



How software architecture defines boundaries and APIs bridge them

Fabian Mauz

Architecture

- Definition and Motivation
- Example : Microservices
- Example : Ports and Adapters

Architecture

- Definition and Motivation
- Example : Microservices
- Example : Ports and Adapters

APIs

- A definition from reality
- Examples : Java Interface & REST
- Open API

Architecture

- Definition and Motivation
- Example : Microservices
- Example : Ports and Adapters

APIs

- A definition from reality
- Examples : Java Interface & REST
- Open API

Tools & Tipps

- Security
- Testing APIs with good architecture
- Github Actions

The software architecture of a program or computing system is the **structure or structures** of the system, which comprise software **components**, the externally visible **properties** of those components, and the **relationships** among them^[1]

[1] L. Bass, P. Clements and R. Kazman. *Software Architecture in Practice*. Addison Wesley, **1999**, ISBN 0-201-19930-0.

The software architecture of a program or computing system is the **structure or structures** of the system, which comprise software **components**, the externally visible **properties** of those components, and the **relationships** among them^[1]

[1] L. Bass, P. Clements and R. Kazman. *Software Architecture in Practice*. Addison Wesley, **1999**, ISBN 0-201-19930-0.

What is software architecture?



Martin Fowler

*„The goal of software architecture is to minimize the human resources required to **build** and **maintain** the required system“*

Robert C. Martin, Clean Architecture, 2017



Robert C. Martin

„Software architecture is those decisions which are both **important** and **hard to change** “

Martin Fowler

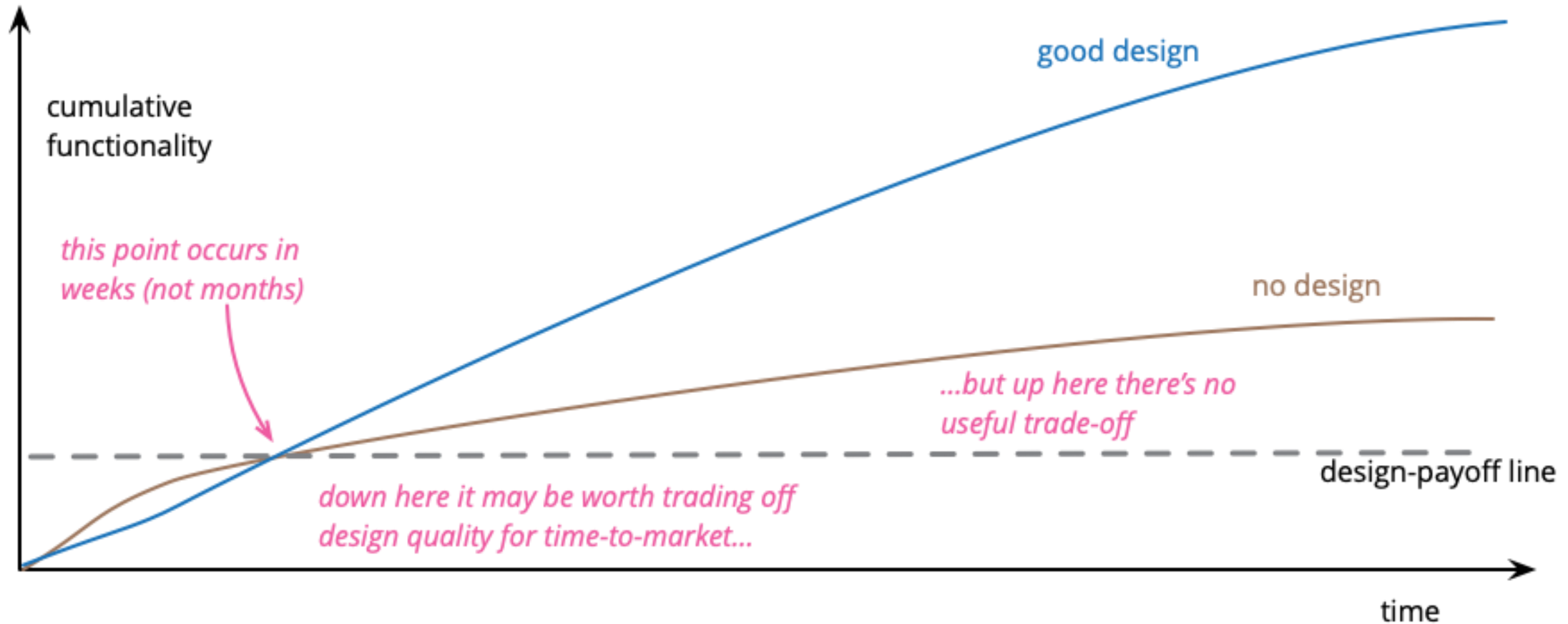
„Architecture represents the significant design decisions that **shape** a **system**, where **significant** is measured by **cost of change** “

Grady Booch



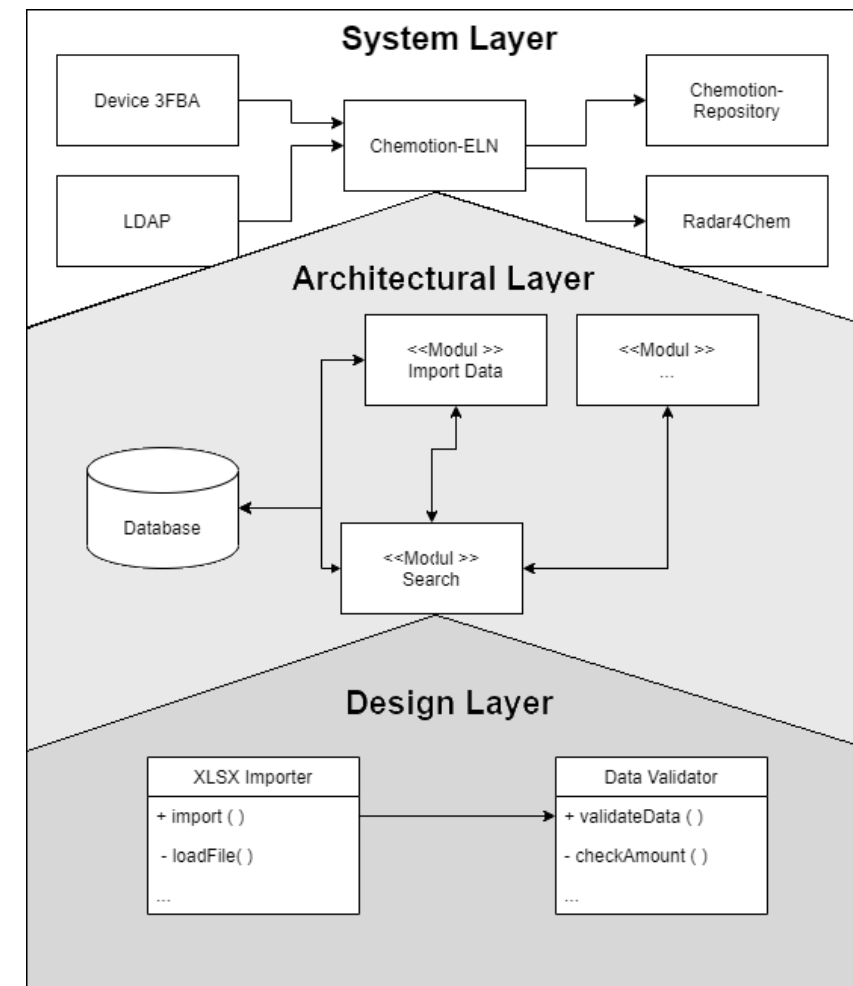
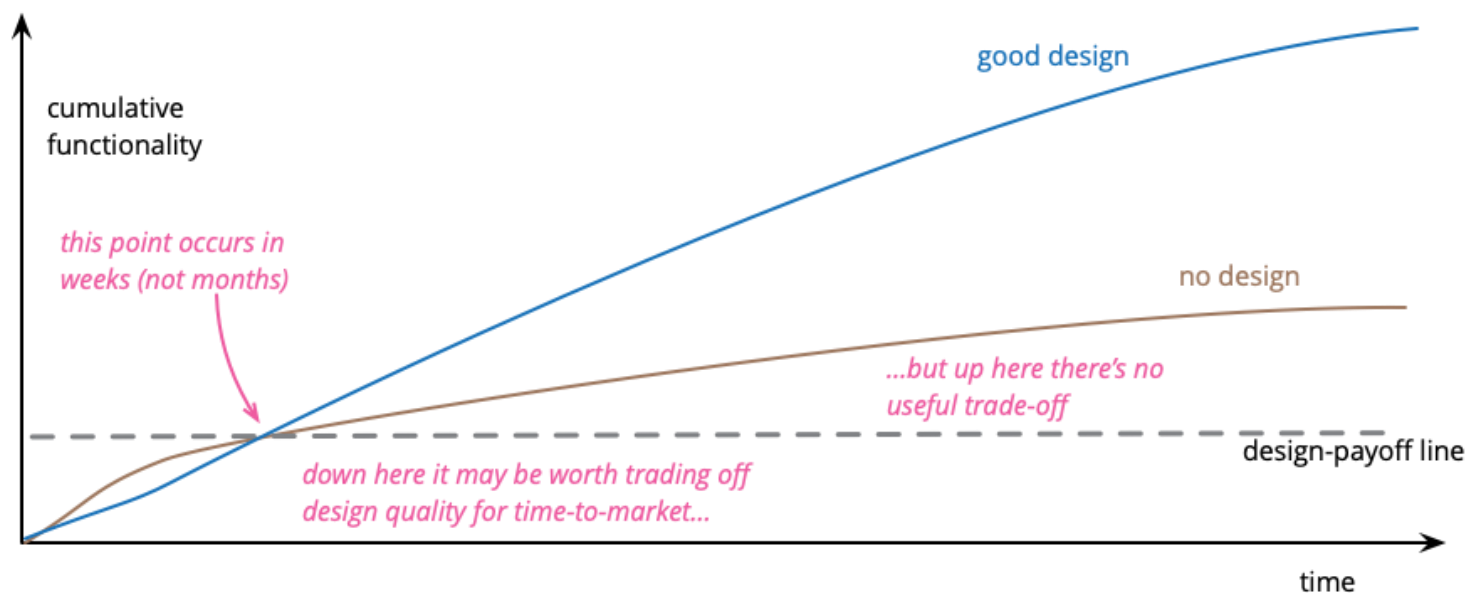
Grady Booch

Why thinking about architecture?



