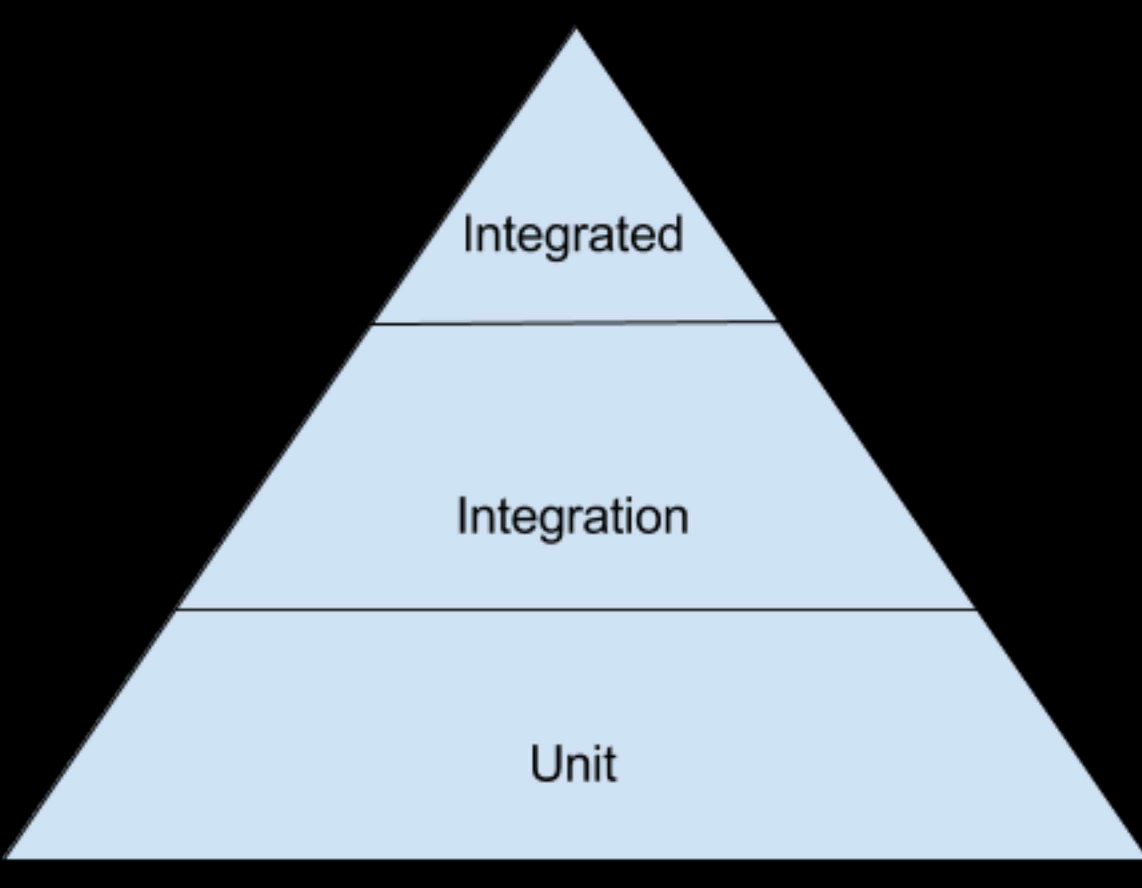
WORKSHOP TESTCONTAINERS

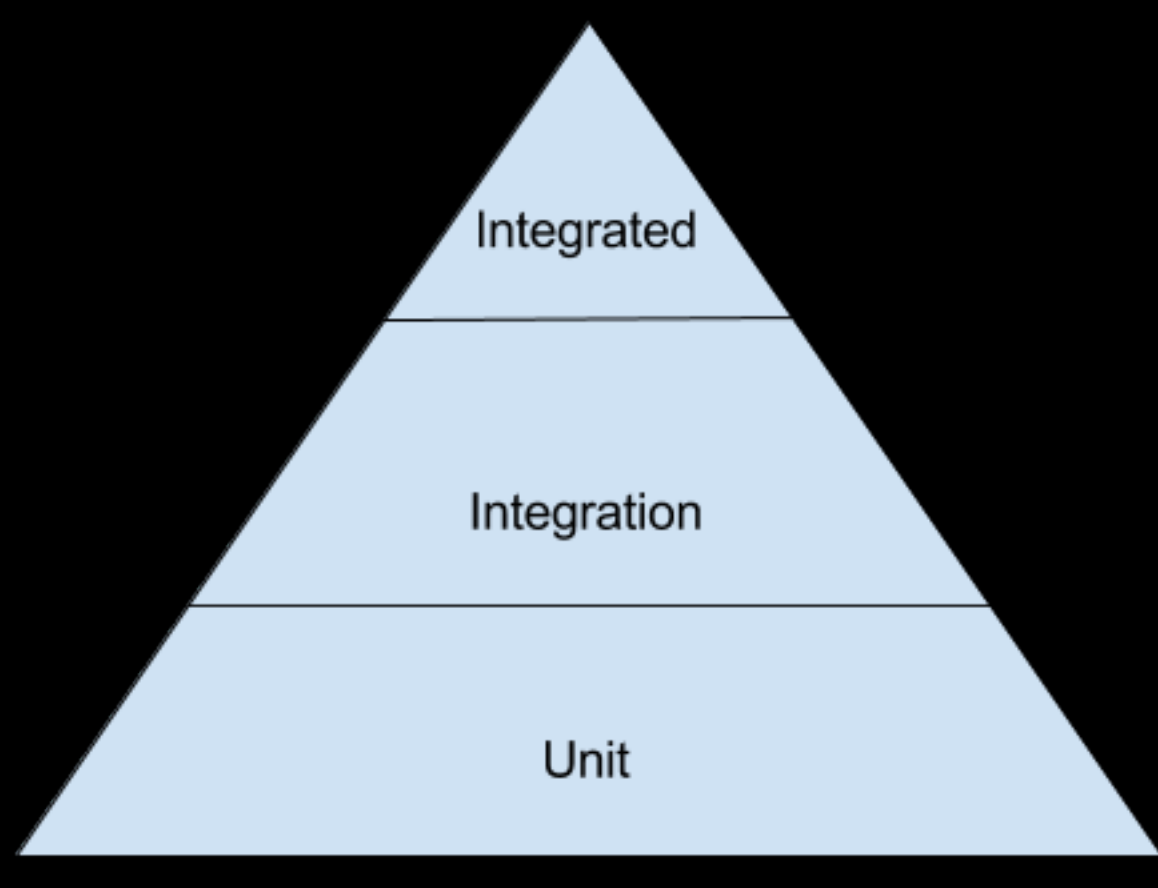
Schedule

- Current state of testing
- Testcontainers introduction
- 🧪 **Lab 1**
- 🧪 **Lab 2**
- 🧪 **Lab 3**

Your test is called what?!

