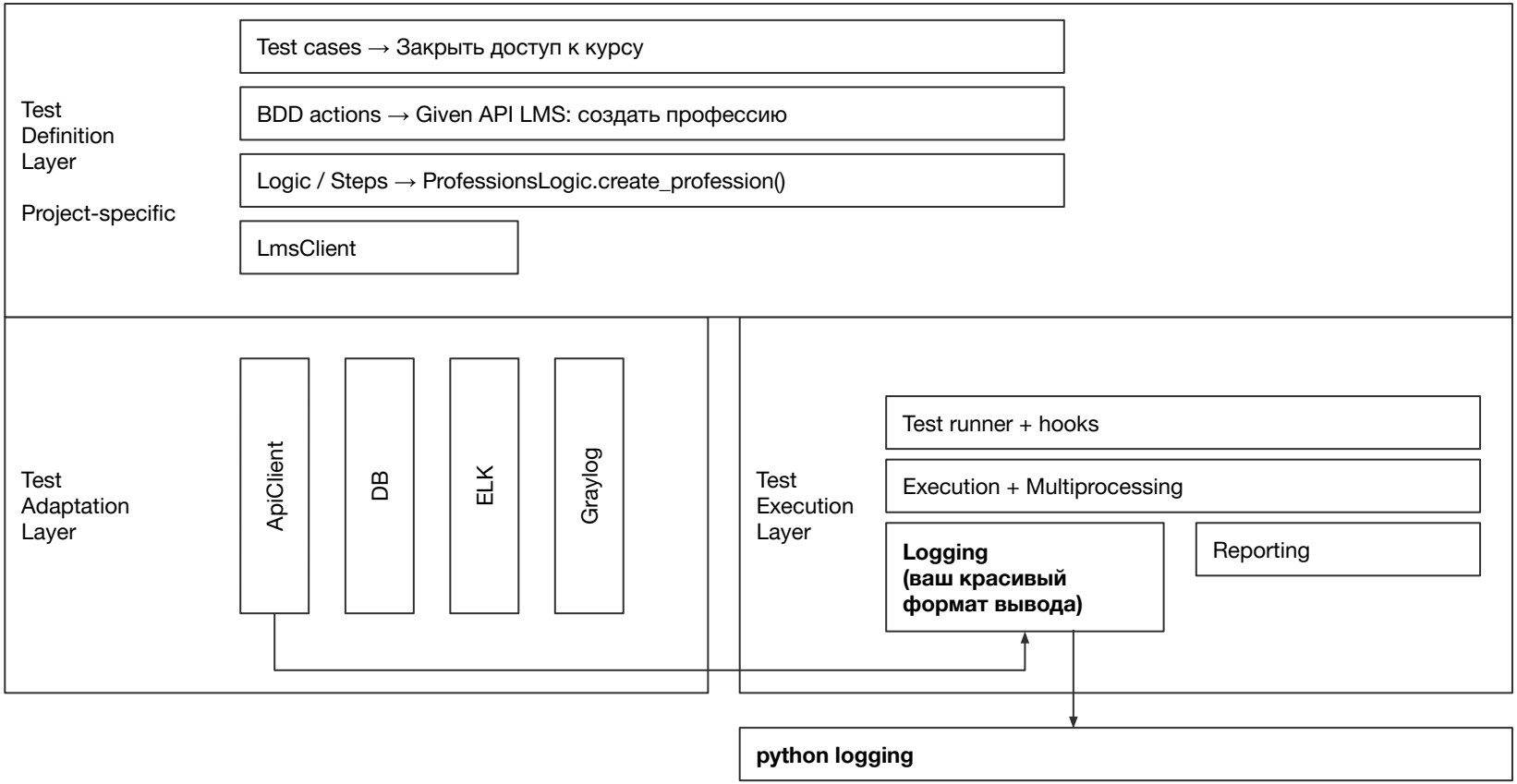
Задачка

Как залогировать API запрос и ответ

Залогировать API запрос/ответ



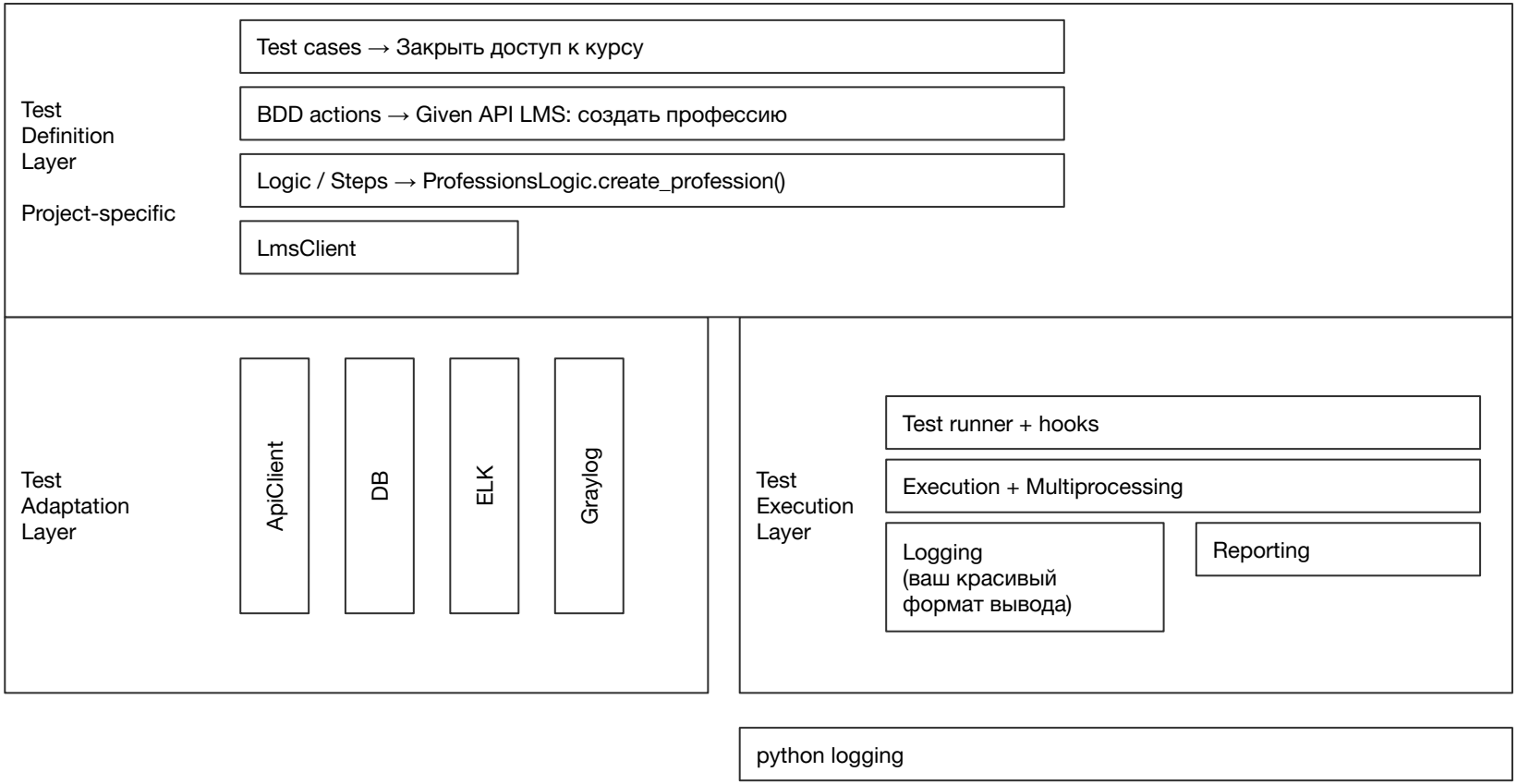
Залогировать API запрос/ответ



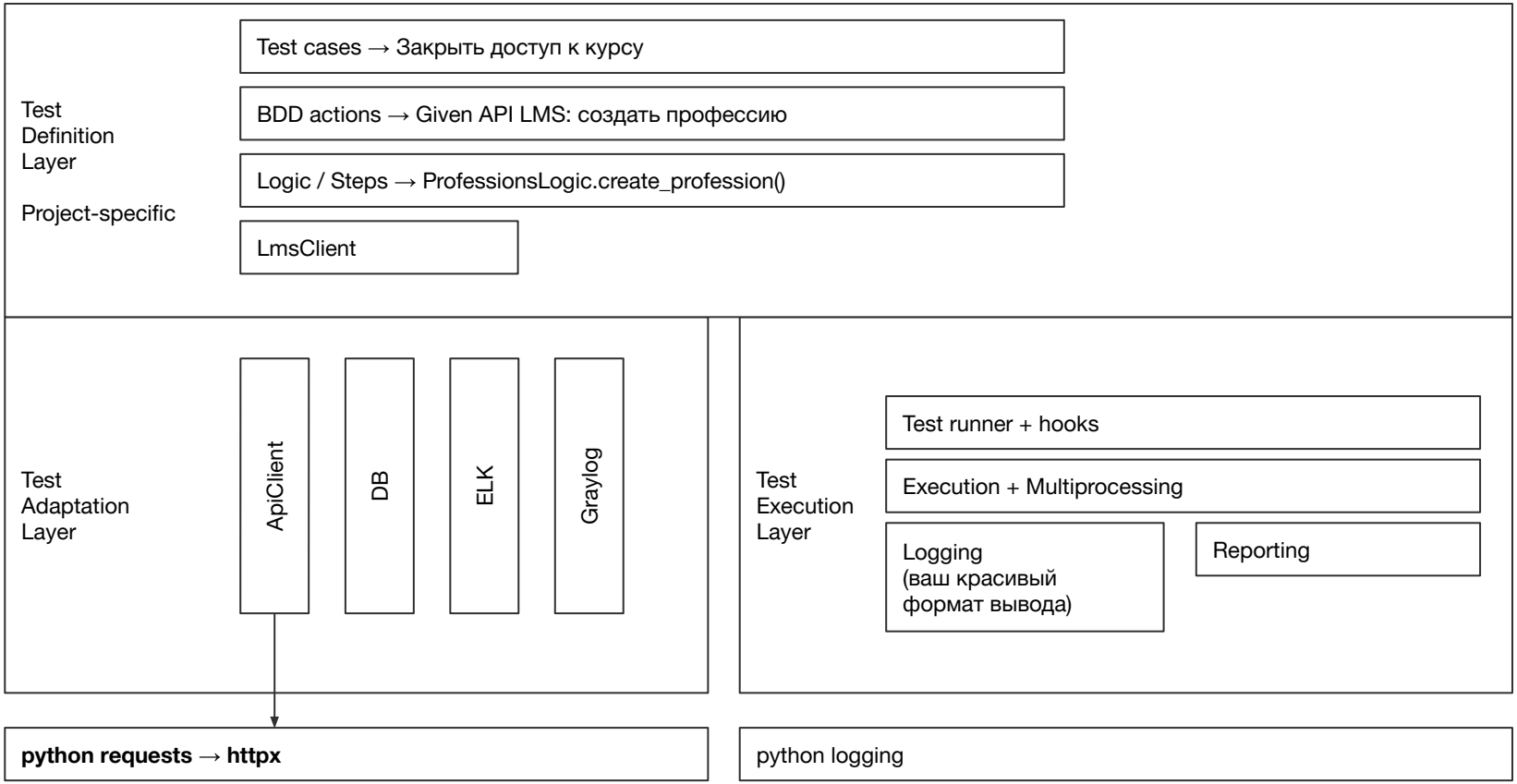
Задачка

Как поменять библиотеку HTTP

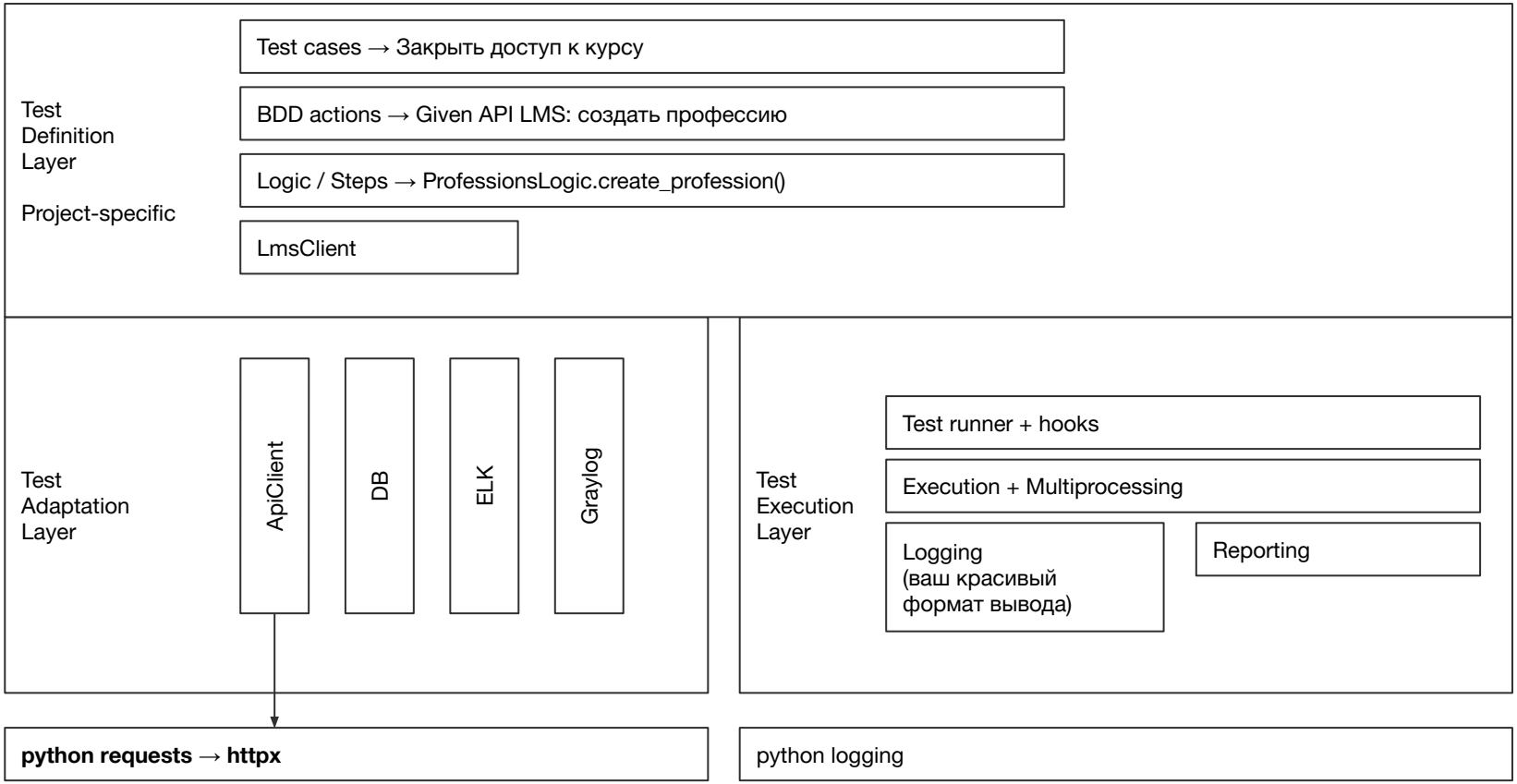
Поменять библиотеку



Поменять библиотеку