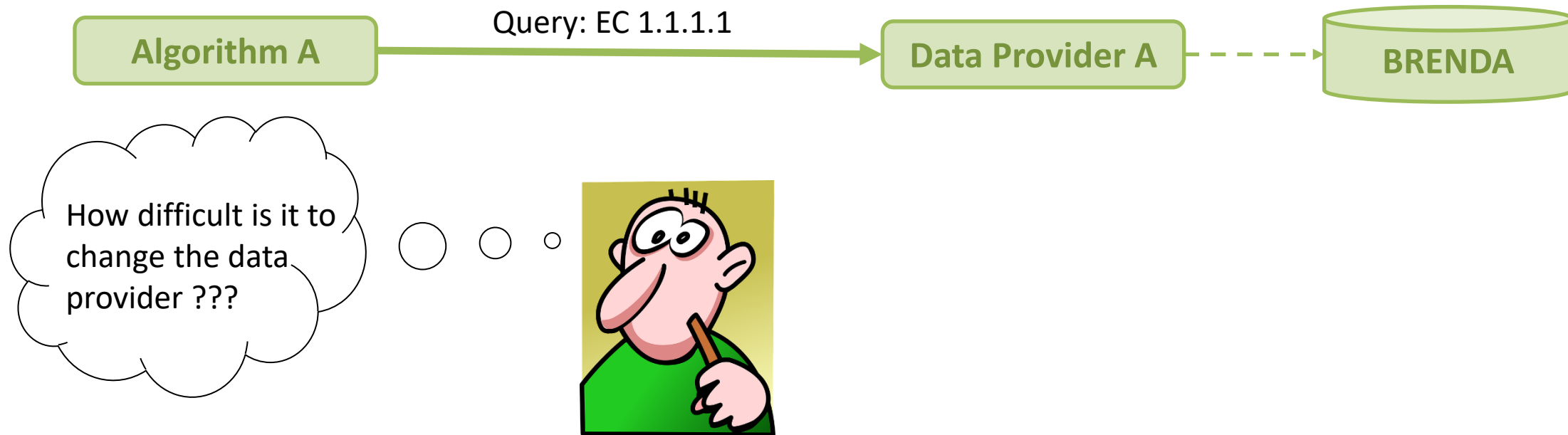
M. Fowler, *Design Stamina Hypothesis*, 2007

Why thinking about architecture?



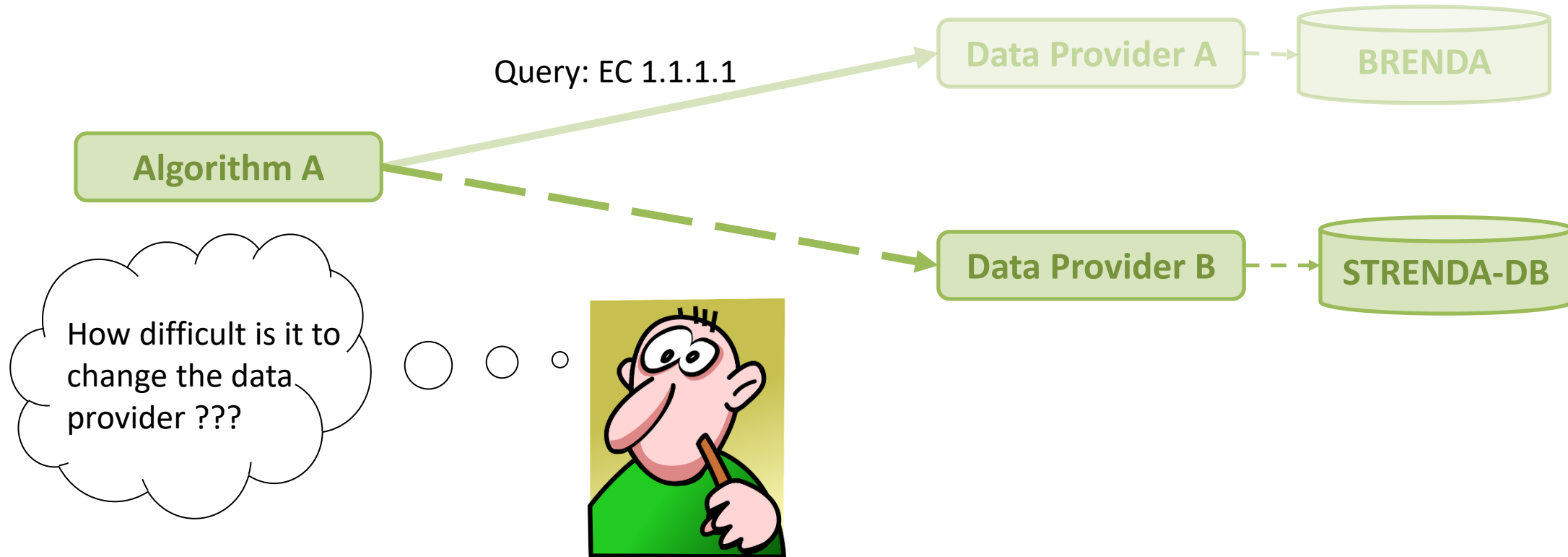
System \neq Architecture \neq Design ???

One main problem to change software are **dependencies**



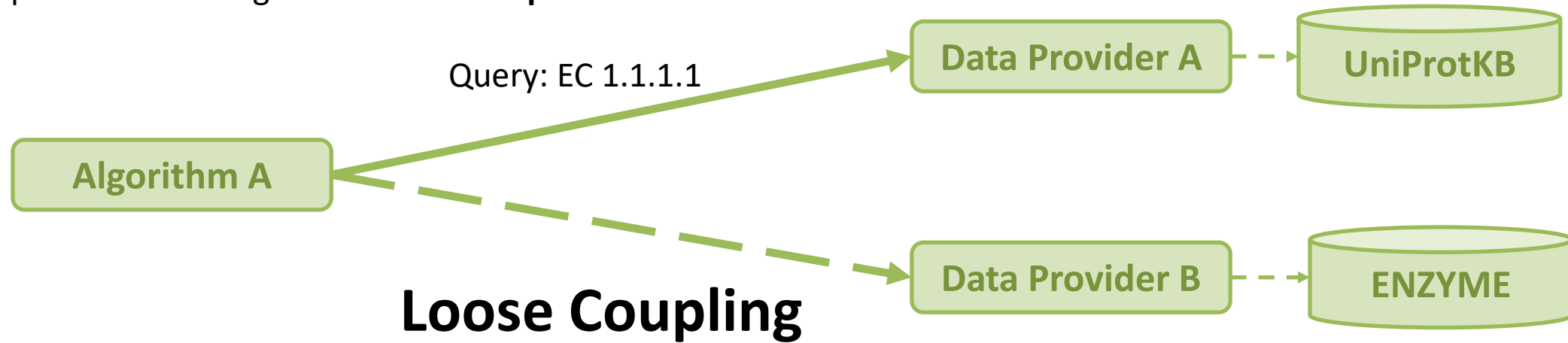
Dependencies

One main problem to change software are **dependencies**

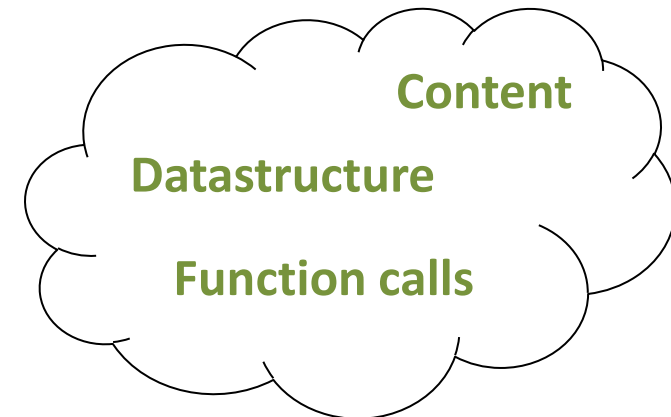
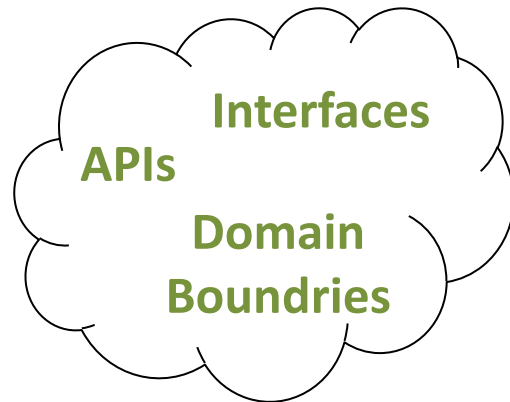


