Test Automation Strategy

for Legacy System

Artur Basak

GROCON

GRODNO 2019

Few words about Me

Artur Basak

Programmer, ex-Lead UI Engineer at Lition Energy

Developing, Implementing and Testing User Interfaces more than 9 years.

Passionate about Web, Human-Computer Interaction and Automated Testing.

/artur.basak

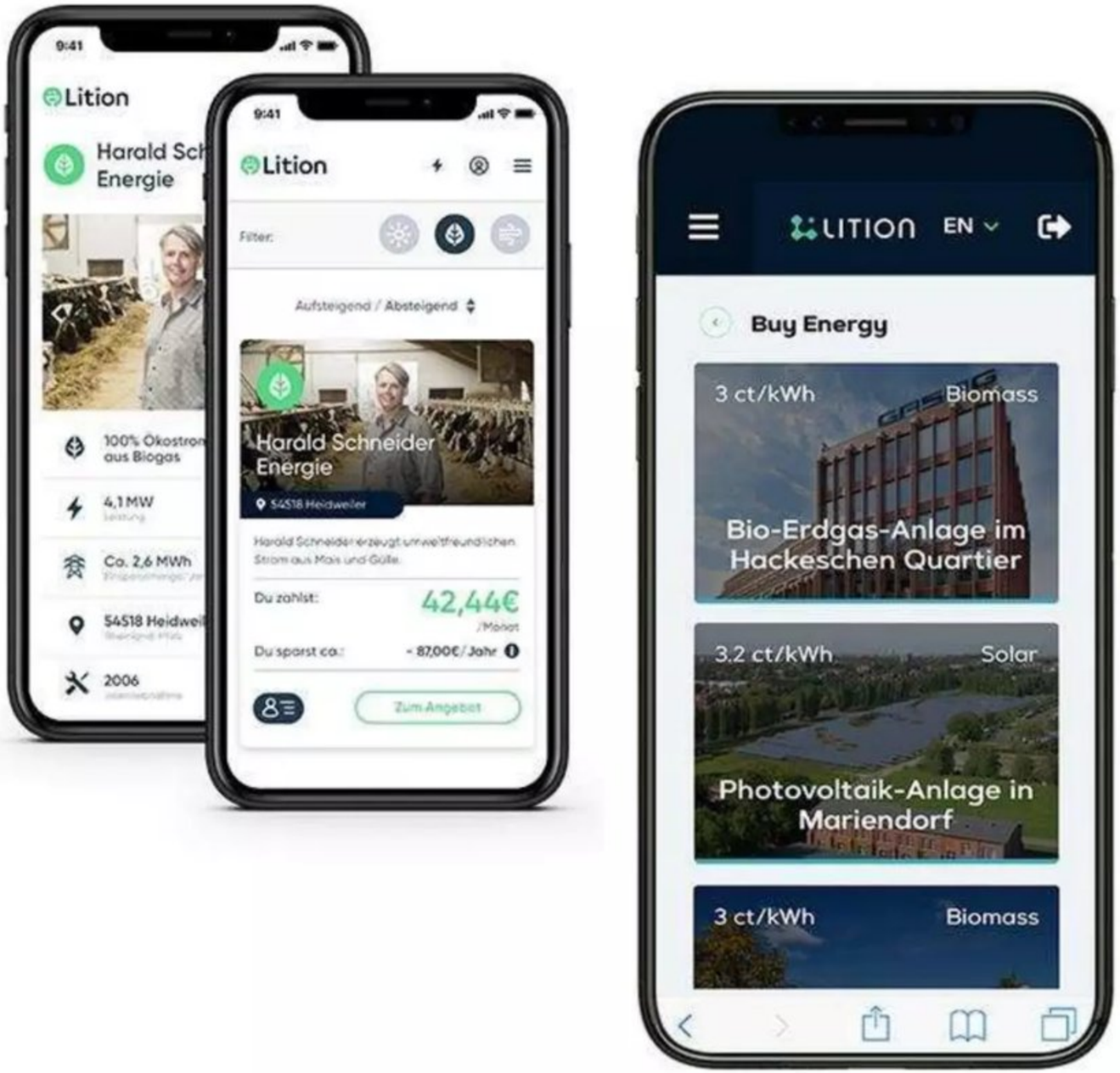
/arturbasak

/front-end-in-regions-grodno

Lition Energy Project

Lition Energy

Lition Energy is an innovative electricity provider that connects consumers directly with renewable energy producers - without detours and unnecessary additional costs. On our digital platform, members choose their green power plant quickly and easily themselves.


9:41
Lition
Harald Sch
Energie
100% Ökostrom
aus Biogas
4,1 MW
Co. 2,6 MWh
54518 Heidwei
2006
9:41
Lition
Filter:
Aufsteigend / Absteigend
Harald Schneider
Energie
54518 Heidweiler
Harald Schneider erzeugt umweltfreundlichen Strom aus Mais und Gülle.
Du zahlst:
42,44€
/Monat
Du sparst ca.:
- 87,00€/Jahr
Zum Angebot
LITION
EN
Buy Energy
3 ct/kWh
Biomass
Bio-Erdgas-Anlage im
Hackeschen Quartier
3.2 ct/kWh
Solar
Photovoltaik-Anlage in
Mariendorf
3 ct/kWh
Biomass

UI / Frontend

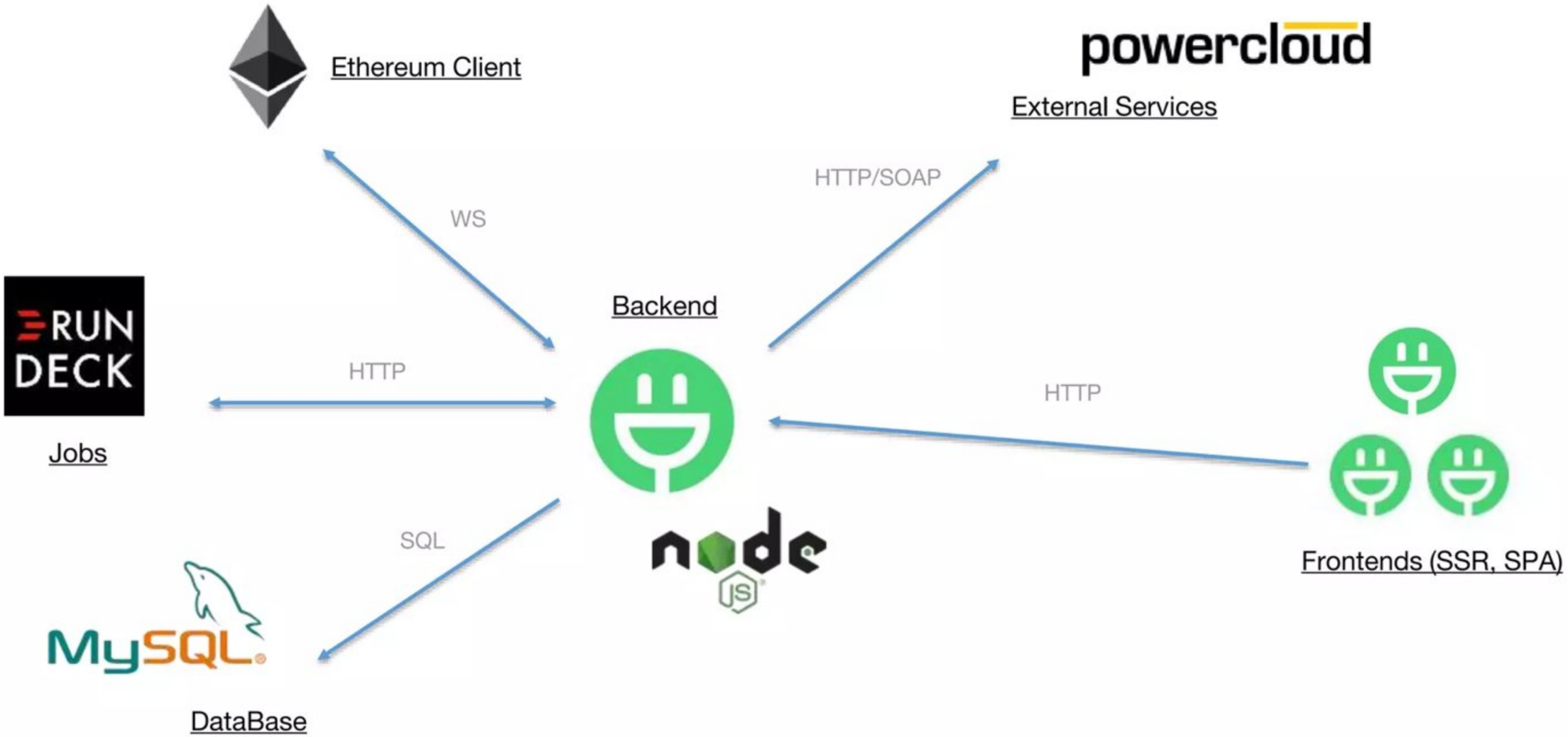
Backend: Need Help! SOS!