Поменять библиотеку



Так это же…
dependency inversion

```python
from requests import Session


class ApiClient(ABC):

    def __init__(self, base_url: str):
        self.base_url = base_url
        self.session = Session()

    @log_request_timings
    @retry(2)
    def send(self, *args, **kwargs):
        try:
            self.last_response = self.session.request(*args, **kwargs) # ⟵ change here
        except RequestException as e:
            self.log_error(e, *args, **kwargs)
            raise e

        return self.last_response


```

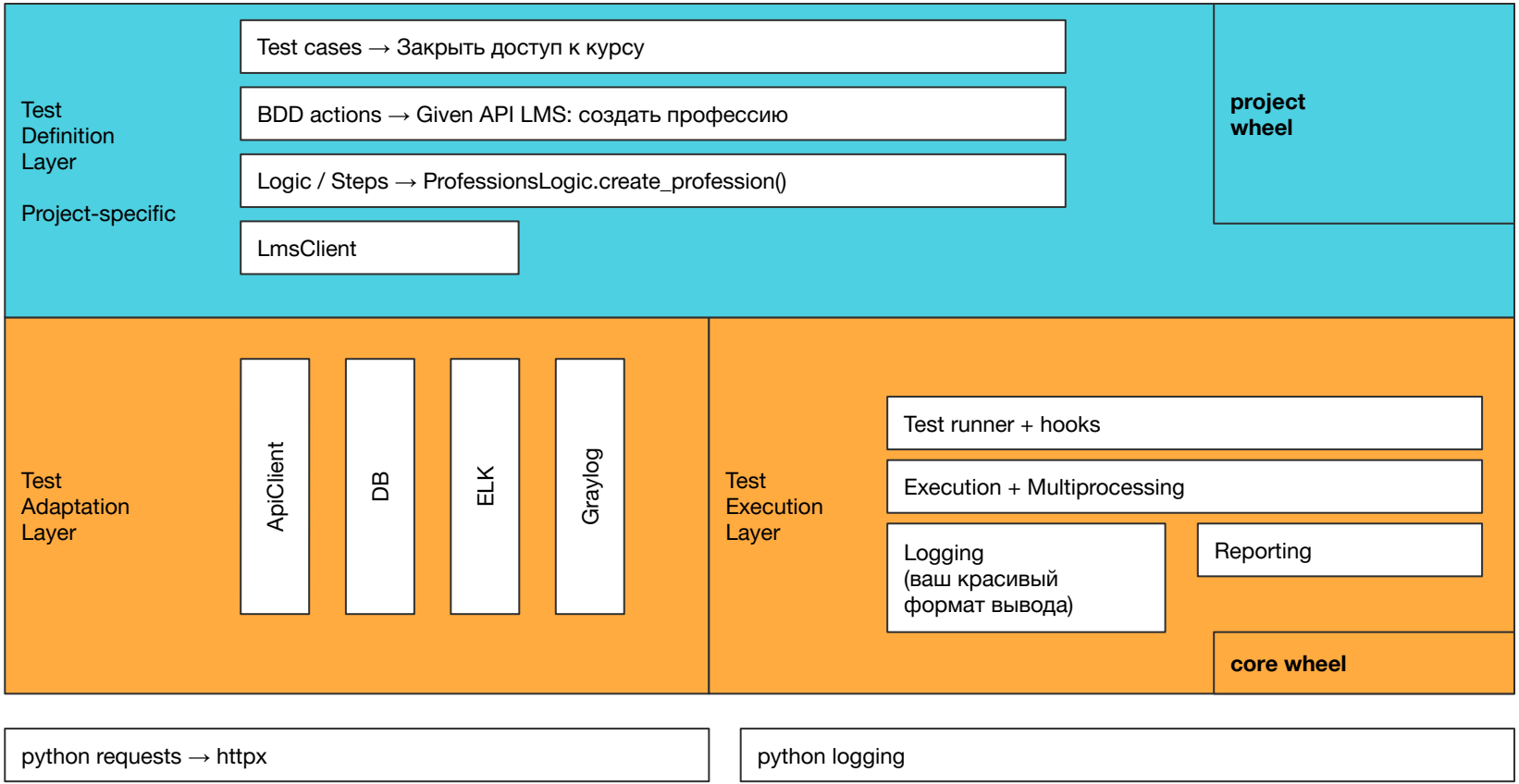
Dependency inversion

ISTQB: components should not depend on specific automated test scenarios.