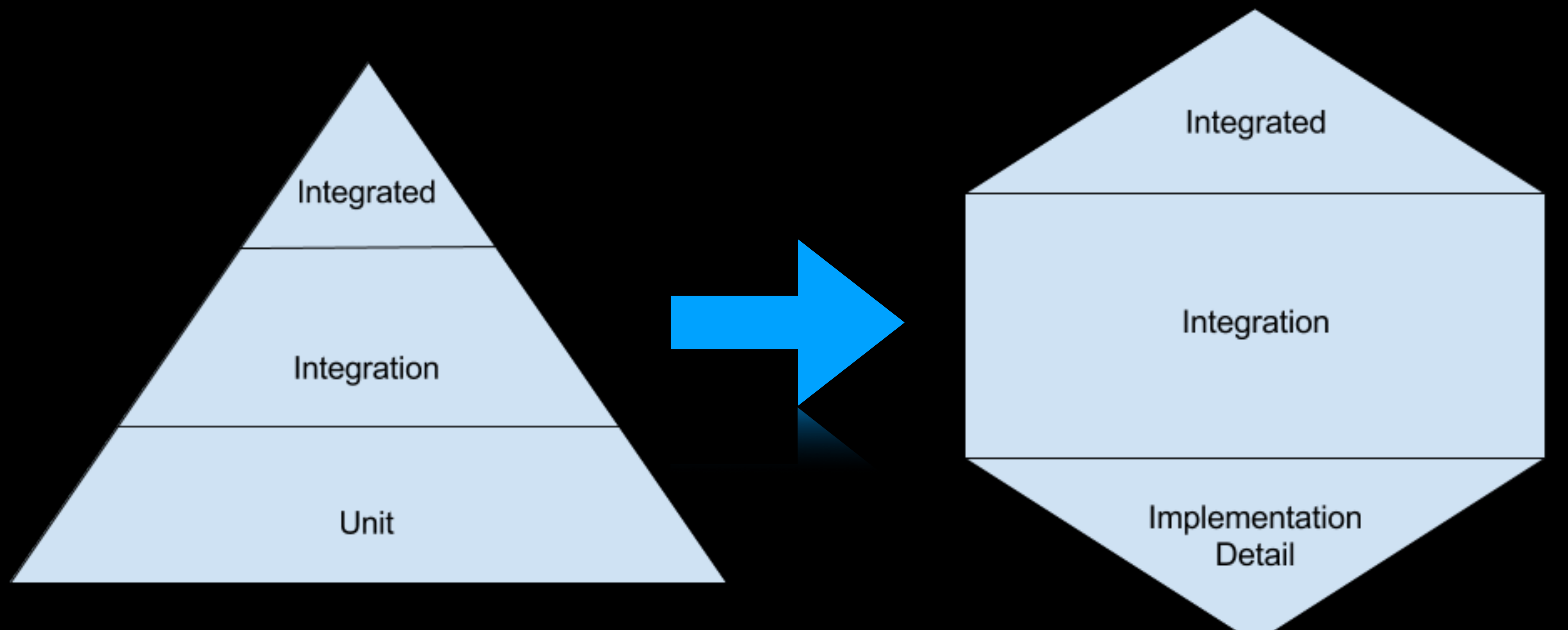


https://martinfowler.com/bliki/TestPyramid.html



https://martinfowler.com/bliki/TestPyramid.html

2: The pyramid is based on the assumption that broad-stack tests are expensive, slow, and brittle compared to more focused tests, such as unit tests. While this is usually true, there are exceptions. If my high level tests are fast, reliable, and cheap to modify - then lower-level tests aren't needed.