Coupling

One main problem to change software are **dependencies**



Loose Coupling



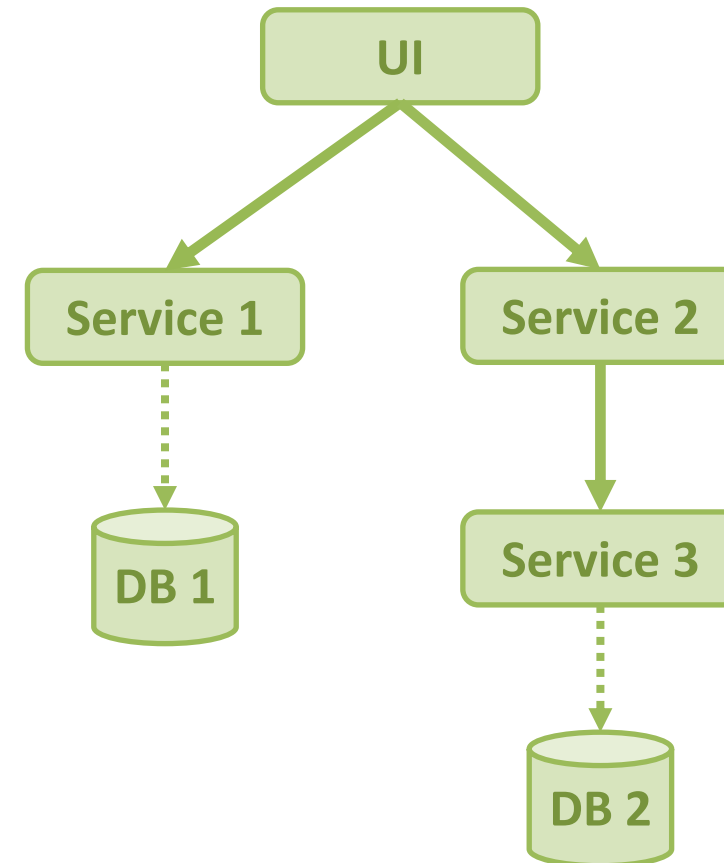
- Small independent, specialized services
- Own isolated data persistence
- Communication takes place via well-defined APIs

The good

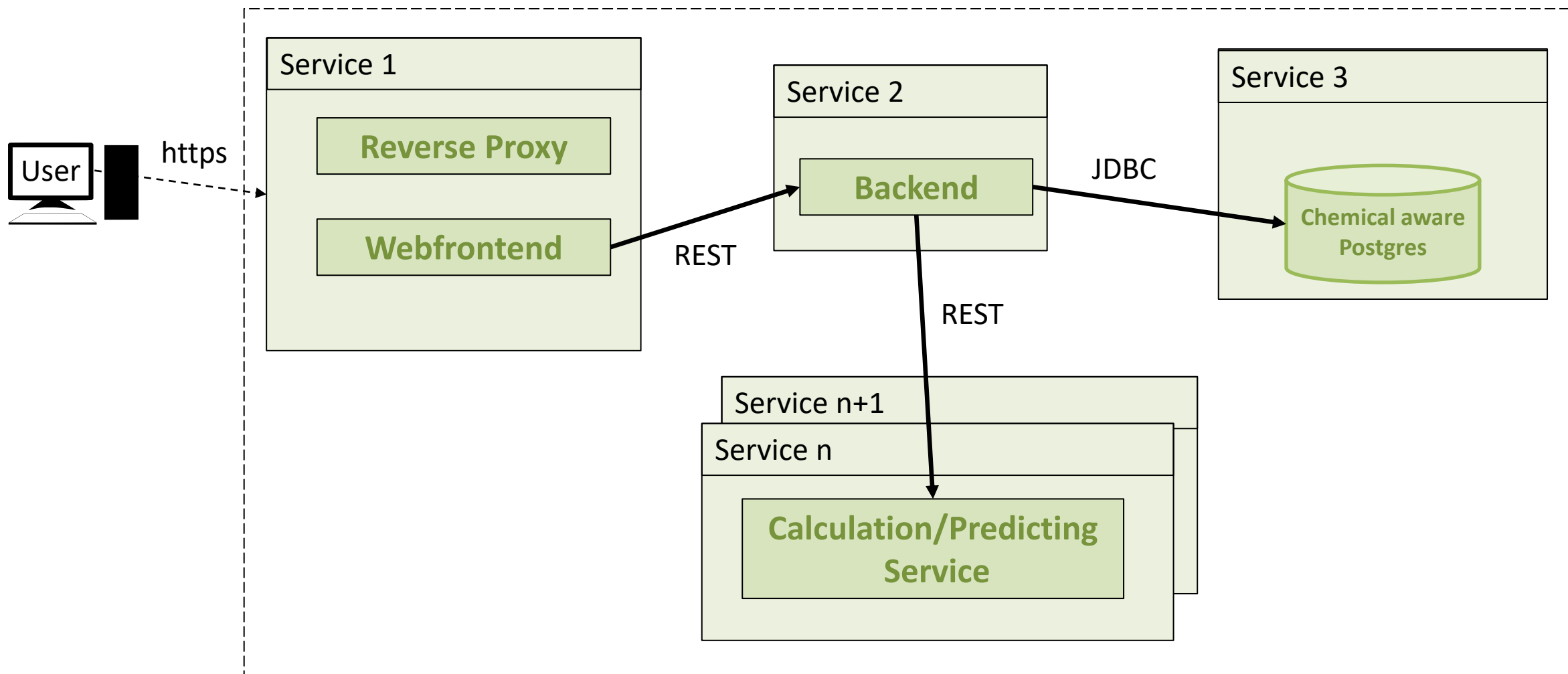
- „Small“ codebase, can be implemented independently
- High scalability
- Technological flexibility
- Reliability

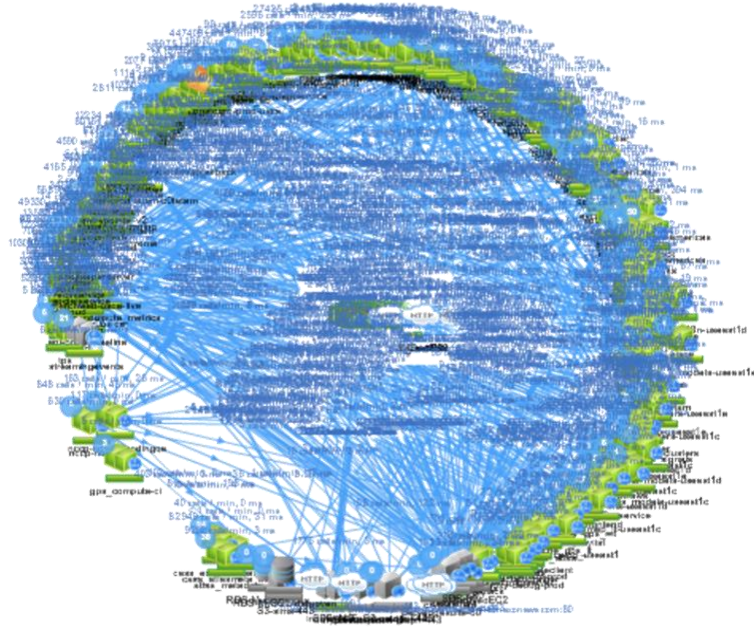
The bad

- More complex than monolith
- Deployment more complex
- Data Consistency
- Communication slower