Реализация тест-кейсов
зависит от абстракций,
а не конкретных библиотек.

Запаковать результат в библиотеку

Библиотека


Test
Definition
Layer
Project-specific
Test cases → Закрыть доступ к курсу
BDD actions → Given API LMS: создать профессию
Logic / Steps → ProfessionsLogic.create_profession()
LmsClient
project
wheel
Test
Adaptation
Layer
ApiClient
DB
ELK
Graylog
Test
Execution
Layer
Test runner + hooks
Execution + Multiprocessing
Logging
(ваш красивый
формат вывода)
Reporting
core wheel
python requests → httpx
python logging

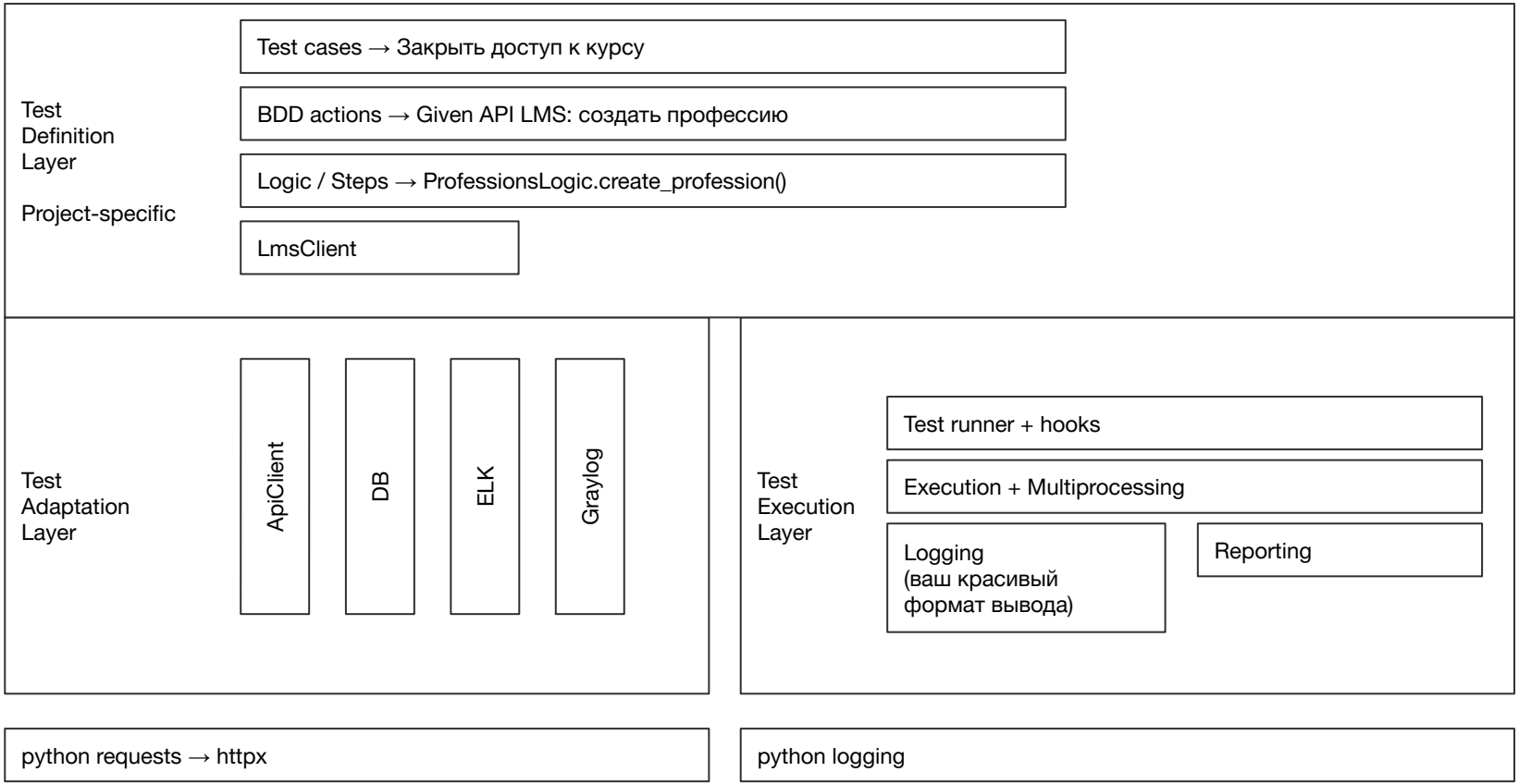
Задачка

Как добавить User-Agent header с номером теста?