Frame from «Game of Thrones» series

Need Help!

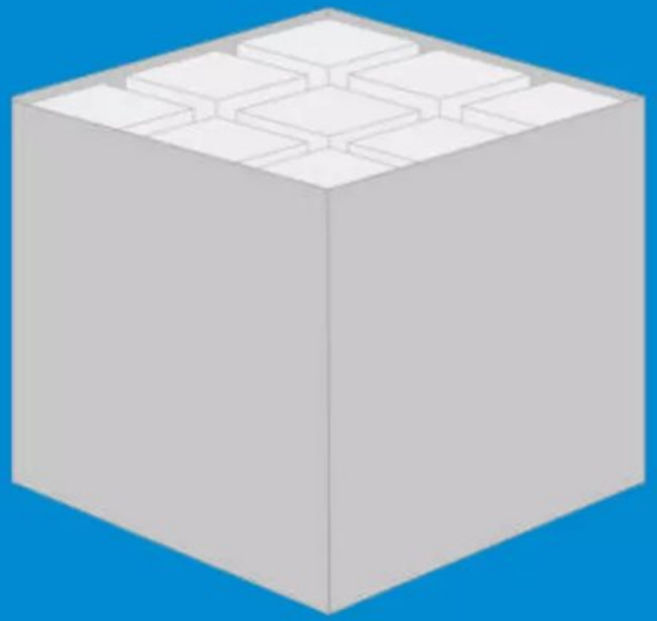
SOS!

Backend on High-Level



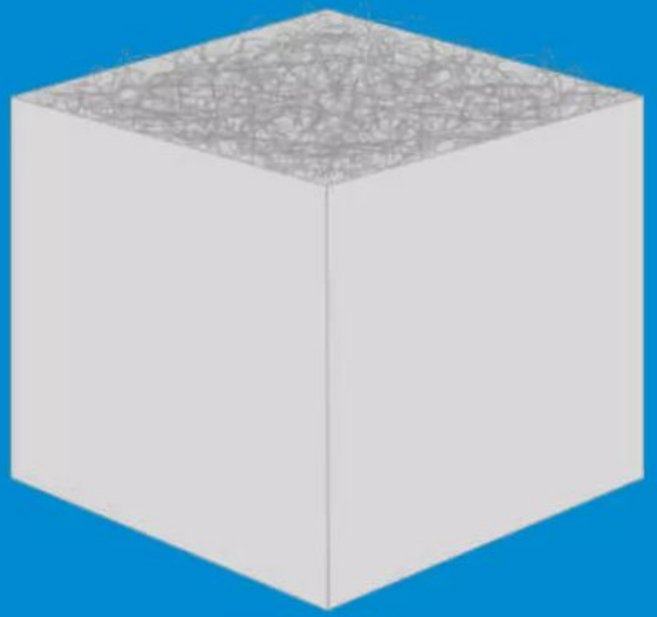
Don’t start with a monolith

… when your goal is a microservices architecture



Hope

vs.



Reality

Stefan Tilkov (c)

https://martinfowler.com/articles/dont-start-monolith.html

Initial State

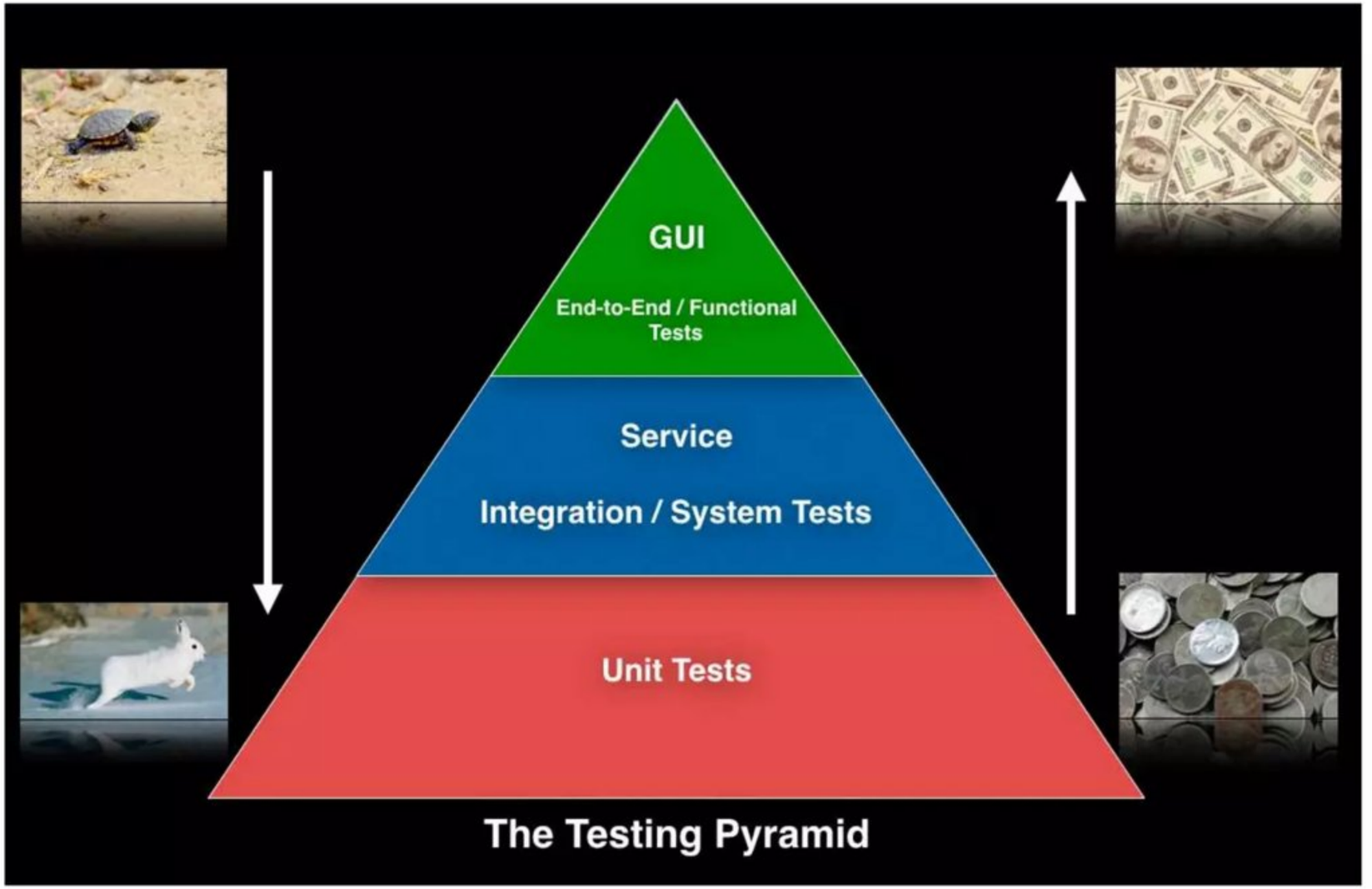
- Legacy Codebase
- Callback Hell
- No DB Migrations
- No Tests!
- No Lints!
- Not Styled!