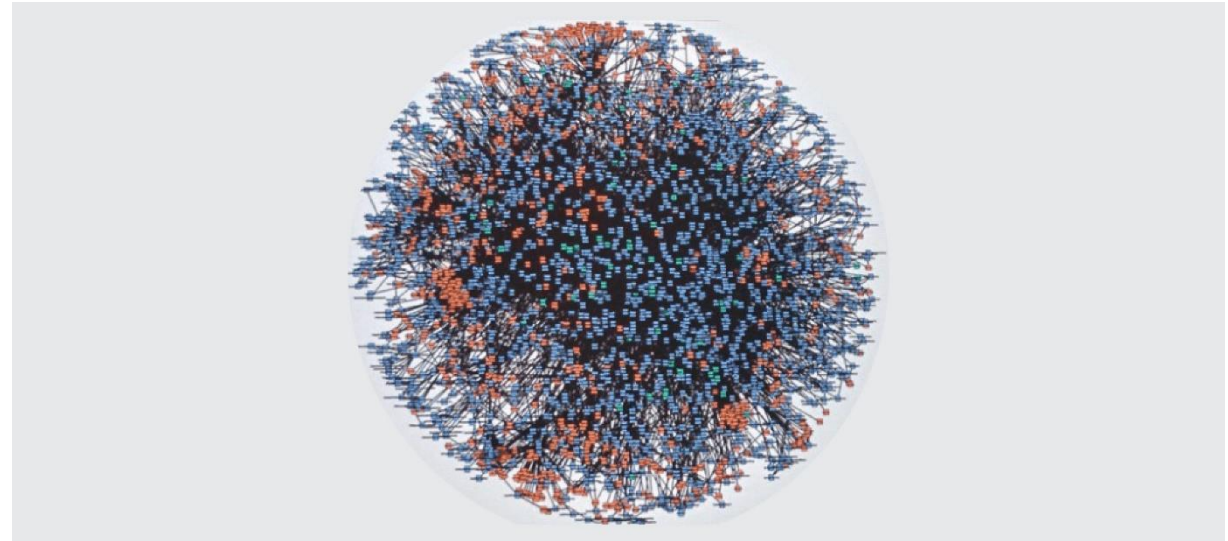
O.Al-Debagy, P.Martinek, *A Comparative Review of Microservices and Monolithic Architectures*, 2018





Netflix-Death-Star Architecture

Amazon claims to have reduced the cost of its video streaming service by 90% by switching from a microservice/serverless architecture to a monolith

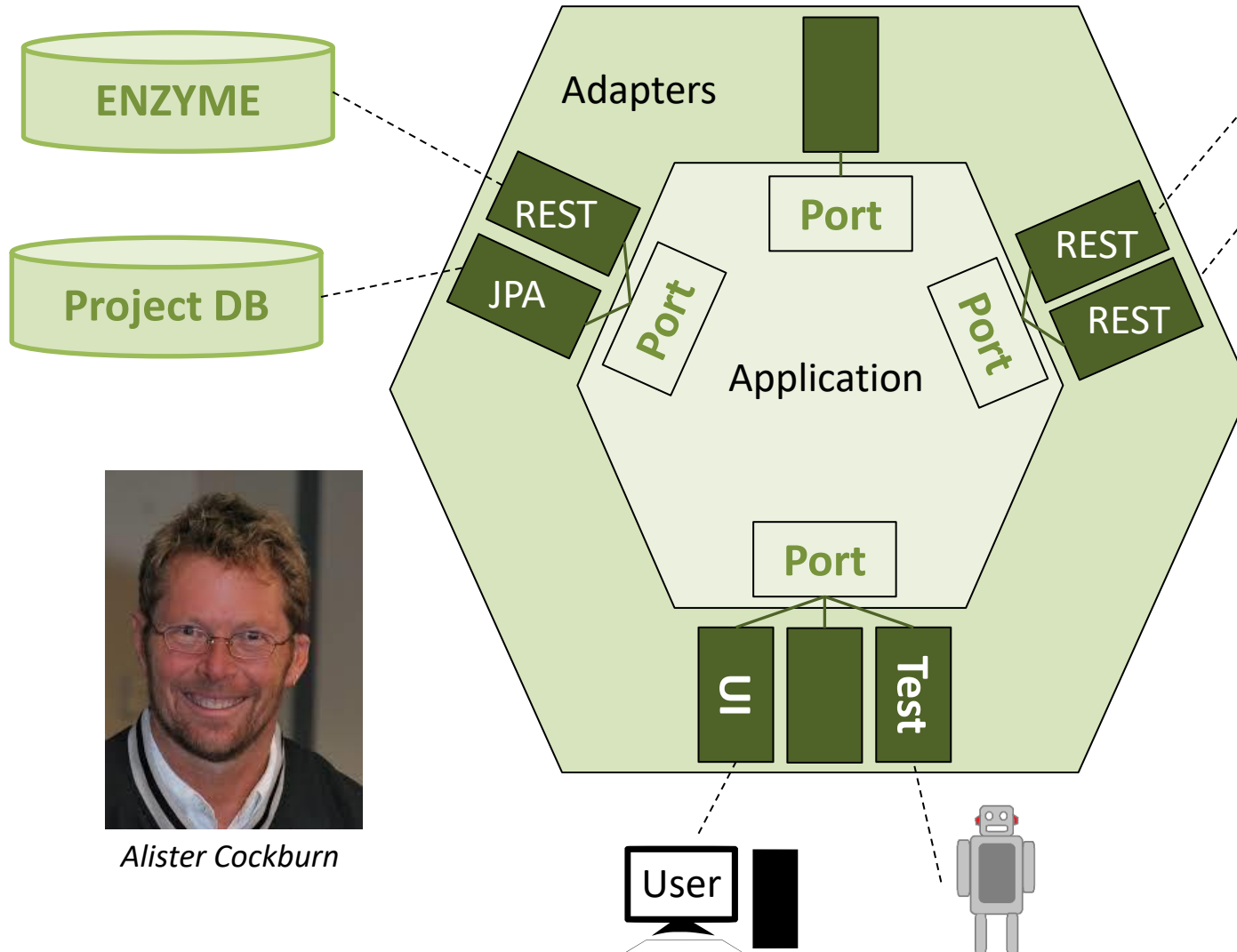


Amazon, 2000

„Microservices and serverless components are tools that do work at high scale, but whether to use them over monolith has to be made on a case-by-case basis“