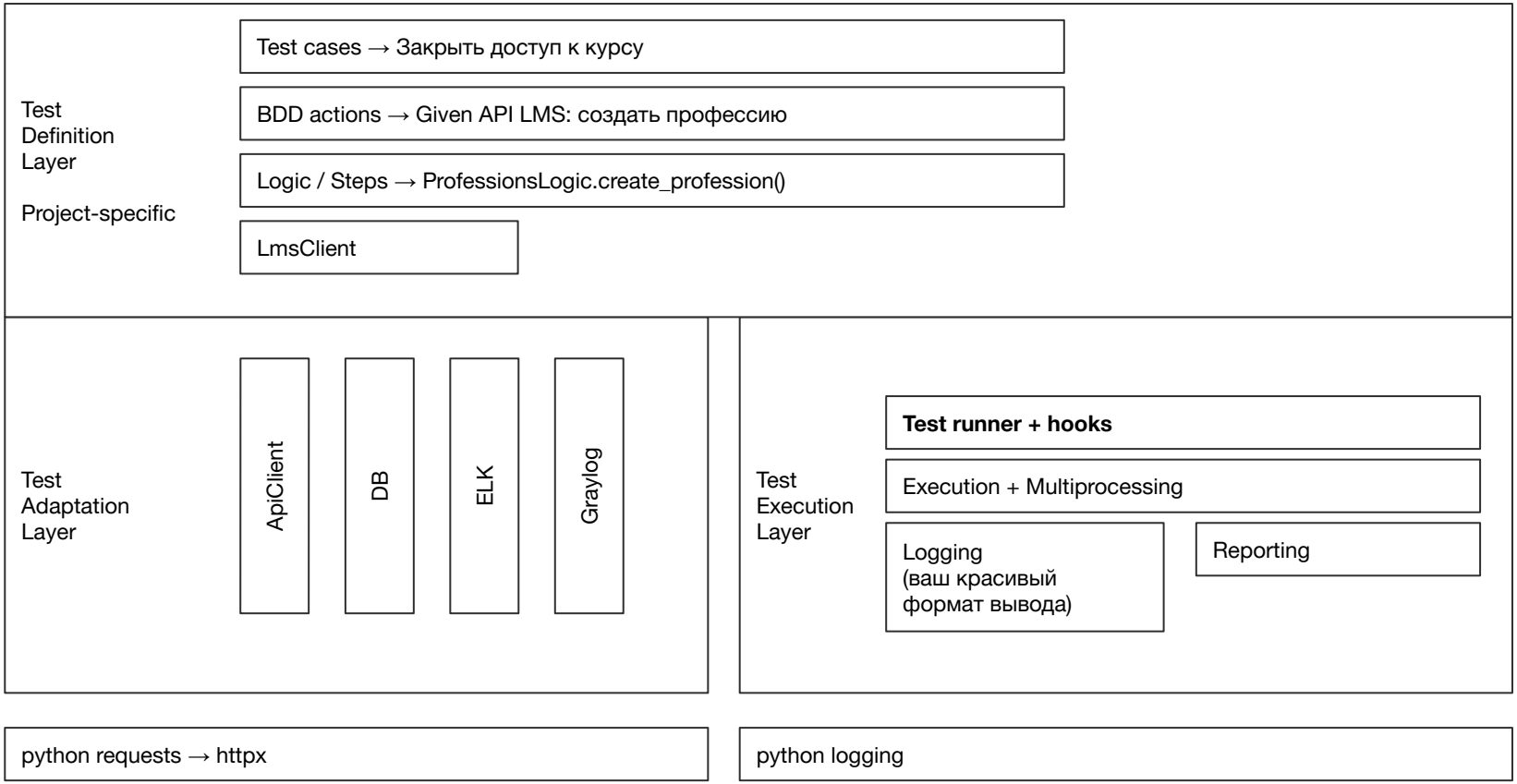
Куда добавить User-Agent

```
GET /api/v1/whoami HTTP/2
user-agent: python-requests/2.28.2 (qa; allure.id:14001)
accept: */*
```