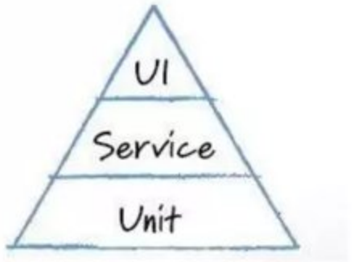
Frame from «Game of Thrones» series

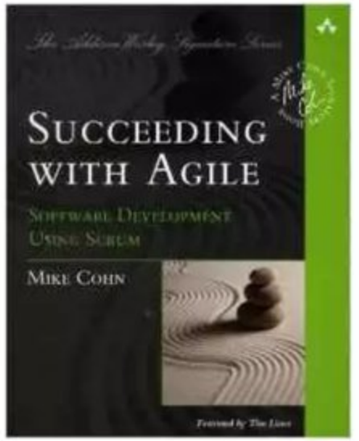
Test Strategy?

Frame from «Game of Thrones» series

The Testing Pyramid







https://medium.com/front-end-in-regions-grodno/the-testing-pyramid-2fec1c1fef91

The Testing Trophy

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called "functional testing" or e2e.

Integration

Verify that several units work together in harmony.

Unit

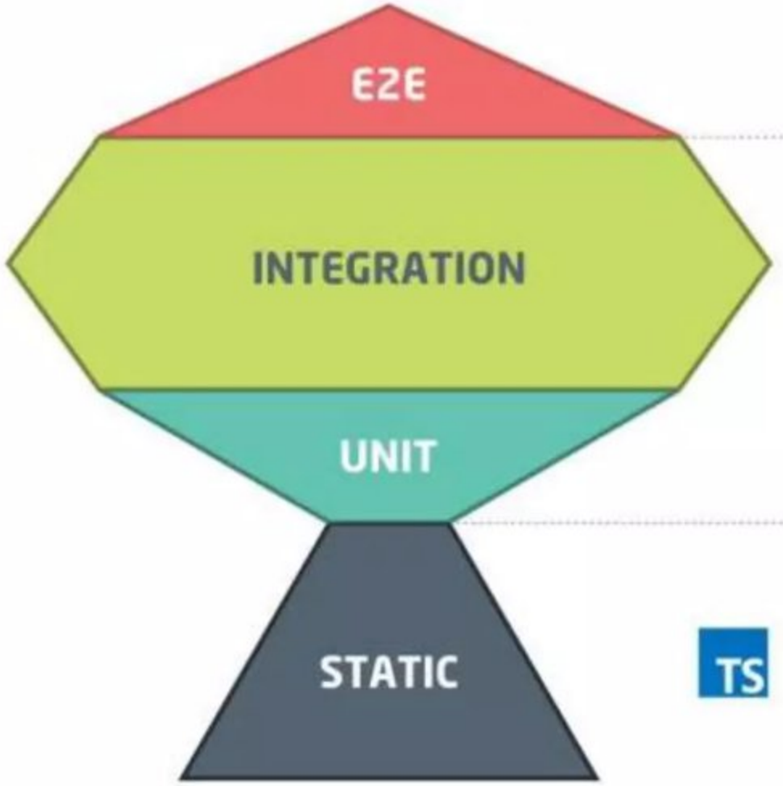
Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.