Marcin Kolney, Amazon Prime Senior Software Development Engineer

Ports and Adapters/Hexagonal architecture



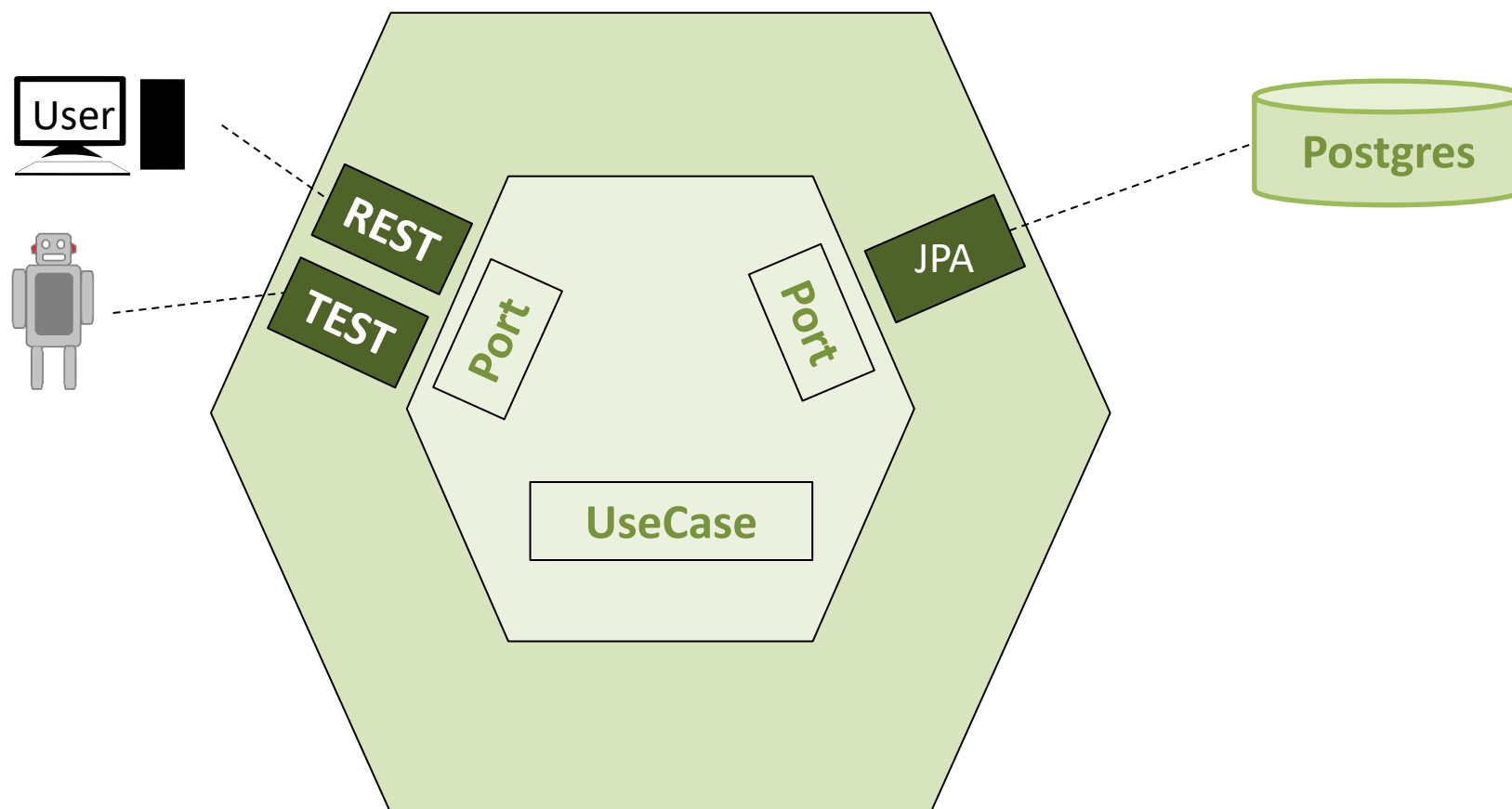
Alister Cockburn

Core Ideas

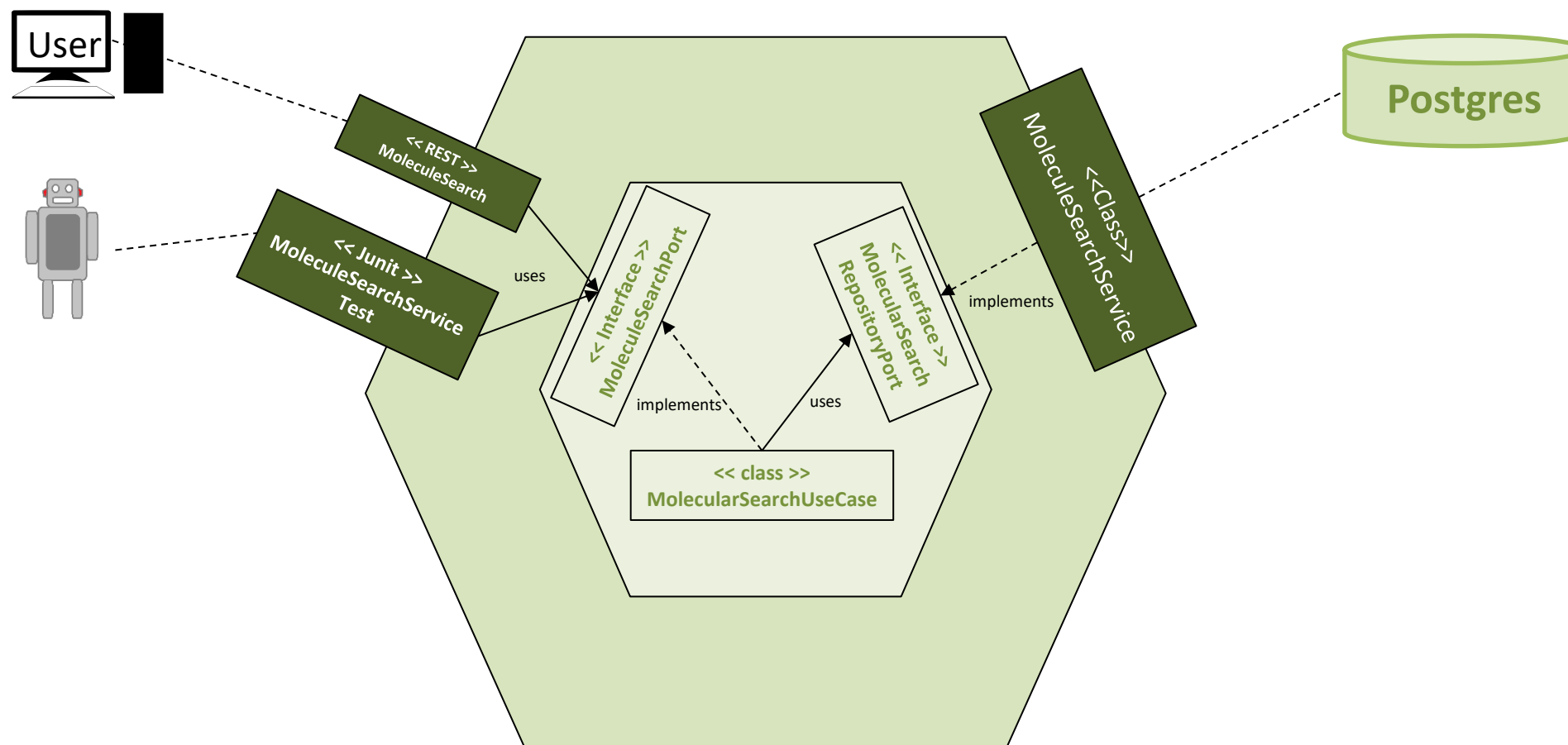
- Business logic is **isolated** from infrastructure
- Changing infrastructure does not effect BL
- Infrastructure and BL communicate with **ports** and **adapters** (loose coupling)
- **Primary** (triggers an usecase) and **Secondary Actors**

<https://alistair.cockburn.us/hexagonal-architecture/>, 2005

UseCase : Search for molecule with substructure search



UseCase : Search for molecule with substructure search



Ports and Adapters – Example

Adaper – Junit Test

```
@ExtendWith(PostgresqlContainerExtension.class)
@ExtendWith(ArquillianExtension.class)
public class MolecularSearchUseCaseTest extends TestBase {

    @Inject
    private MolecularSearchUseCasePort port;

    @Test
    public void find_Reaction_with_Valid_Smile() {
        List<Molecule> reactions = port.getMoleculesBySubstructure("O=Cc1ccccc1")
        Assert.assertEquals(1, reactions.size());
    }

    @Deployment
    public static WebArchive createDeployment() {
        return prepareDeployment("MolecularSearchUseCaseTest.war")
            .addClass(MolecularSearchServiceStub.class)
            .addClass(MolecularSearchUseCase.class);
    }
}
```



It's me again.

Still thinking about how
difficult it is to change
the dependency ???

Adaper – REST

```
@Path("/molecules")
@Stateless
public class MoleculeSearch {

    @Inject
    private MolecularSearchUseCasePort port;

    private static final Logger LOGGER = LogManager.getLogger();
    ObjectMapper jsonSerializer = new ObjectMapper();

    @GET
    @Produces("text/json")
    public Response getMolecule(
        @QueryParam("smarts") String smiles,
        @QueryParam("search_type") String searchType,
        @QueryParam("similarity_value") float similarityValue
    ) throws Exception {
        try {
            MoleculeSearchType.valueOf(searchType);
        } catch (Exception e) {
            return Response
                .status(500)
                .header("error-message", "param 'search_type' " + searchType)
                .header("Access-Control-Allow-Origin", "*")
                .build();
        }
        List<Molecule> molecules = port.getMoleculesBySubstructure(smiles);

        return Response
            .ok(jsonSerializer.writeValueAsString(molecules))
            .build();
    }
}
```


Ports and Adapters – Example

Adaper – Junit Test

```
@ExtendWith(PostgresqlContainerExtension.class)
@ExtendWith(ArquillianExtension.class)
public class MolecularSearchUseCaseTest extends TestBase {

    @Inject
    private MolecularSearchUseCasePort port;

    @Test
    public void find_Reaction_with_Valid_Smile() {
        List<Molecule> reactions = port.getMoleculesBySubstructure("O=Cc1ccccc1");
        Assert.assertEquals(1, reactions.size());
    }

    @Deployment
    public static WebArchive createDeployment() {
        return prepareDeployment("MolecularSearchUseCaseTest.war")
            .addClass(MolecularSearchServiceStub.class)
            .addClass(MolecularSearchUseCase.class);
    }
}
```



Wow
That was easy !!!

Adaper – REST

```
@Path("/molecules")
@Stateless
public class MoleculeSearch {

    @Inject
    private MolecularSearchUseCasePort port;

    private static final Logger LOGGER = LogManager.getLogger();
    ObjectMapper jsonSerializer = new ObjectMapper();

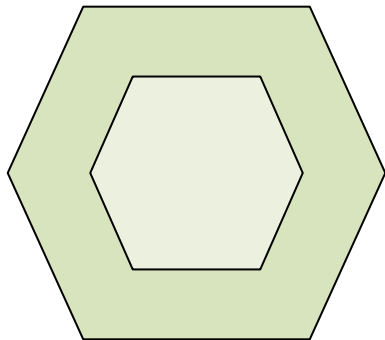
    @GET
    @Produces("text/json")
    public Response getMolecule(
        @QueryParam("smarts") String smiles,
        @QueryParam("search_type") String searchType,
        @QueryParam("similarity_value") float similarityValue
    ) throws Exception {
        try {
            MoleculeSearchType.valueOf(searchType);
        } catch (Exception e) {
            return Response
                .status(500)
                .header("error-message", "param 'search_type' " + searchType)
                .header("Access-Control-Allow-Origin", "*")
                .build();
        }
        List<Molecule> molecules = port.getMoleculesBySubstructure(smiles);

        return Response
            .ok(jsonSerializer.writeValueAsString(molecules))
            .build();
    }
}
```


API – A practical definition

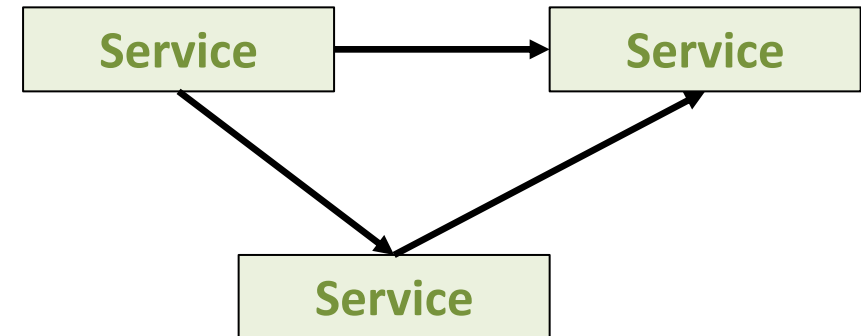
Both architectures define **boundaries**