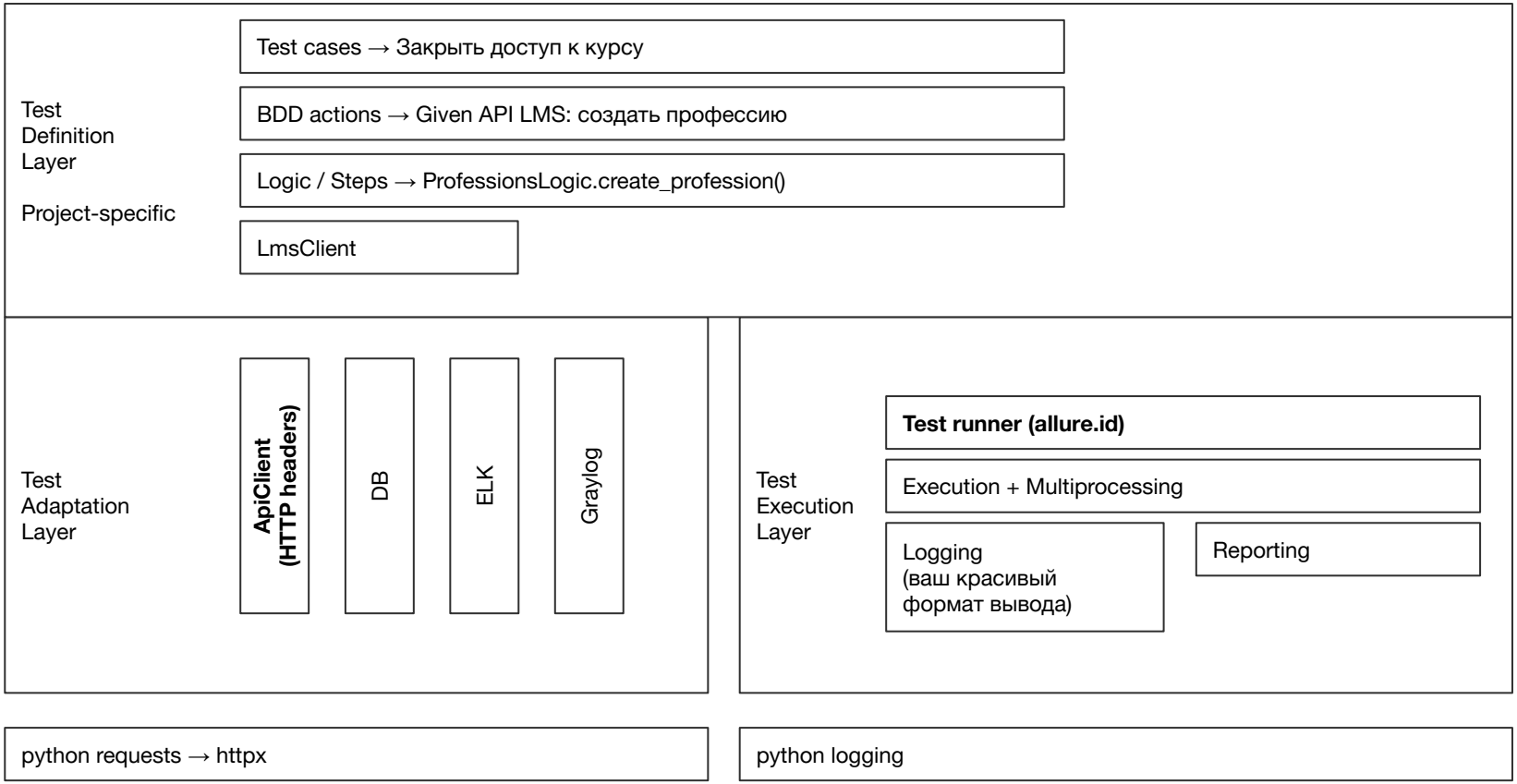
Куда добавить User-Agent



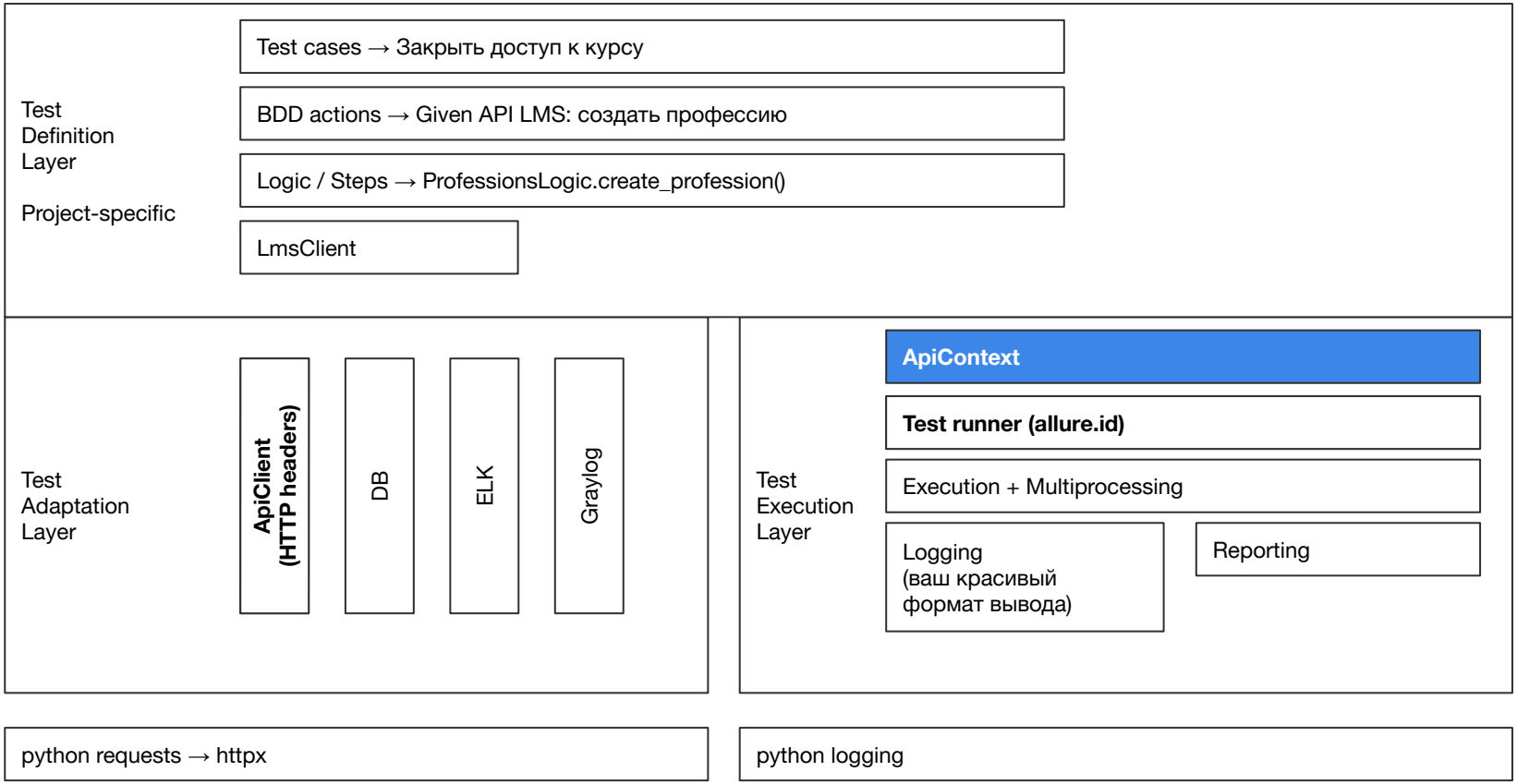
Куда добавить User-Agent



Куда добавить User-Agent



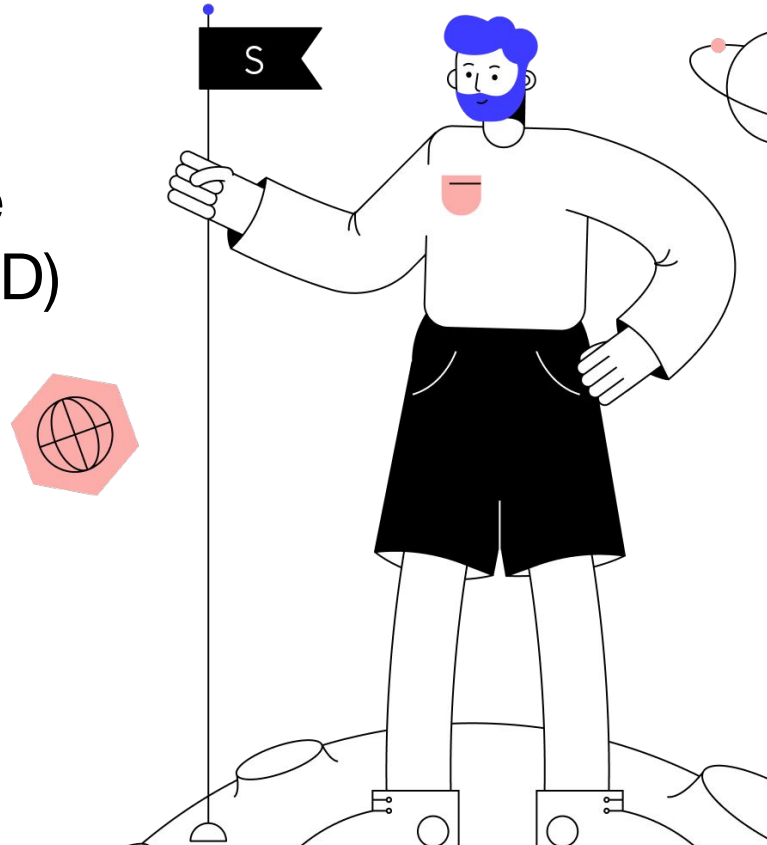
Куда добавить User-Agent



Подведем **итоги**

Подведем итоги

1. Учитесь разработке
2. Нет идеальной архитектуры
3. Композиция > Наследование
4. Выделяйте абстракции (SOLID)
5. Нарисуйте C1-C2



О чем мой доклад

~~BDD / Cucumber~~
~~ISTQB~~
~~Паттерны~~
~~Архитектурные принципы~~

Автоматизатор → разработчик

Автоматизаторы занимаются **разработкой**

Как построить дом

Дом, построенный на песке, не устоит

Есть схема ISTQB, а есть реальность
Если ваш проект не соответствует схеме?