https://testingjavascript.com

The Testing Trophy

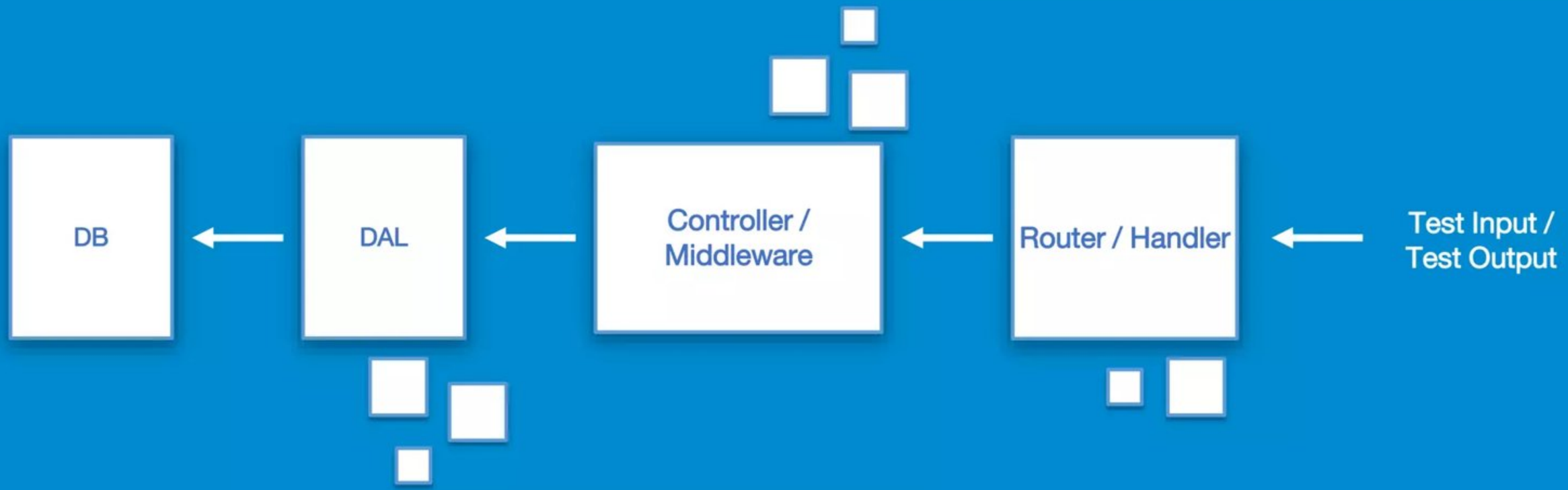


cypress.io

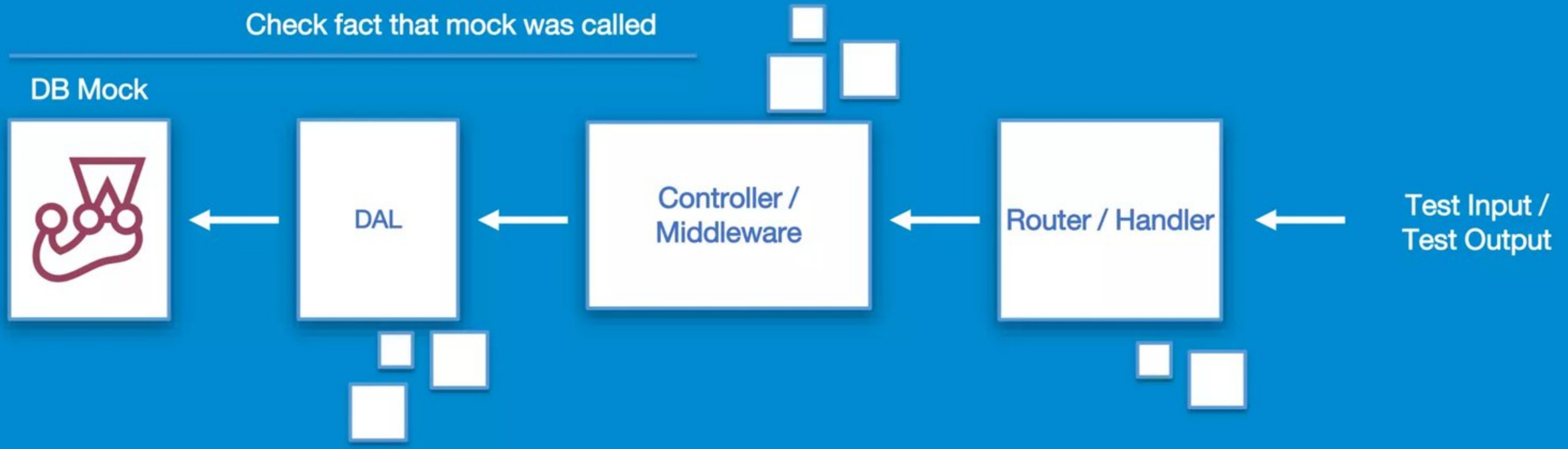
Jest

TS TypeScript

Anatomy of Integration Test



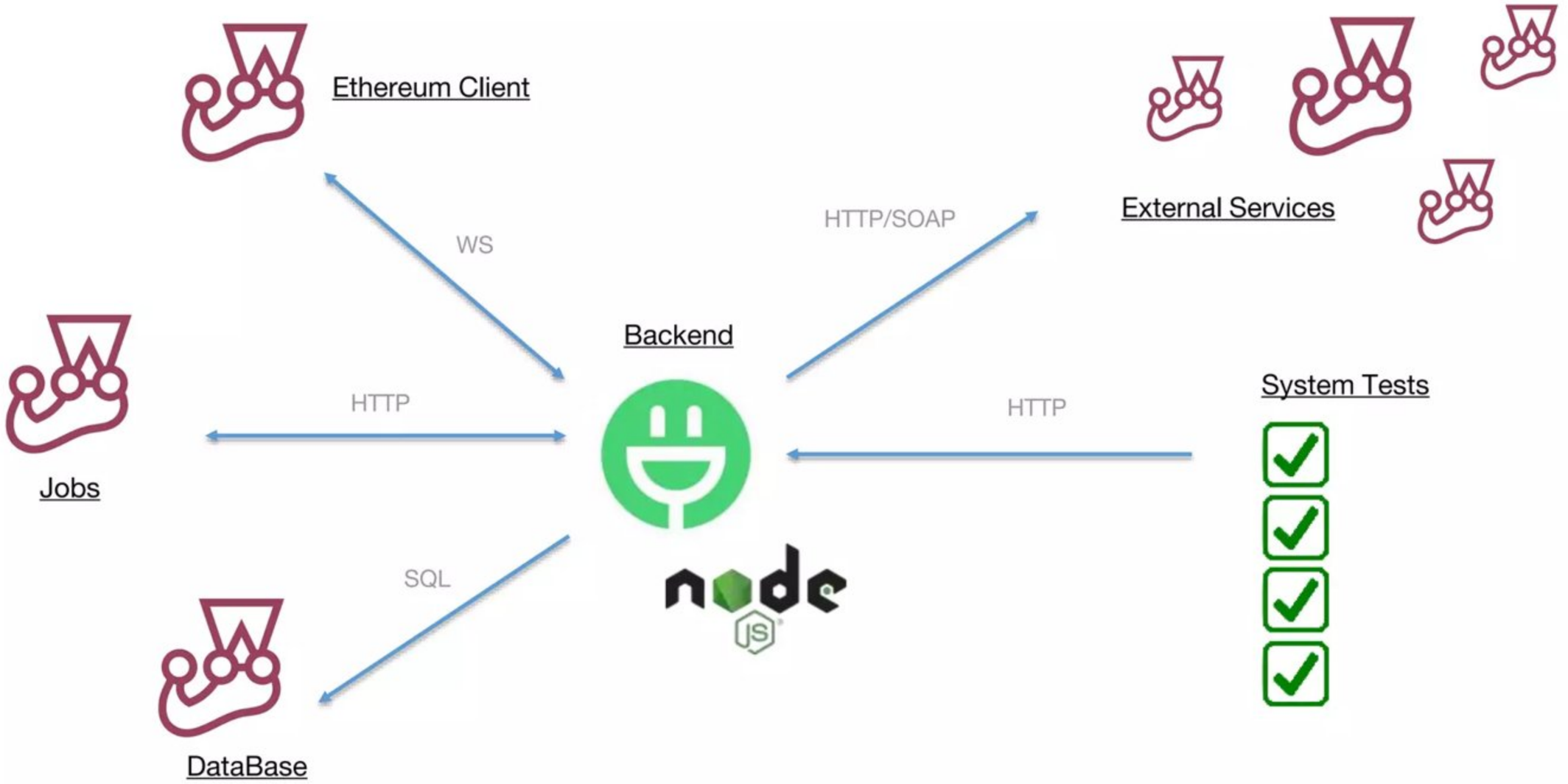
Anatomy of System Test



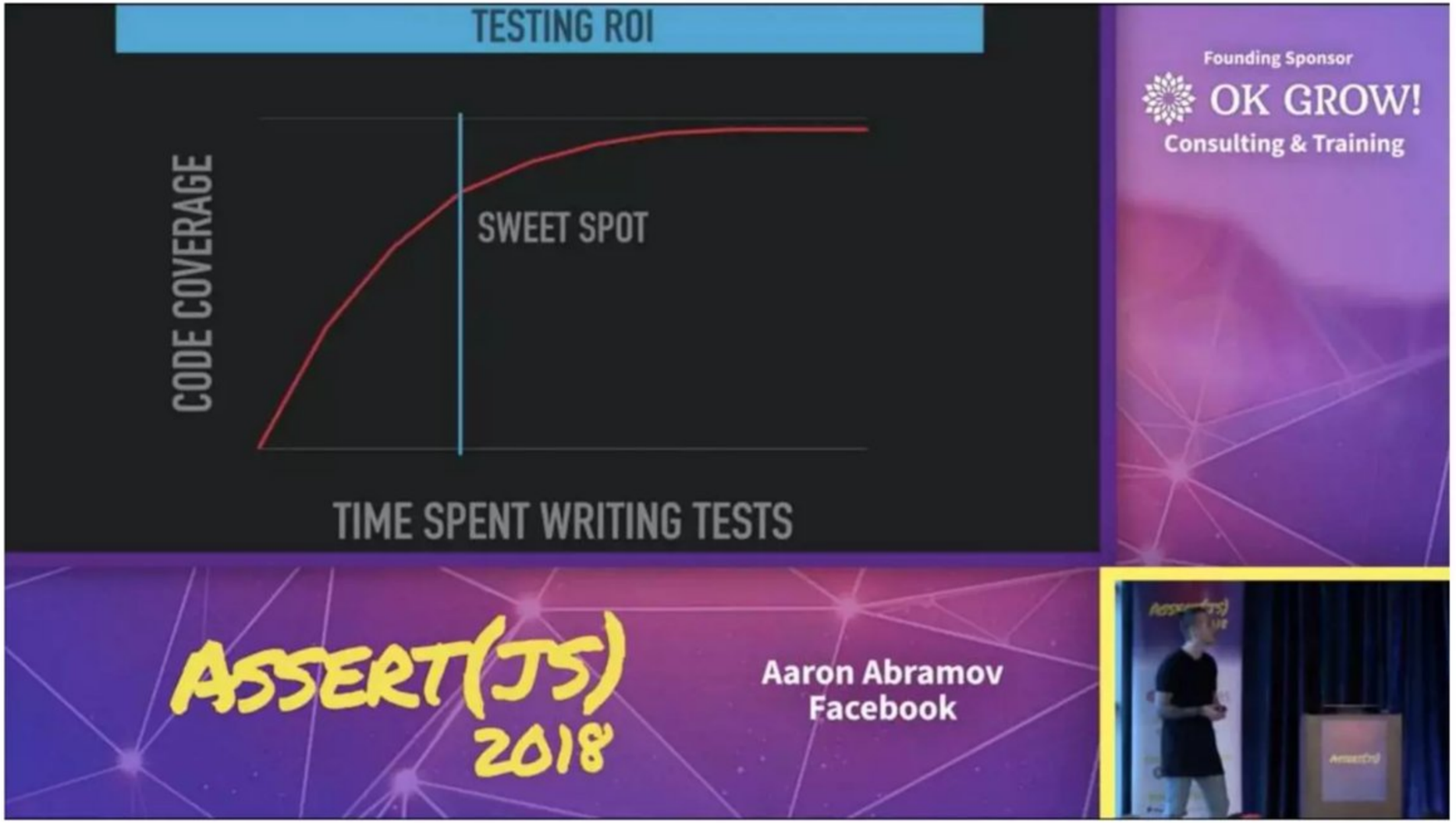
Test Strategy «Siege»