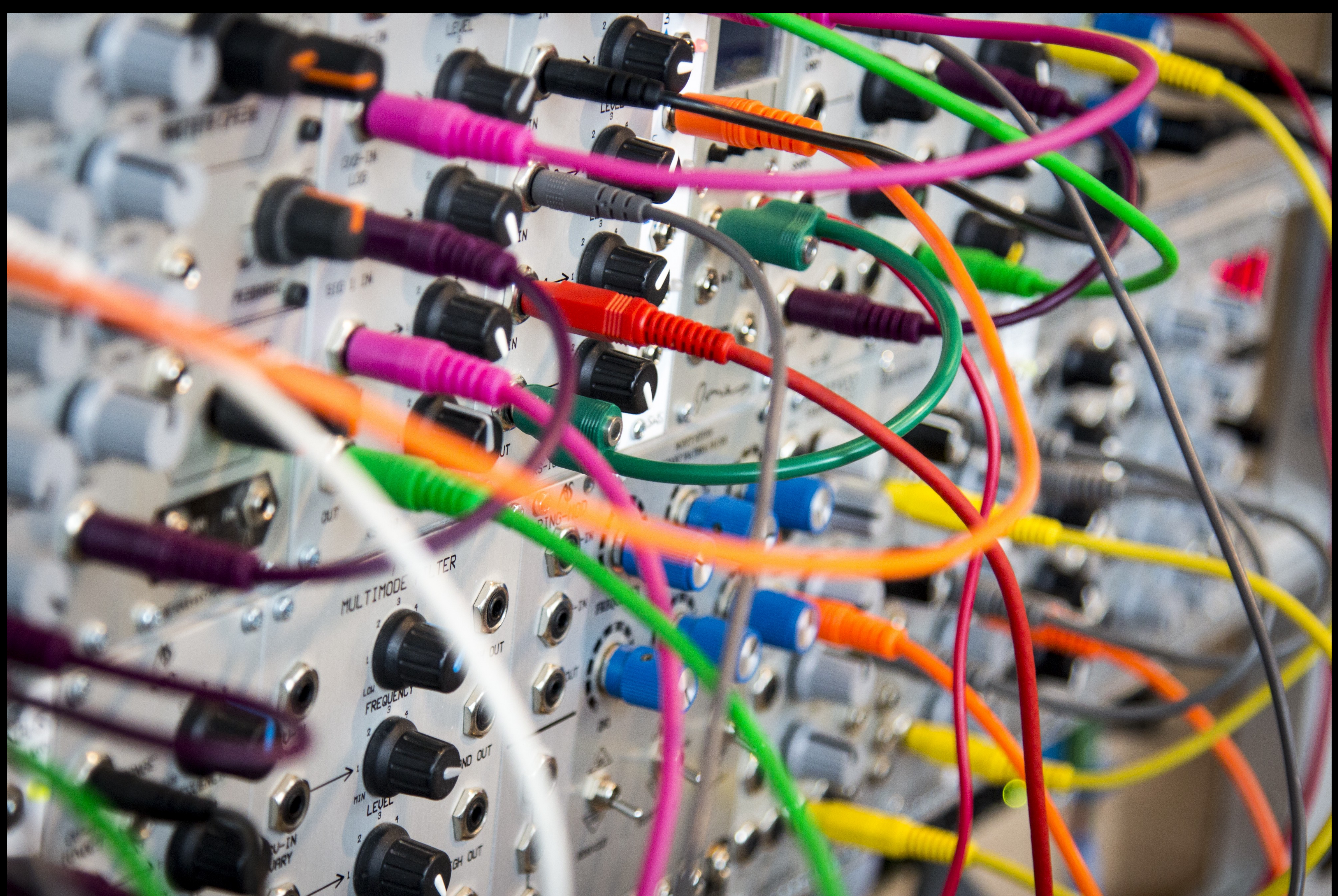

Integrated
Integration
Unit
Integrated
Integration
Implementation
Detail

“Write tests. Not too many. Mostly integration.”

https://kentcdodds.com/blog/write-tests

“You’re probably mocking too much.”

https://medium.com/extreme-programming/tests-dependencies-65f592a46529


MULTIMODE
LOW
FREQUENCY
MIN
LEVEL

```java
import static org.mockito.Mockito.*;

@RunWith(MockitoJUnitRunner.class)
public class PaymentWithRedirectionServiceImplTest {
    @Spy
    @InjectMocks
    private PaymentWithRedirectionServiceImpl service;

    @Mock
    private PaySafeHttpClient httpClient;

    @Before
    public void init() throws InvalidRequestException {
        PaySafeCaptureRequest captureRequest = new PaySafeCaptureRequest("dumbId", Utils.createContractConfigura
        doReturn(captureRequest).when(service).createRequest(any(RedirectionPaymentRequest.class));
        doReturn(captureRequest).when(service).createRequest(any(TransactionStatusRequest.class));
        doReturn(httpClient).when(service).getHttpClientInstance(any());
    }

    @Test
    public void finalizeRedirectionPayment() throws IOException, URISyntaxException {
        RedirectionPaymentRequest redirectionPaymentRequest = Mockito.mock(RedirectionPaymentRequest.class, Mock
        when(httpClient.retrievePaymentData(any(PaySafeCaptureRequest.class), anyBoolean())).thenReturn(Utils.cr
        when(httpClient.capture(any(PaySafeCaptureRequest.class), anyBoolean())).thenReturn(Utils.createSuccessP

        PaymentResponse response = service.finalizeRedirectionPayment(redirectionPaymentRequest);

        PaymentResponseSuccess responseSuccess = (PaymentResponseSuccess) response;
        Assert.assertEquals("0", responseSuccess.getStatusCode());
    }
```