У каждого своя реализация ISTQB

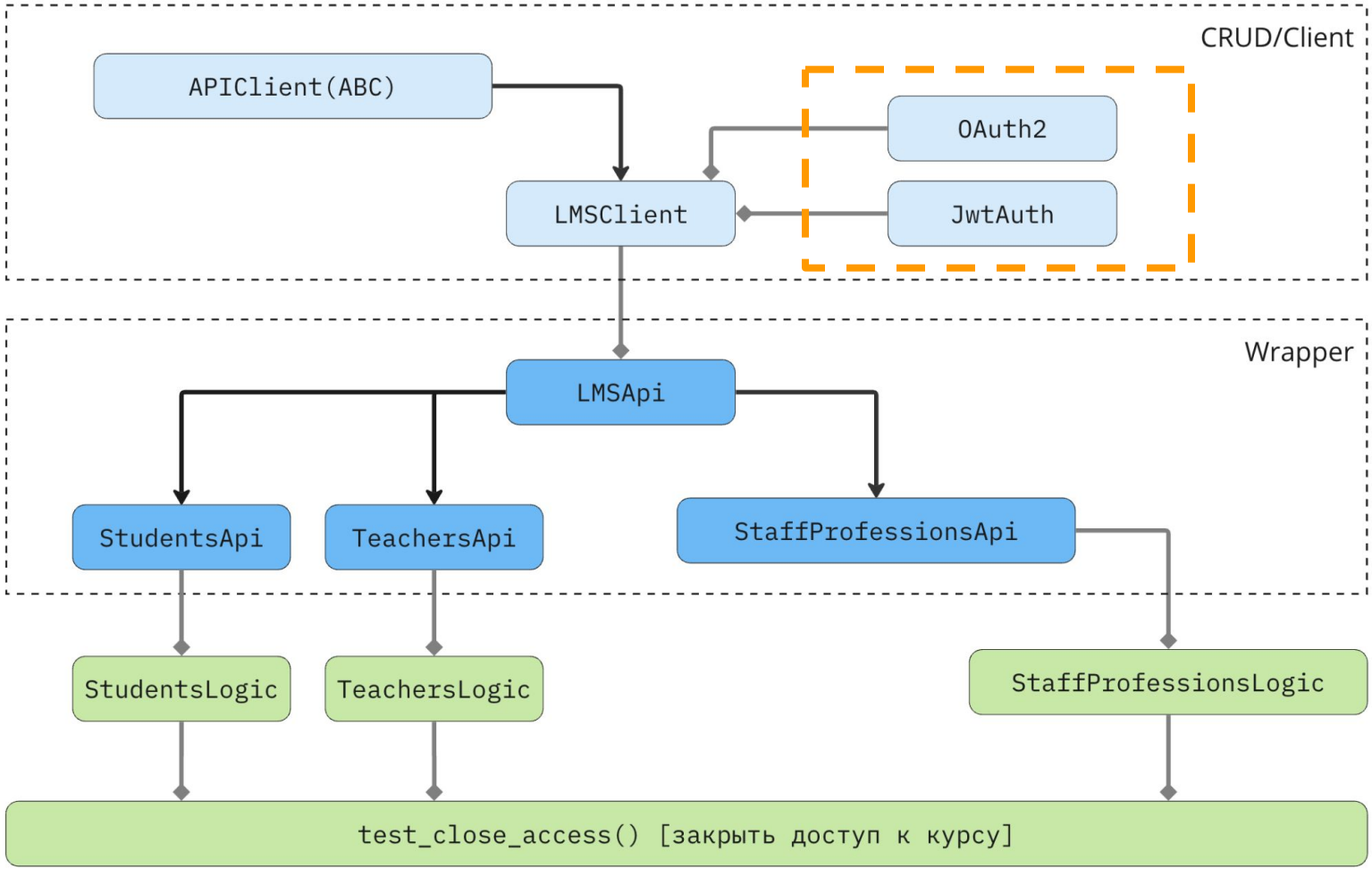
```python
class ISTQBPerfectArch(ABC):
    pass


class MyNotPerfectFramework(ISTQBPerfectArch):

    def not_implemented():
        raise NotImplementedError
```

Композиция vs наследование

Composition over inheritance principle

Наследование ограничивает свободу действий.

Оно приковывает нас к определенной структуре, которую невозможно изменить.

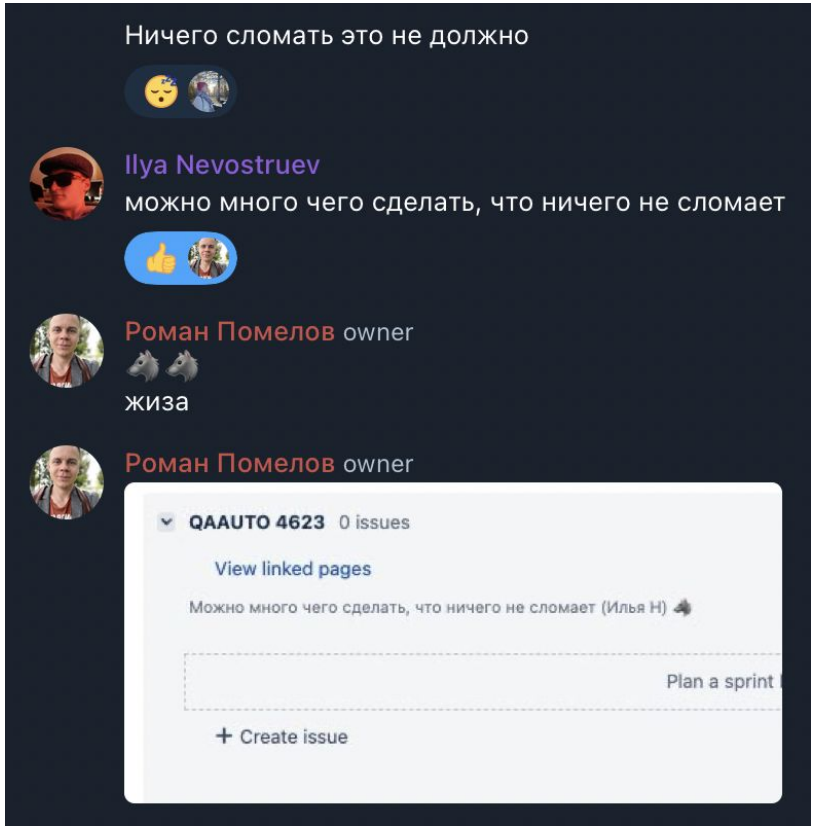

CRUD/Client
APIClient(ABC)
OAuth2
LMSClient
JwtAuth
Wrapper
LMSApi
StudentsApi
TeachersApi
StaffProfessionsApi
StudentsLogic
TeachersLogic
StaffProfessionsLogic
test_close_access() [закрыть доступ к курсу]