Frame from «Game of Thrones» series

System under Testing



Right Way for Test Coverage



~~100%~~

~80%

Assert(js) 2018 Conference. Screen from @aarondjents Twitter

Delta Coverage

All files / lition-de-site/components/TariffCalculationForm TariffCalculationForm.js

100% Statements 21/21 **90.91%** Branches 20/22 **88.89%** Functions 8/9 **100%** Lines 21/21

istanbul

```
1    import React from 'react';
2    import PropTypes from 'prop-types';
3    import classNames from 'classnames';
4    import { Button, ConsumptionSelect, LocationInput, TextInput } from 'lition-
5    import { CONSUMPTION_OPTIONS, ERRORS } from '../../constants';
6    import './TariffCalculationForm.css';
7
8    export default class TariffCalculationForm extends React.Component {
9        constructor(props) {
10   6x      super(props);
11   6x      const { query } = props;
12   6x      this.state = {
13               zip: query.zip || '',
14               consumption: query.consumption || '',
15               requiredFormError: null,
16               errors: {
17                   zip: '',
18                   consumption: ''
19               }
20           };
21       }
22
```

«Herb Derby came up with the notion of delta coverage - what coverage does this test provide that's unique? Tests with zero delta coverage should be deleted unless they provide some kind of communication purpose.»

Kent Beck, «Is TDD dead? #4»

https://martinfowler.com/articles/is-tdd-dead

Test Tools

Jest

https://www.npmjs.com/package/jest

1. Already used on Frontend Side
2. Backend Team have no Experience with Tests
3. All inside Box
    1. Test Executor
    2. Test Doubles (Spies, Mocks, Stubs)
    3. Expectation Functions
    4. Test Coverage Report
4. Most Popular

Supertest

https://www.npmjs.com/package/supertest

Abstract TestBed

abstractOld.js

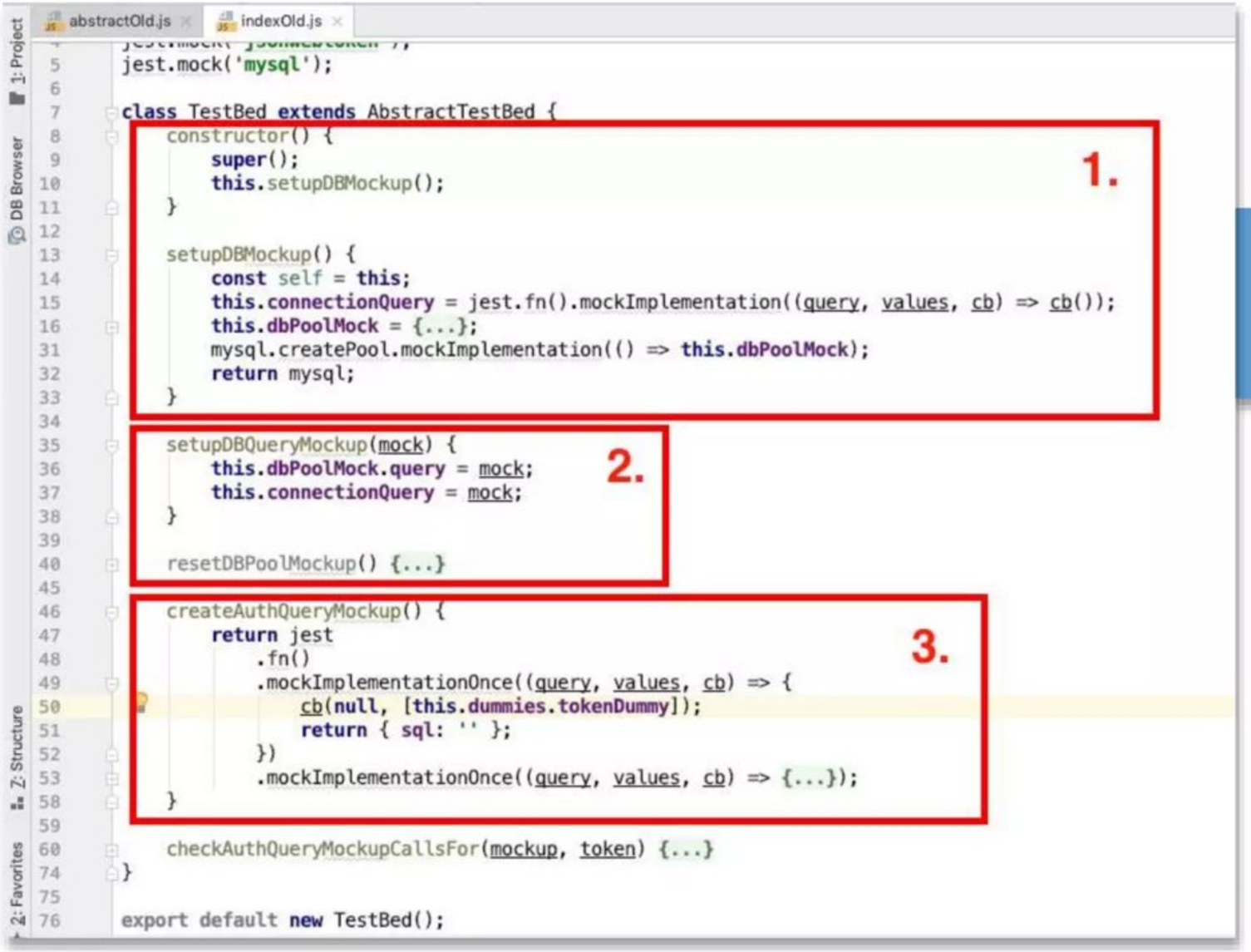
```
1  import express from 'express';
2  import bodyParser from 'body-parser';
3  import cookieParser from 'cookie-parser';
4  import request from 'supertest';
5  import createParametersContainer from '../../middlewares/common/createParametersContainer';
6  import mainRoute from '../../routes/index';
7  import basicErrorHandler from '../../middlewares/errorHandler/basicErrorHandler';
8
9  class AbstractTestBed {
10     constructor() {
11         const app = express();
12
13         app.use(cookieParser());
14         app.use(bodyParser.json());
15         app.use(bodyParser.urlencoded({ extended: true }));
16         app.use(createParametersContainer);
17         app.use('/', mainRoute);
18         app.use(basicErrorHandler);
19
20         this.sut = request(app);
21         this.dummies = {
22             userDummy: {id: 1...},
27             tokenDummy: {...}
32         };
33         this.expectations = {
34             externalServiceURL: 'https://demo.service.de/rest:client',
35             tokenQuery: 'SELECT * FROM authenticationToken WHERE token=?',
36             userQuery: 'SELECT * FROM user WHERE id=?'
37         };
38     }
39
40     getSUT() {
41         return this.sut;
42     }
43 }
44
45 export default TestBedAbstract;
```

1.

2.

1. Build SUT
2. Define dummies & expectations

TestBed



1. Setup DB Mockup
2. Manage Query Mockup
3. Utils

System Test for Logout End-point

logout.test.js

```
1  import testBed from '../../../../../testSetup/testBed';
2
3  const expectedResponse = {message: 'Successfully logged out user 1 and invalidated t
6  const expectedInvalidateTokenQuery = 'UPDATE authenticationToken SET isValid=? WHERE
7  const validTokenQuery = testBed
8          .createAuthQueryMockup()
9          .mockImplementationOnce((query, values, cb) => {
10             cb(null);
11             return { sql: '' };
12         });
13
14 const token = 'test_token';
15
16 describe('User Logout Router', () => {
17     describe('success cases:', () => {
18         beforeEach(() => {
19             testBed.setupDBQueryMockup(validTokenQuery);
20         });
21
22         it('should handle GET /users/logout', done => {
23             testBed
24                 .getSUT()
25                 .get('/v0.1/user/logout')
26                 .set('Authorization', `Bearer ${token}`)
27                 .expect('Content-type', /json/)
28                 .expect(200)
29                 .end((err, res) => {
30                     expect(err).toBeFalsy();
31                     expect(res.body).toEqual(expectedResponse);
32                     expect(res.body.message).toBeTruthy();
33
34                     testBed.checkAuthQueryMockupCallsFor(validTokenQuery, token);
35
36                     const [
37                         , , [tokenQuery, tokenValues, tokenQueryCallback]
38                     ] = validTokenQuery.mock.calls;
39
40                     expect(tokenQuery).toEqual(expectedInvalidateTokenQuery);
```

1.