Adapters and application are
conceptually separated from each other



Communication over a
function call defined by an
interface

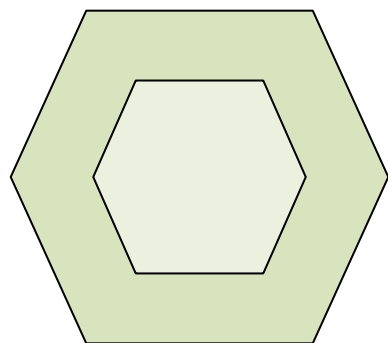
Services are conceptually and
often physically separated
from each other



Communication over a
network based protocol like
REST or JDBC

Both architectures define **boundaries**

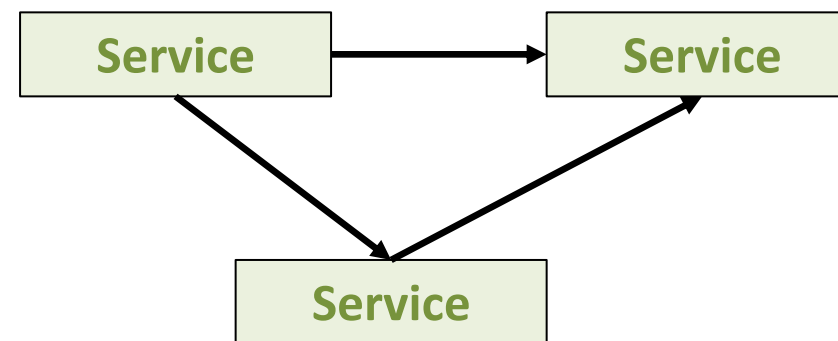
Adapters and application are conceptually separated from each other



Communication over a function call defined by an interface

Can we find a definition that does justice to both ??

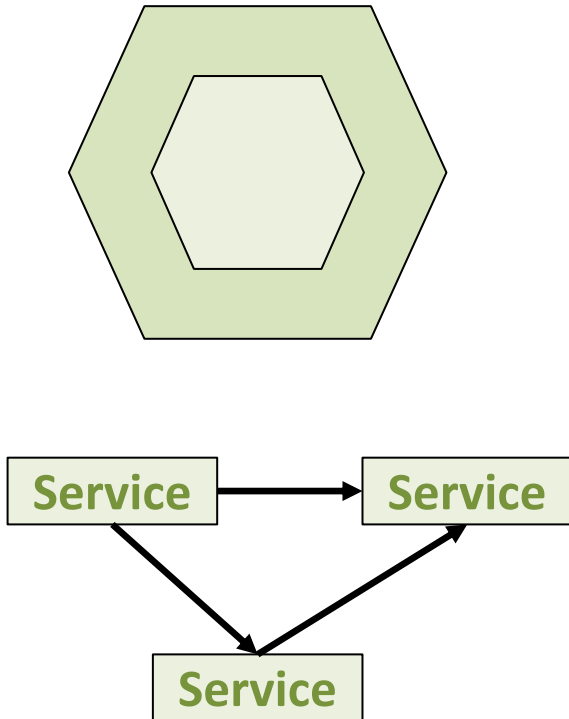
Services are conceptually and often physically separated from each other



Communication over a network based protocol like REST or JDBC

API – A practical definition

API – Application programming interface



- both represent an abstraction of the underlying implementation → Domain
- both introduce a specification of the required and supplied data → Signature
- both define a semantics or a behavior of the exchange → Protocol

API - Java function call defined by an interface

The keywords and the environment (java)
defines the protocol

The naming of the method and
parameters indicates it's functionality

Protocol

Domain

```
public interface MolecularSearchRepositoryPort {  
    List<Molecule> getMoleculesBySubstructure(String smarts);  
}
```

Signature

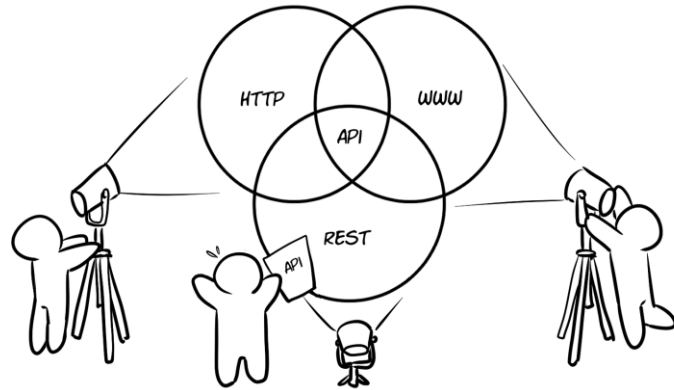
The method name, input and output
arguments and their types

- There is no statement about **how** something is done only **what**
- There is no indication of whether the request is idempotent. (Idempotent calls would return each time the same result. This can be defined in part by the protocol type)

API – REST Definition

REST : Representational State Transfer

Not a standard or protocol, like HTTP but more of a style



Set of rules/constraints

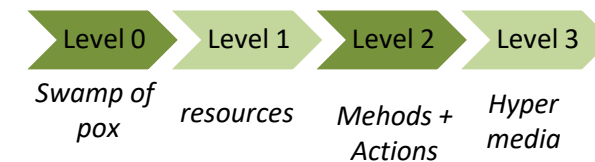
- Client-server
- **Stateless systems**
- Cache
- **Uniform interface**
- Layered system
- **Code on demand**

Protocol: Not strictly defined but in reality, 99% http(s)

Domain: domain specific resources

Signature: Query parameters, server address+path and Response

Maturity levels of REST



R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, 2000

M. Amundsen, *Design and Build Great Web APIs*, 2020

API – REST Example - Browser

pubchem - searching for synonyms

<https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/sid/10000/synonyms/json>

↑ protocol ↑ server ↑ path

InformationList:

Information:

0:

SID: 10000

Synonym:

0: "dihydroergotamine"

1: "511-12-6"

2: "C07798"

ChEMBL - searching for molecules with name pattern

https://www.ebi.ac.uk/chembl/api/data/target?pref_name__contains=cyclin

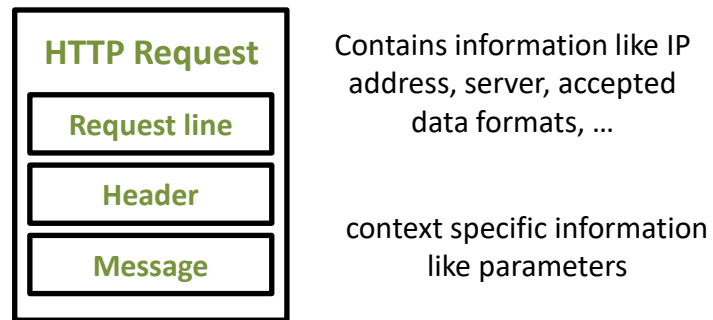
↑ protocol ↑ server ↑ path ↑ parameter

```
<?xml version='1.0' encoding='utf-8'?>
<response>
  <page_meta>
    <limit>20</limit>
    <next>/chembl/api/data/target?limit=20&offset=20&pref_name__contains=cyclin</next>
    <offset/>
    <previous/>
    <total_count>30</total_count>
  </page_meta>
  <targets>
    <target>
      <cross_references>
        <target>
          <xref_id>P30277</xref_id>
          <xref_name/>
          <xref_src>canSAR-Target</xref_src>
        </target>
      </cross_references>
      <organism>Rattus norvegicus</organism>
      <pref_name>G2/mitotic-specific cyclin B1</pref_name>
      <species_group_flag/>
      <target_chembl_id>CHEMBL2093</target_chembl_id>
      <target_components>
        <target_component>
          <accession>P30277</accession>
          <component_description>G2/mitotic-specific cyclin-B1</component_description>
          <component_id>430</component_id>
        </target_component>
      </target_components>
    </target>
  </targets>
</response>
```

<https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest-tutorial>

<https://www.ebi.ac.uk/chembl/ws>

API – REST Example javascript and python



The request method is also defined in the header

- **GET** : Used to receive information
- **POST**: send information to server
- **PUT**: update information on server
- **DELETE**: remove information on server

API – REST Example javascript and python



Contains information like IP address, server, accepted data formats, ...

context specific information like parameters

```
const promise = () => fetch('/api/v1/wellplates/', {
  credentials: 'same-origin',
  method: 'post',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(wellplate.serialize())
})
```

Example java script

```
import requests

requests.post(
  '/api/v1/wellplates/',
  data=json.dumps({'wellplate': ' ... ' }),
  headers={'Content-type': 'application/json'}
);
```

Example python

The request method is also defined in the header

- **GET** : Used to receive information
- **POST**: send information to server
- **PUT**: update information on server
- **DELETE**: remove information on server

Both create



▼ POST
Scheme: https
Host: uncharted.chemotion.ibcs.kit.edu
Filename: /api/v1/wellplates/
Address: 141.52.79.135:443

Status	201 Created ?
Version	HTTP/2
Transferred	18.16 kB (17.1%)
Referrer Policy	strict-origin
Request Priority	Highest
DNS Resolution	System

▼ Request Headers (1.113 kB)

- Accept: application/json
- Accept-Encoding: gzip, deflate, br
- Accept-Language: de,en;q=0.7,en-US;q=0.3
- Cache-Control: no-cache
- Connection: keep-alive
- Content-Length: 10360
- Content-Type: application/json
- Cookie: _chemotion_session=MfYguXBdPrVIII...PaQaMPueaYKdDDt86WuVa2fkWtIANbIc7ODd...