Выделяйте абстракции вовремя

Не делайте ничего лишнего до тех пор, пока не начнёт болеть.

Гораздо проще изменять код, который лежит в одном месте и не разбит на множество мелких частей

Можно много чего сделать



Вредные мысли про паттерны

Чувствую, что нужен паттерн, но какой сюда применить не знаю…

Мысль закралась в сторону стратегии, но не уверен, как ее сюда применить…

Гуру паттернов, посоветуйте, как быть…

SOLID

Задачка для собеседования:

- расшифруйте любую букву из SOLID с примером в автоматизации

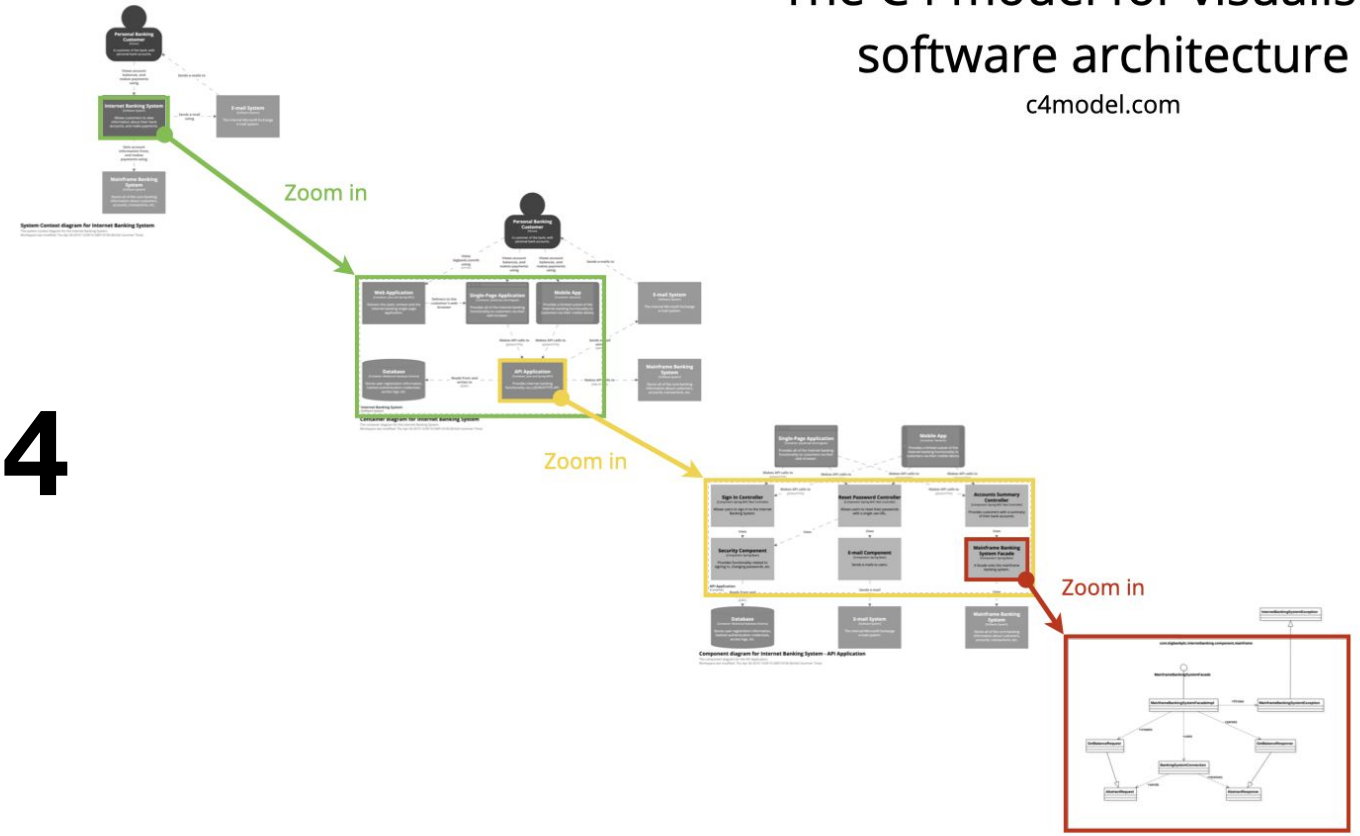
d = dependency inversion

```python
from requests import Session


class ApiClient(ABC):

    def __init__(self, base_url: str):
        self.base_url = base_url
        self.session = Session()

    @log_request_timings
    @retry(2)
    def send(self, *args, **kwargs):
        try:
            self.last_response = self.session.request(*args, **kwargs) # ⟵ change here
        except RequestException as e:
            self.log_error(e, *args, **kwargs)
            raise e

        return self.last_response


```

The C4 model for visualising software architecture

c4model.com

C4



Level 1	Level 2	Level 3	Level 4
Context	Containers	Components	Code

12-factor-apps

Принципы создания
software-as-a-service apps

THE TWELVE-FACTOR APP


ВОТ ТАК, С ПОМОЩЬЮ НЕХИТРЫХ ПРИСПОСОБЛЕНИЙ БУХАНКУ БЕЛОГО (ИЛИ ЧЕРНОГО) ХЛЕБА МОЖНО ПРЕВРАТИТЬ В ТРОЛЛЕЙБУС... НО ЗАЧЕМ?!
УМЕЛЫЕ РУКИ.

- Кто хочет стать SDET?
- **Начинайте сейчас!**

Спасибо!

Роман Помелов,
QA Automation Lead

Skillbox

https://t.me/pomelov