2.

3.

1. Create Query Mockup
2. Setup Query Mockup
3. Compare result & expectations

Test Coverage

Frames from «Game of Thrones» series

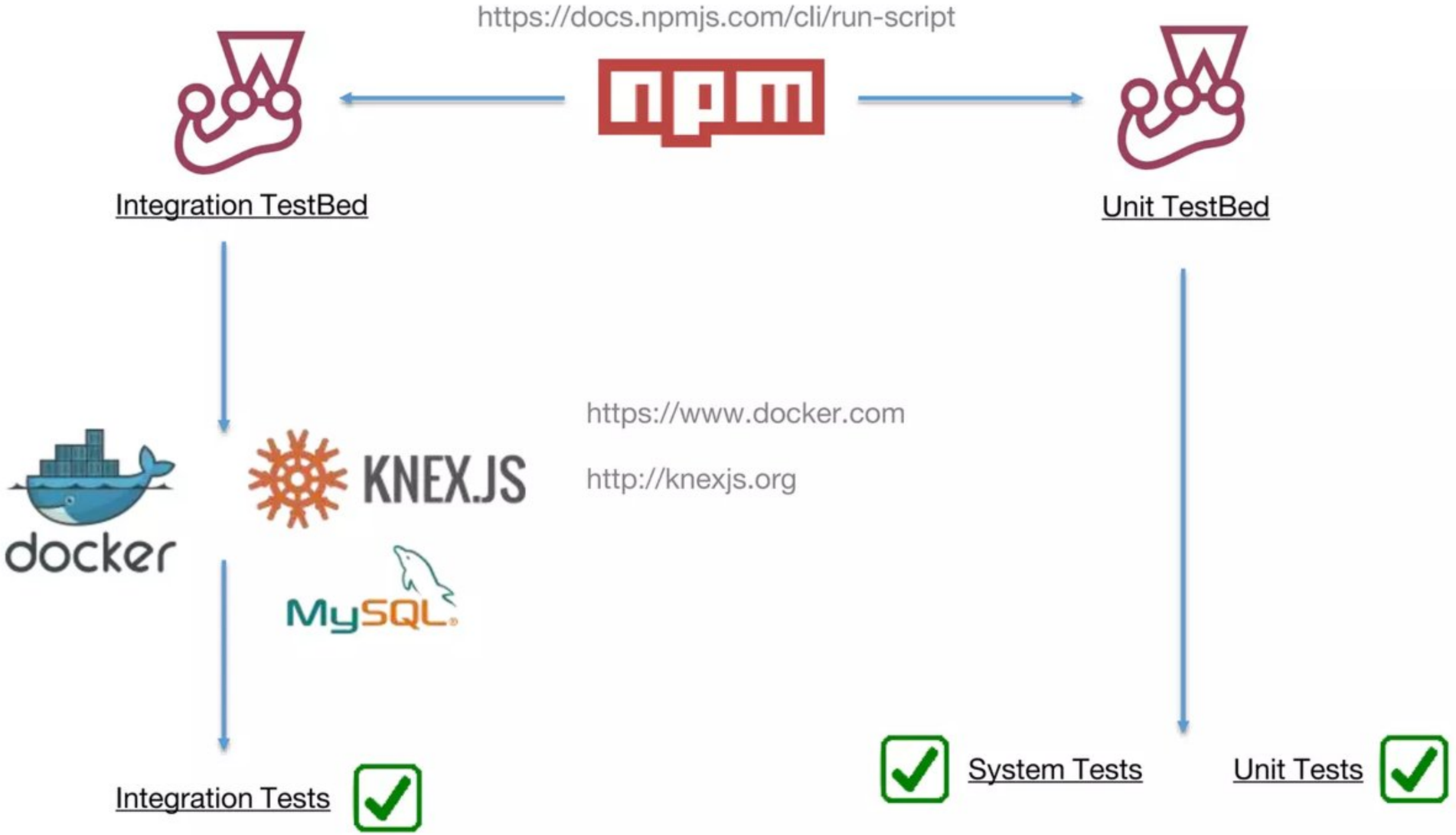
1. Statements
2. Branches
3. Functions
4. Lines

~40%

Dive into codebase

Frames from «Game of Thrones» series

Jest Test Configs



Integration TestBed

integration.js

```
1   import ...
10
11  jest.mock('../../assets/logger');
12
13  class TestBed extends TestBedAbstract {
14      constructor() {...}
28
29      getSUT() {
30          return this.sut;
31      }
32
33      loadInitialData() {
34          return global.__KNEX__.seed.run();
35      }
36  }
37
38  export default new TestBed();
39
```

1.

testEnvironment.js

```
1   const knex = require('knex');
2   const NodeEnvironment = require('jest-environment-node');
3   const { knexOptions } = require('../../scripts/getConfig');
4
5   class TestEnvironment extends NodeEnvironment {
6       constructor(config) {
7           super(config);
8           this.connection = null;
9       }
10
11      async setup() {
12          this.global.__KNEX__ = knex(knexOptions);
13          await super.setup();
14      }
15
16      async teardown() {
17          await this.destroyConnection();
18          await super.teardown();
19      }
20
21  }
```

2.

abstract.js

```
1
2   class AbstractTestBed {
3       constructor() {
4           this.expectations = {
5               externalServiceURL: 'https://demo.service.de/rest:client'
6           };
7       }
8   }
9
10  export default AbstractTestBed;
11
```