JSON

```
attachments: []
collection_id: 778
container: {...}
description: {...}
height: 8
id: "121aaa81-9504-11ef-bdec-47bddfccc56d"
is_new: true
name: "Hallo Wellplate!!!"
```

API – REST Example

<https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/sid/10000/synonyms/json>



https://www.ebi.ac.uk/chembl/api/data/target?pref_name__contains=cyclin

But how do i know
what paths or
parameters i have to
enter ???

API – Open API

<https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/sid/10000/synonyms/json>



https://www.ebi.ac.uk/chembl/api/data/target?pref_name__contains=cyclin

But how do i know
what paths or
parameters i have to
enter ???

Many but not all web services
offer sufficient API
documentation and they vary in
form and quality

<https://www.openapis.org/>

API – Open API

<https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/sid/10000/synonyms/json>



https://www.ebi.ac.uk/chembl/api/data/target?pref_name__contains=cyclin

But how do i know
what paths or
parameters i have to
enter ???

OpenAPI

**The OpenAPI Specifications provides a formal
standard for describing HTTP APIs**

<https://www.openapis.org/>

API – Open API Specification 3.0

```
1 openapi: 3.0.3
2 info:
3   title: Molecule Search API
4   version: 0.1.0
5 paths:
6   /molecules/:
7     get:
8       summary: Search for a molecule
9       description: 'Search for a molecule'
10      operationId: moleculeSearchID
11      parameters:
12        - in: query
13          name: smarts
14          example: CCO
15          schema:
16            type: string
17        - in: query
18          name: search_type
19          schema:
20            type: string
21            enum:
22              - SUBSTRUCTURE
23              - EXACT
24              - SIMILARITY
25            default: EXACT
26        - in: query
27          name: similarity_value
28          schema:
29            type: number
30            format: float
31            minimum: 0
32            maximum: 1
33            example: 0.5
34      responses:
35        '200':
36          description: successful operation
37          content:
38            application/json:
39              schema:
40                $ref: '#/components/schemas/Molecule'
41            application/xml:
42              schema:
43                $ref: '#/components/schemas/Molecule'
44        '404':
45          description: Molecule not found
```

```
46 components:
47   schemas:
48     Molecule:
49       type: object
50       properties:
51         id:
52           type: integer
53           format: int64
54           example: 10
55         molecule:
56           type: string
57           description: Molecule description as sdf or mol file
58         iupac:
59           type: string
60         smiles:
61           type: string
62         chebiid:
63           type: string
64         inchi:
65           type: string
```


API – Open API Specification 3.0

```
1 openapi: 3.0.3
2 info:
3   title: Molecule Search API
4   version: 0.1.0
5 paths:
6   /molecules/:
7     get:
8       summary: Search for a molecule
9       description: 'Search for a molecule'
10      operationId: moleculeSearchID
11      parameters:
12        - in: query
13          name: smiles
14          example: CCO
15          schema:
16            type: string
17        - in: query
18          name: search_type
19          schema:
20            type: string
21            enum:
22              - SUBSTRUCTURE
23              - EXACT
24              - SIMILARITY
25            default: EXACT
26        - in: query
27          name: similarity_value
28          schema:
29            type: number
30            format: float
31            minimum: 0
32            maximum: 1
33            example: 0.5
34      responses:
35        '200':
36          description: successful operation
37          content:
38            application/json:
39              schema:
40                $ref: '#/components/schemas/Molecule'
41            application/xml:
42              schema:
43                $ref: '#/components/schemas/Molecule'
44        '404':
45          description: Molecule not found
```

```
@Path("/molecules")
@Stateless
public class MoleculeSearch {

    @Inject
    private MoleculeSearchPort port;

    private static final Logger LOGGER = LogManager.getLogger();
    ObjectMapper jsonSerializer = new ObjectMapper();

    @GET
    @Produces("text/json")
    public Response getMolecule(
        @QueryParam("smiles") String smiles,
        @QueryParam("search_type") String searchType,
        @QueryParam("similarity_value") float similarityValue
    ) throws Exception {
        List<Molecule> molecules = port.getMoleculesBySubstructure(smiles);

        return Response
            .ok(jsonSerializer.writeValueAsString(molecules))
            .header("Access-Control-Allow-Origin", "*")
            .build();
    }
}
```

```
46 components:
47   schemas:
48     Molecule:
49       type: object
50       properties:
51         id:
52           type: integer
53           format: int64
54           example: 10
55         molecule:
56           type: string
57           description: Molecule description as sdf or mol file
58         iupac:
59           type: string
60         smiles:
61           type: string
62         chebiid:
63           type: string
64         inchi:
65           type: string
```

API – Open API Specification 3.0

```
1 openapi: 3.0.3
2 info:
3   title: Molecule Search API
4   version: 0.1.0
5 paths:
6   /molecules/:
7     get:
8       summary: Search for a molecule
9       description: 'Search for a molecule'
10      operationId: moleculeSearchID
11      parameters:
12        - in: query
13          name: smarts
14          example: CCO
15          schema:
16            type: string
17        - in: query
18          name: search_type
19          schema:
20            type: string
21            enum:
22              - SUBSTRUCTURE
23              - EXACT
24              - SIMILARITY
25            default: EXACT
26        - in: query
27          name: similarity_value
28          schema:
29            type: number
30            format: float
31            minimum: 0
32            maximum: 1
33            example: 0.5
34      responses:
35        '200':
36          description: successful operation
37          content:
38            application/json:
39              schema:
40                $ref: '#/components/schemas/Molecule'
41            application/xml:
42              schema:
43                $ref: '#/components/schemas/Molecule'
44        '404':
45          description: Molecule not found
```

What's in here for me???

```
46 components:
47   schemas:
48     Molecule:
49       type: object
50       properties:
51         id:
52           type: integer
53           format: int64
54           example: 10
55         molecule:
56           type: string
57           description: Molecule description as sdf or mol file
58         iupac:
59           type: string
60         smiles:
61           type: string
62         chebiid:
63           type: string
64         inchi:
65           type: string
```

Automatic code generation

- Automatic code generation for clients and servers for various languages
- Automatic documentation generation for different styles
- Even database shemas could be generated

```
28 @Path("/molecules/")
29 @Api(value = "/", description = "")
30 public interface DefaultApi {
31
32     /**
33      * Search for a molecule
34      *
35      * Search for a molecule
36      *
37      */
38     @GET
39
40     @Produces({ "application/json", "application/xml" })
41     @ApiOperation(value = "Search for a molecule", tags={ })
42     @ApiResponse(value = {
43         @ApiResponse(code = 200, message = "successful operation", response = Molecule.class),
44         @ApiResponse(code = 404, message = "Molecule not found") })
45     public Molecule moleculeSearchID(
46         @QueryParam("smiles") String smiles,
47         @QueryParam("search_type") @DefaultValue("EXACT") String searchType,
48         @QueryParam("similarity_value") @DecimalMin("0") @DecimalMax("1") Float similarityValue);
49 }
```

```
11 public class Molecule {
12
13     @ApiModelProperty(example = "10", value = "")
14     private Long id;
15
16     @ApiModelProperty(value = "Molecule description as sdf or mol file")
17     /**
18      * Molecule description as sdf or mol file
19      */
20     private String molecule;
21
22     @ApiModelProperty(value = "")
23     private String iupac;
24
25     @ApiModelProperty(value = "")
26     private String smiles;
27
28     @ApiModelProperty(value = "")
29     private String chebiid;
30
31     @ApiModelProperty(value = "")
32     private String inchi;
```

Automatically generated java REST server endpoint

<https://openapi-generator.tech/>

Out of the box documentation

Molecule Search API 0.1.0 OAS 3.0

default ^

GET /molecules/ Search for a molecule ^

Search for a molecule

Parameters Try it out

Name	Description
smarts	Example : CCO
string	CCO
(query)	
search_type	Available values : SUBSTRUCTURE, EXACT, SIMILARITY Default value : EXACT
string	EXACT
(query)	
similarity_value	0.5
number(\$float)	
(query)	