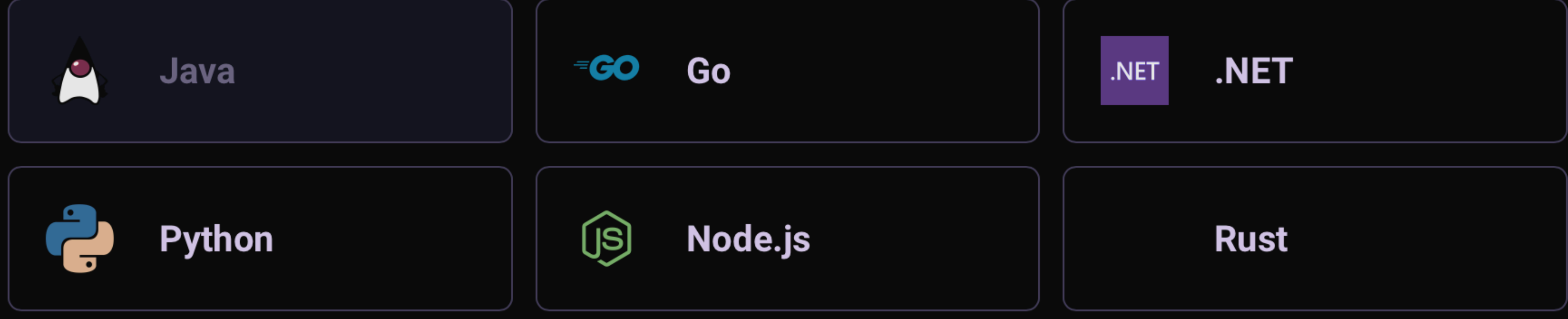
Testcontainers

Testcontainers for Java

- Home
- Quickstart
- Features
- Modules
  - Databases
  - Azure Module
  - Hashicorp Consul Module
  - Docker Compose Module
  - Elasticsearch container
  - GCloud Module
  - HiveMQ Module
  - K3s Module
  - Kafka Containers
  - LocalStack Module
  - Mockserver Module
  - Nginx Module
  - Apache Pulsar Module
  - RabbitMQ Module
  - Redpanda
  - Solr Container