1. Manage seeds (data fixtures)
2. Configure test env

Integration Test for Logout End-point

logout.e2e.js

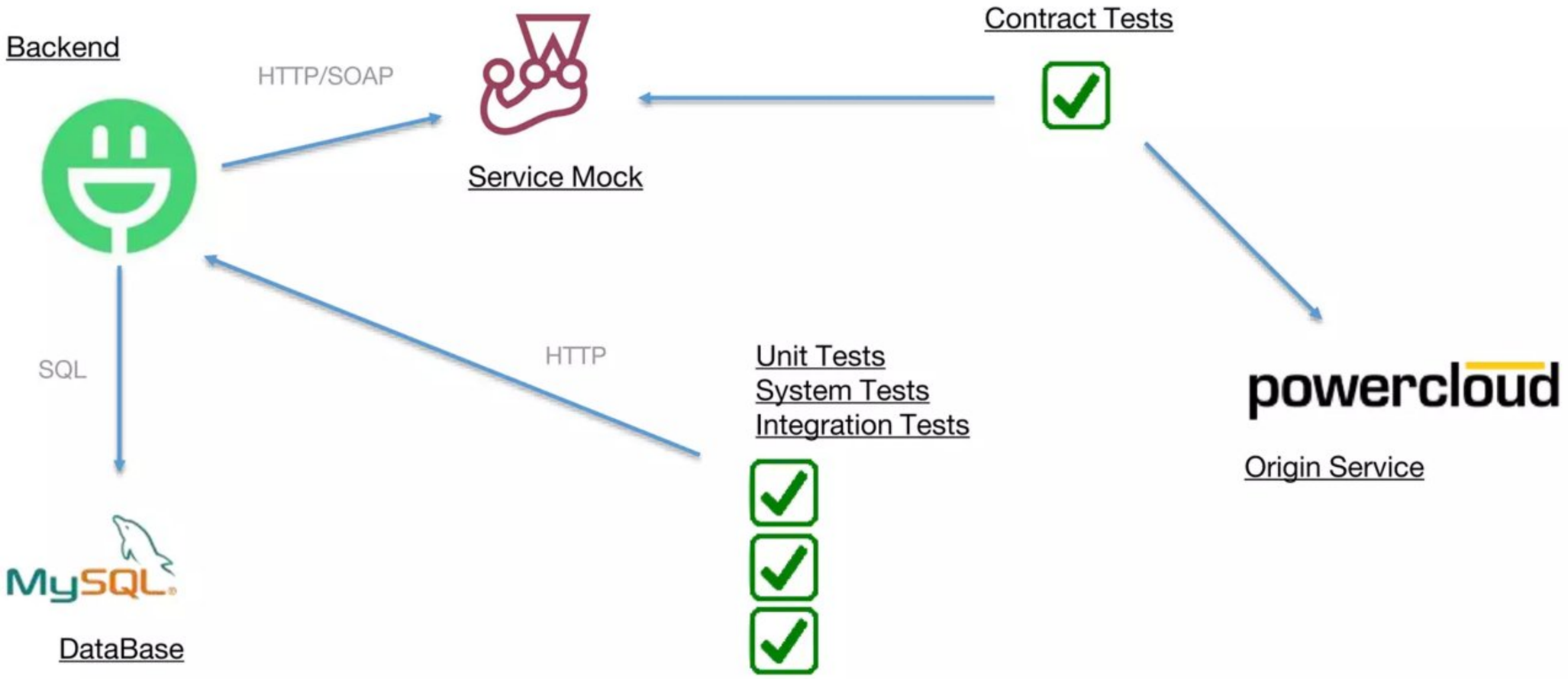
```
1  import testBed from '../../../../testSetup/testBed/integration';
2  import { authenticationTokenSeeds } from '../../../../testSetup/fixtures/db/authenticationToken';
3
4  const [tokenSeed] = authenticationTokenSeeds;
5  const { token } = tokenSeed;
6
7  describe('GET /users/logout', () => {
8      beforeAll(async () => {
9          await testBed.loadInitialData();
10     });
11
12     it('should response with OK status code (200) and success message if user logout is succeed', d
13         testBed
14             .getSUT()
15             .get('/v0.1/user/logout')
16             .set('Authorization', `Bearer ${token}`)
17             .expect('Content-type', /json/)
18             .expect(200)
19             .end((err, res) => {
20                 expect(err).toBeNull();
21                 expect(res.body).toEqual({ message: 'Successfully logged out user 1 and invalidated
22
23                 done();
24             });
25     });
26 });
27
```

2.

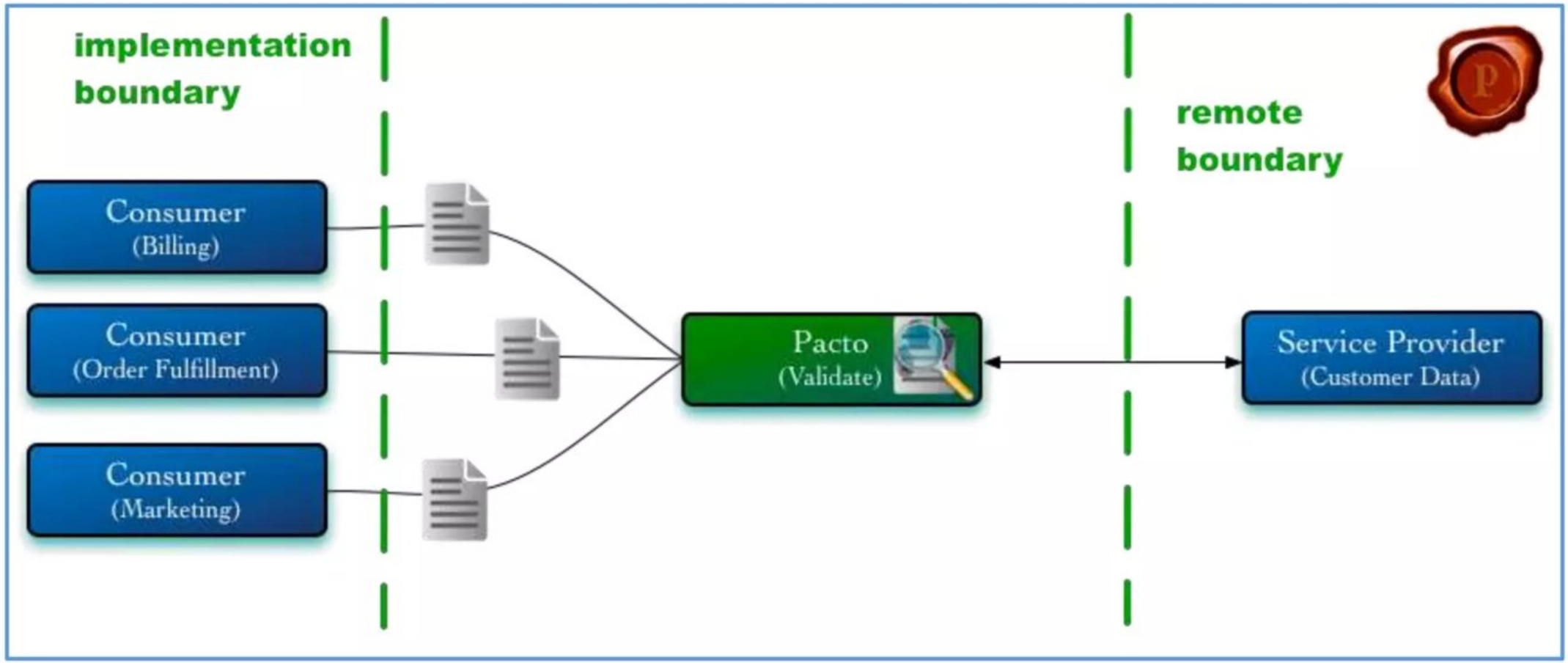
1.

1. Load test data
2. Reuse fixtures

System under Testing



Consumer-Driven Contracts



https://thoughtworks.github.io/pacto/patterns/cdc

PACTS

https://docs.pact.io

External Service Fake

integration.js | fakeExternalService.js

```
const {
    EXTERNAL_SERVICE_SETTINGS: { URL, API_HASH, API_AUTHORIZATION }
} = config;

export default function fakeExternalService() {
    const expectedServiceURL = `${URL}/rest:client/${API_HASH}`;
    const expectedHeaders = {
        Authorization: `Basic ${API_AUTHORIZATION}`
    };

    needle.get = jest.fn().mockImplementation((url, options, callback) => {
        expect(callback).toEqual(expect.any(Function));
        expect(options).toEqual({ headers: expectedHeaders });
        expect(url).toContain(expectedServiceURL);

        const [operation, params] = url.replace(expectedServiceURL, '').split('?');
        const valueOf = getURLParamValue.bind(this, params);

        switch (operation) {
            case '/getCustomerById': {
                successResponse.body.data = fixtures.customers.find(c => c.id == valueOf('id'))
                callback(null, successResponse, successResponse.body);
                break;
            }
            /*
            * Other end-points
```

1. Identify endpoint
2. Use needed fixtures
3. Setup fake service

integration.js | fakeExternalService.js

```
import ...

jest.mock('../../assets/logger');

class TestBed extends TestBedAbstract {
    constructor() {...}

    getSUT() {
        return this.sut;
    }

    loadInitialData() {
        fakeExternalService();
        return global.__KNEX__.seed.run();
    }
}

export default new TestBed();
```