Responses

Code	Description	Links
200	successful operation	No links

Media type

application/json

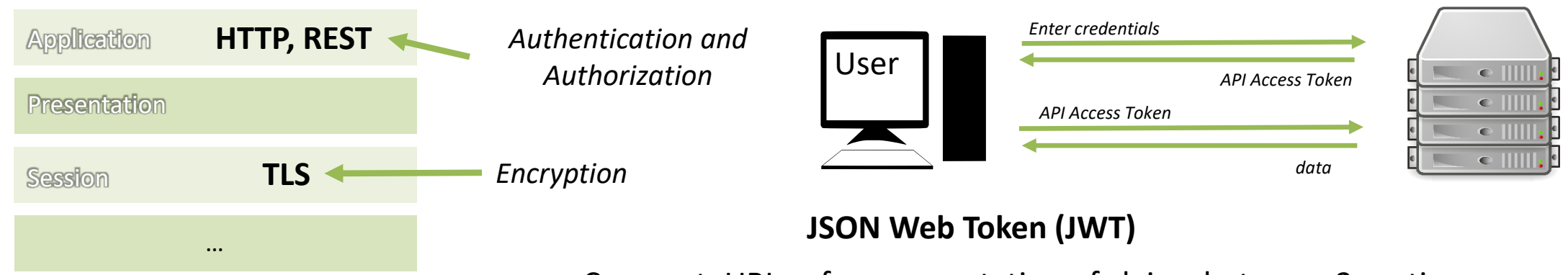
Schemas ^

```
Molecule {
  id integer($int64)
    example: 10
  molecule > [...]
  iupac > [...]
  smiles > [...]
  chebiid > [...]
  inchi > [...]
}
```

- Extensive documentation of the API routes
- Detailed description of the data resources
- Playground for trying out the API
- Ways to simulate various security mechanisms

<https://editor.swagger.io/>

API – Securing API endpoints



JSON Web Token (JWT)

Compact, URL-safe representation of claims between 2 parties

xxxx.yyyy.zzzz

Authentication < > Authorisation

Who

What

- Header: defines type of token and signing algorithm
- Payload: contains the claims
- Signature: Signature for validation

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

More sophisticated mechanisms:
OAuth2.0 , OpenID connect, ...

M.Jones, J. Bradley,N. Sakimura, <https://datatracker.ietf.org/doc/html/rfc7519>, 2015

M. Amundsen, Design and Build Great Web APIs, 2020, p. 207ff

API – Testing of APIs

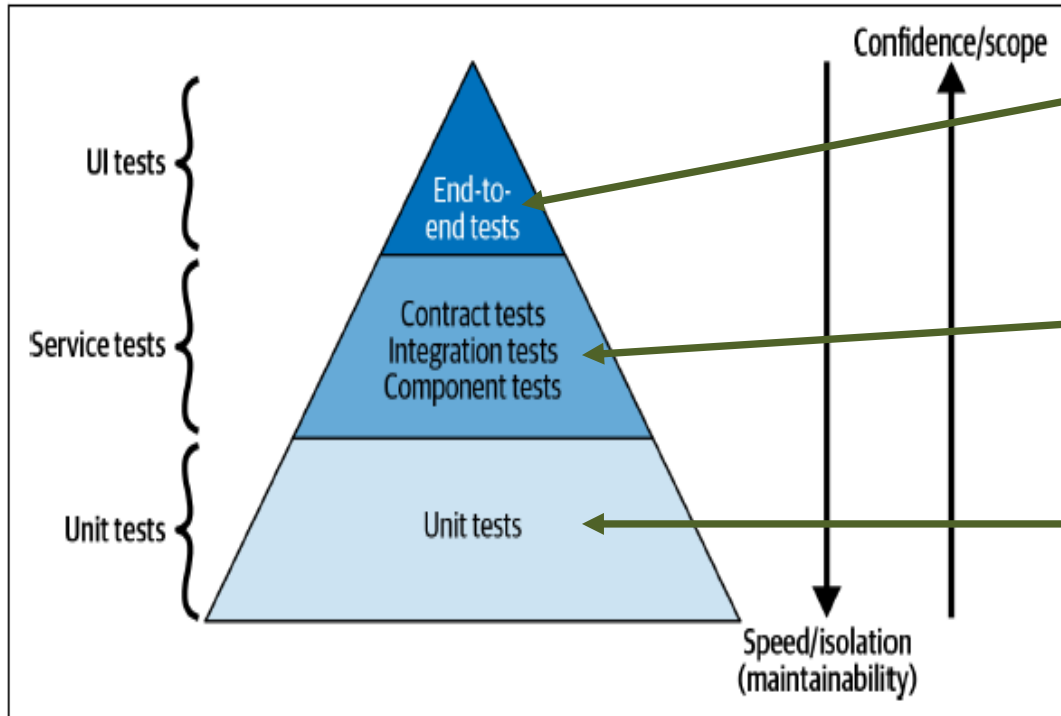


Figure 2-3. The test pyramid, showing the proportion of tests desired

- *Simulates a client and a whole usecase*
- *Frameworks: Cypress, Playwright, ...*
- *By using the Hexagon architecture, components could be mocked*
- *Using frameworks could help: testcontainers, virtual archives, ...*
- *Usually APIs are tested in upper layers*

!! All tests in all layers must be executed automatically regardless of the environment !!

“... Automate, customize and execute your software development workflows directly in your repository.”

- **Workflow:** configurable automated sequence of tasks
- **Event:** specific occurrence in your repository that triggers a workflow
- **Job:** collection of steps that are executed in a workflow
- **Action:** individual building blocks of a workflow
- **Runner:** computer on which your workflows are executed
- **Step:** smallest executable unit in a job

```
1  name: BioCasNavi - frontend CI
2
3  on:
4    push:
5      branches: [ "main" ]
6    pull_request:
7      branches: [ "main" ]
8
9  jobs:
10   test-job:
11     runs-on: ubuntu-latest
12     steps:
13       - name: Checkout
14         uses: actions/checkout@v3
15
16       - name: Use Node 20.x
17         uses: actions/setup-node@v3
18         with:
19           node-version: '20.x'
20
21       - name: Install dependencies
22         working-directory: ./frontend
23         run: npm install
24
25       - name: Test (jest and react-testing-library)
26         working-directory: ./frontend
27         run: npm test
28
29
30
```

<https://docs.github.com/de/actions>

Hundreds of provided actions for many areas

```
name: End-to-end tests
on: push
jobs:
  cypress-run:
    runs-on: ubuntu-22.04
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      # Install npm dependencies, cache them correctly
      # and run all Cypress tests
      - name: Cypress run
        uses: cypress-io/github-action@v6
```

```
steps:
  - uses: actions/checkout@main
  - uses: codecov/codecov-action@v4
    with:
      fail_ci_if_error: true # optional (default = false)
      files: ./coverage1.xml,./coverage2.xml # optional
      flags: unittests # optional
      name: codecov-umbrella # optional
      token: ${ secrets.CODECOV_TOKEN } # required
      verbose: true # optional (default = false)
```

```
test-job:
  runs-on: ubuntu-latest
  steps:
    - name: Checkout
      uses: actions/checkout@v3

    - name: Use Node 20.x
      uses: actions/setup-node@v3
      with:
        node-version: '20.x'

    - name: Cache npm dependencies
      uses: actions/cache@v2
      with:
        path: '~/.npm'
        key: ${ runner.os }-node-${ hashFiles('**/package-lock.json') }
        restore-keys: |
          ${ runner.os }-node-

    - name: Install dependencies
      working-directory: ./frontend
      run: npm install

    - name: Test (jest and react-testing-library)
      working-directory: ./frontend
      run: npm test
```

<https://github.com/sdras/awesome-actions>


```
test-job
Started 2m 5s ago

> Set up job
> Checkout
> Use Node 20.x

Cache npm dependencies
1 Run actions/cache@v2
7 Cache not found for input keys: Linux-node-, Linux-node-

Install dependencies
1 Run npm install
4
4 npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use this package. Please use the npmcli package instead.
5 npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
7 added 663 packages, and audited 664 packages in 32s
8 134 packages are looking for funding
9 run `npm fund` for details
10 found 0 vulnerabilities

Test (jest and react-testing-library)

Post Cache npm dependencies
1 Post job cleanup.
2 /usr/bin/tar --posix -z -cf cache.tgz -P -C /home/runner/work/BioCasNavi/BioCasNavi --files-from .
3 Cache Size: ~95 MB (99716364 B)
4 Cache saved successfully
5 Cache saved with key: Linux-node-
```

Execution without cached dependencies

```
test-job
succeeded 1 minute ago in 33s

> Set up job
> Checkout
> Use Node 20.x

Cache npm dependencies
1 Run actions/cache@v2
7 Received 33554432 of 99716364 (33.6%), 32.0 MBs/sec
8 Received 67108864 of 99716364 (67.3%), 32.0 MBs/sec
9 Received 78356236 of 99716364 (70.6%), 22.4 MBs/sec
10 Received 78356236 of 99716364 (70.6%), 16.8 MBs/sec
11 Received 78356236 of 99716364 (70.6%), 13.4 MBs/sec
12 Received 99716364 of 99716364 (100.0%), 17.8 MBs/sec
13 Cache Size: ~95 MB (99716364 B)
14 /usr/bin/tar -z -xf /home/runner/work/_temp/c5e3f198-3bff-41de-bef4-ddd40752663a/cache.tgz -P -C /home/runner/work/BioCasNavi/BioCasNavi
15 Cache restored successfully
16 Cache restored from key: Linux-node-

Install dependencies
12s

Test (jest and react-testing-library)
8s

Post Cache npm dependencies
2s
```

Execution with cached dependencies

Workflow execution time : **33s** vs. **3,6s**

I would like to thank

Prof. Ludger Wessjohann

Dr. Mehdi Davari

Dr. Frank Broda

NWC department & Dr. Steffen
Neumanns group



Funded by