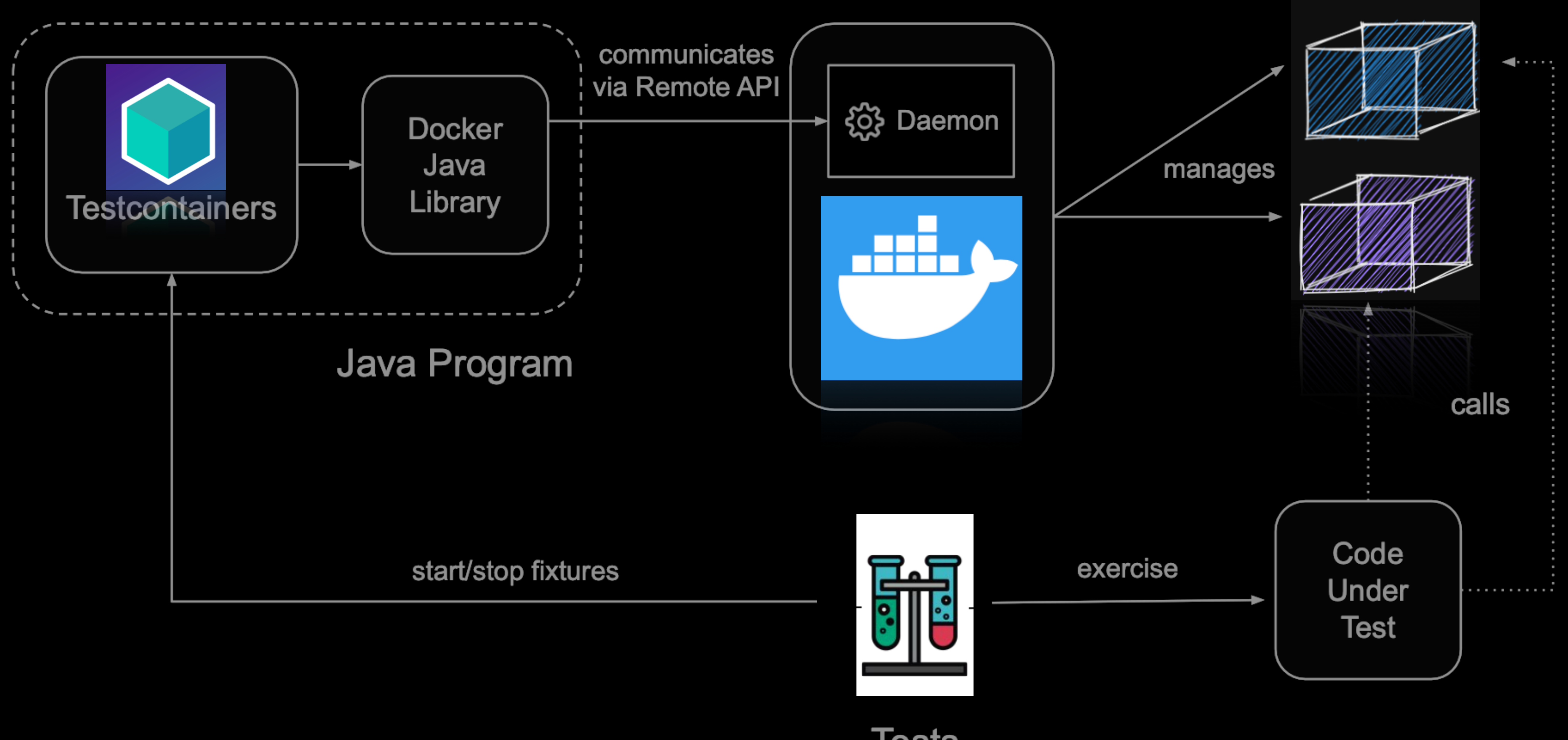
Testcontainers

Not using Java? Here are other supported languages!