Contract Test

getCustomerById.contract.js

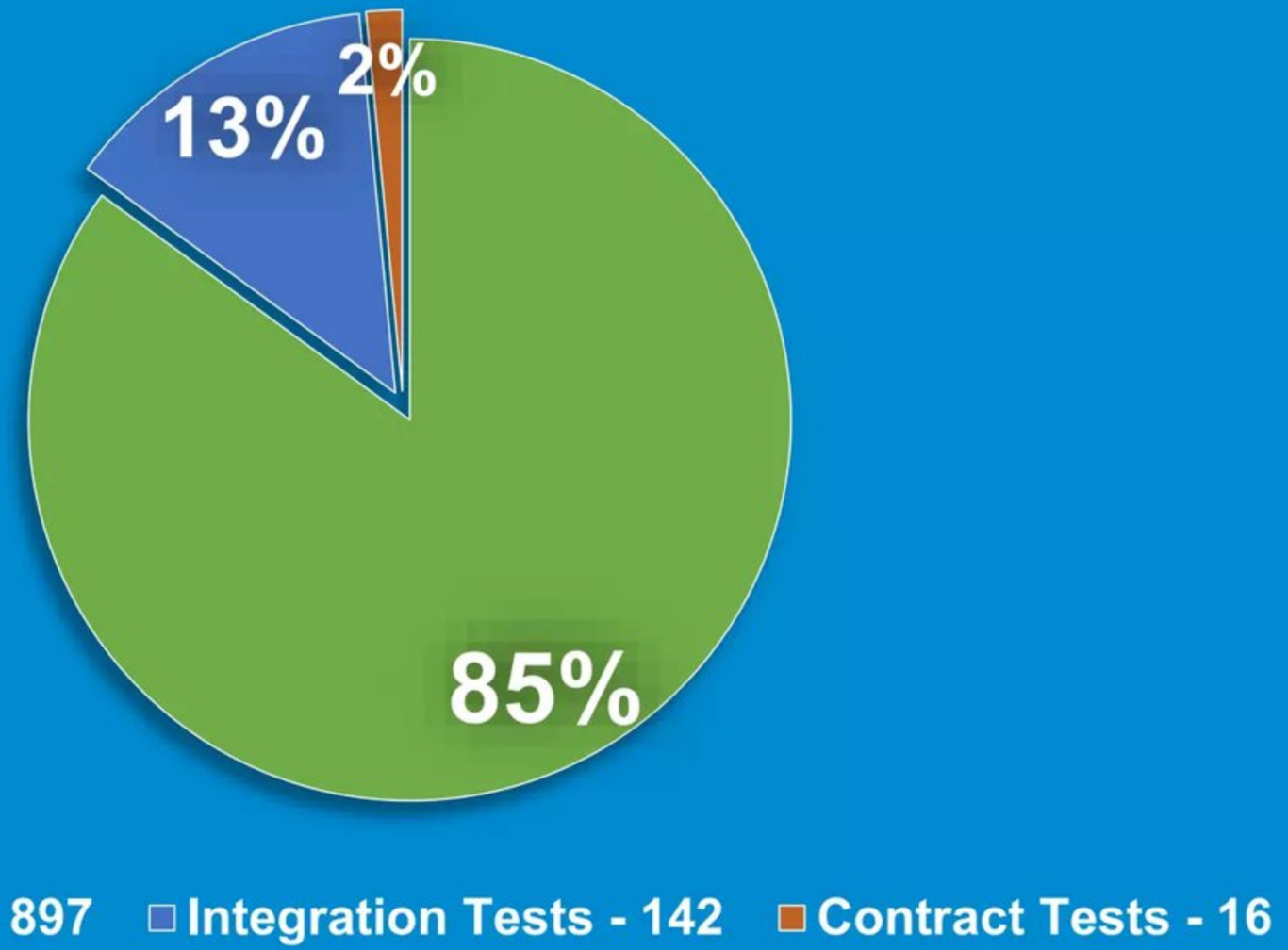
```
1  import client from '../../../../../services/externalService/client';
2  import fixtures from '../../../../fixtures/externalService';
3  import { CUSTOMER_ID, verifySuccessResponseStructure, verifyData } from '../externalServiceTestData';
4
5  describe('getCustomerById', () => {
6      it('should return expected structure with status 200', done => {
7          client.get(`/getCustomerById?id=${CUSTOMER_ID}`, (error, response) => {
8              expect(error).toBeNull();
9              verifySuccessResponseStructure(response);
10
11             const [consumerCustomer] = fixtures.customers;
12             const providerCustomer = response.body.data;
13
14             expect(typeof consumerCustomer.id).toEqual(typeof providerCustomer.id);
15             verifyData(consumerCustomer.data, providerCustomer.data);
16
17             done();
18         });
19     });
20 });
21
```

1.

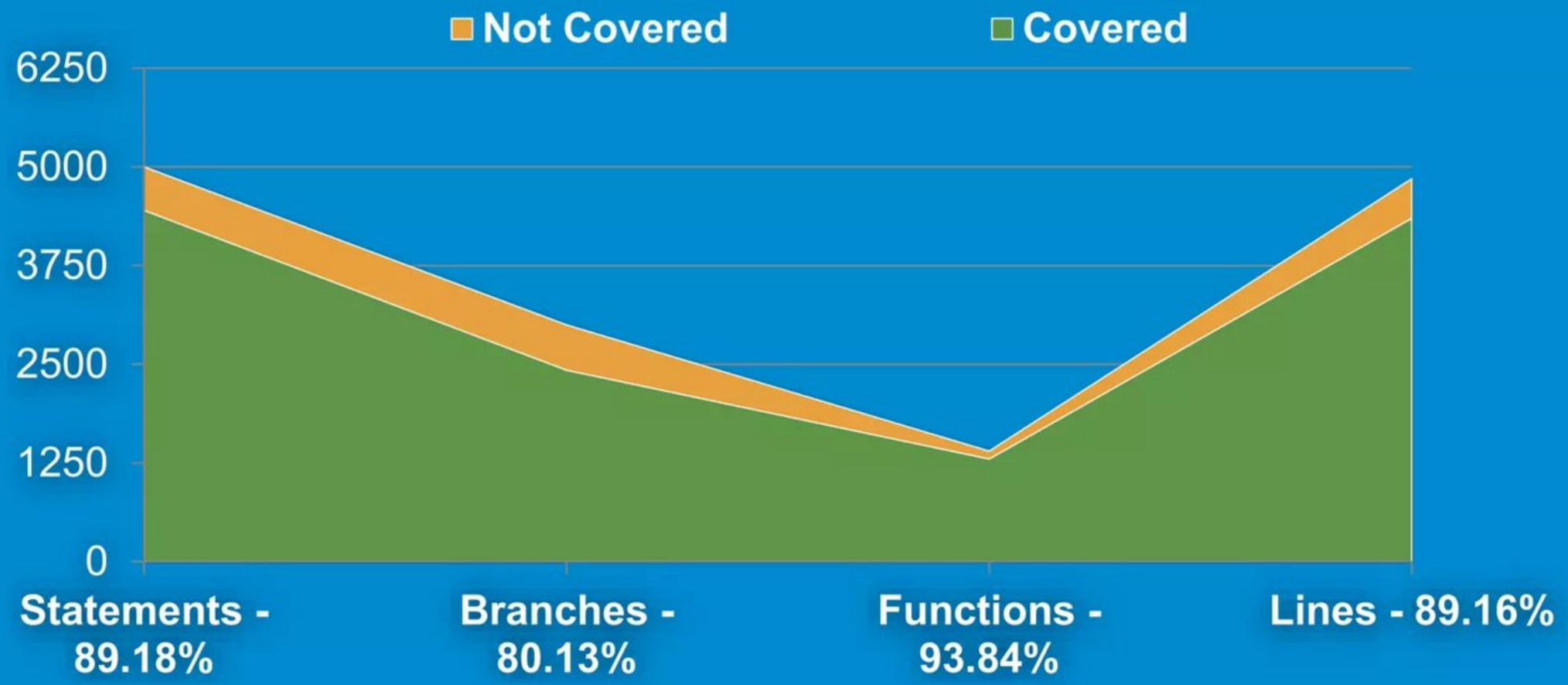
2.

1. Make real request to external service
2. Compare response and fixtures data

Testing Pyramid in Practice



Test Coverage in Practice



Links & Additions

https://github.com/archik408/GROCON19

- https://medium.com/front-end-in-regions-grodno
  - Lition, написанный на React.js
  - Доступность веб-интерфейсов. Что это и как?

- https://medium.com/lition-blog
  - Lition and Reactjs [DevBlog]
  - Layout of Trading Platform [DevBlog]
  - What does blockchain mean for frontenders? [DevBlog]
  - Tests & Docs for Lition frontend [DevBlog]

Thank You and Have fun!

Frame from «Game of Thrones» series