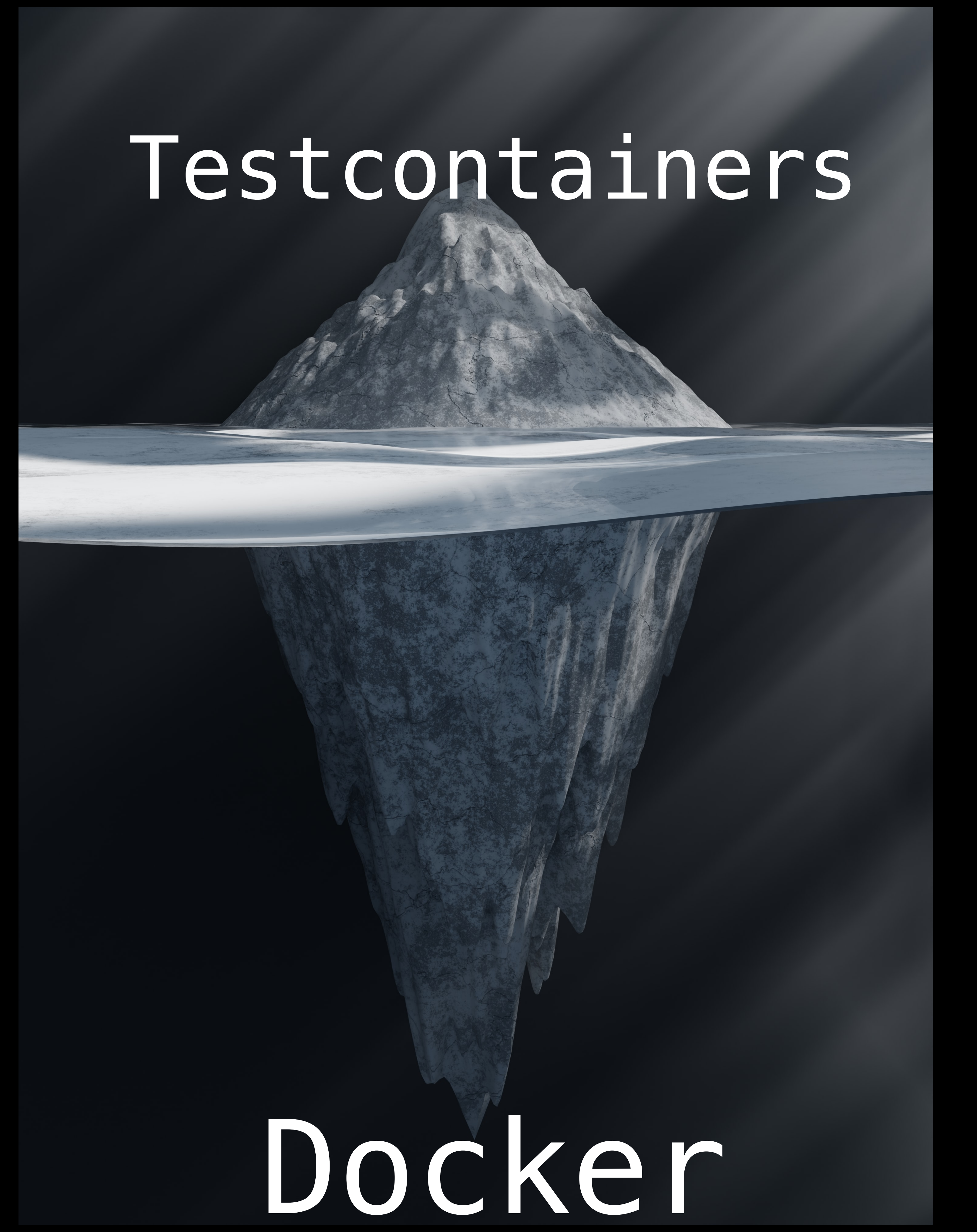
Databases

- Database containers
- JDBC support
- R2DBC support
- Cassandra Module
- CockroachDB Module
- Couchbase Module
- Clickhouse Module
- DB2 Module
- Dynalite Module
- InfluxDB Module
- MariaDB Module
- MongoDB Module
- MS SQL Server Module
- MySQL Module
- Neo4j Module
- Oracle-XE Module
- OrientDB Module
- Postgres Module
- Presto Module
- QuestDB Module


Testcontainers
Docker
Java
Library
communicates
via Remote API
Daemon
manages
Java Program
calls
start/stop fixtures
exercise
Code
Under
Test
Tests

```java
@Test
public void testSimple() throws SQLException {
    try (
        PostgreSQLContainer<?> postgres = new PostgreSQLContainer<>(PostgreSQLTestImages.POSTGRES_TEST_IMAGE)
            .withDatabaseName(DB_NAME)
            .withUsername(USER)
            .withPassword(PWD)
    ) {
        postgres.start();

        ResultSet resultSet = performQuery(postgres, "SELECT 1");

        int resultSetInt = resultSet.getInt(1);
        assertThat(resultSetInt).as("A basic SELECT query succeeds").isEqualTo(1);
    }
}
```

```java
public class VakantieRepositoryTest {

    private static PostgreSQLContainer<?> postgres =
            new PostgreSQLContainer<>( dockerImageName: "postgres:15");

    @DynamicPropertySource
    static void configureContext(DynamicPropertyRegistry registry) {
        registry.add( name: "database.portnr", () -> postgres.getMappedPort( originalPort: 5432));
    }
```

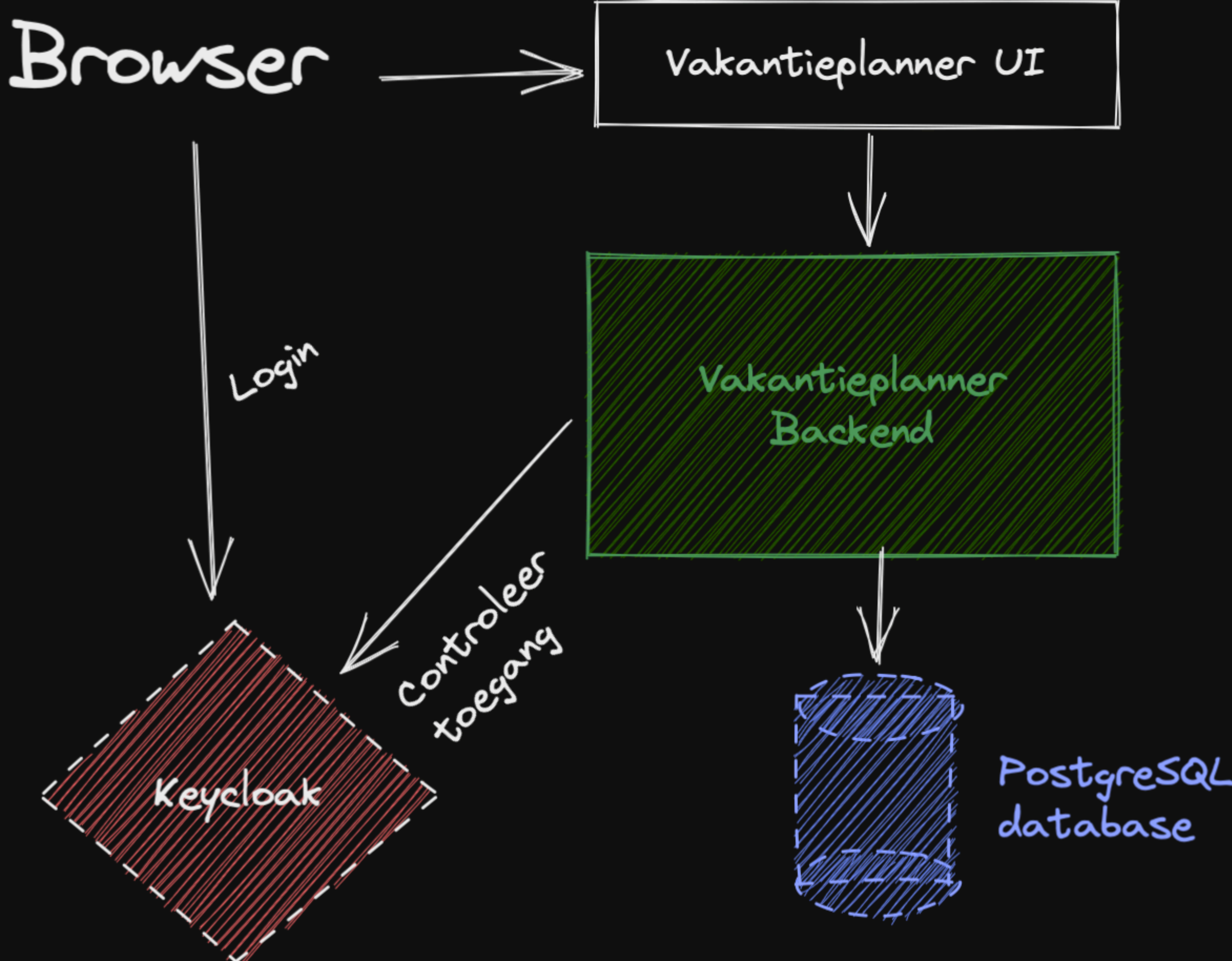

Testcontainers
Docker

“Docker is a great abstraction”

Our Demo App

- Java 17 & Spring boot
- Uses a database (PostgreSQL)
- Uses Keycloak for authentication

- There is frontend available
  - Available as a Docker Image for demo purposes


Browser
Vakantieplanner UI
Login
Vakantieplanner Backend
Controleer toegang
Keycloak
PostgreSQL database

Lab 1.

- Goal: run the app manually (using Docker)
- App consists of 4 parts:
  - Database
  - Keycloak
  - Backend
  - User Interface

```
$ docker ps
```

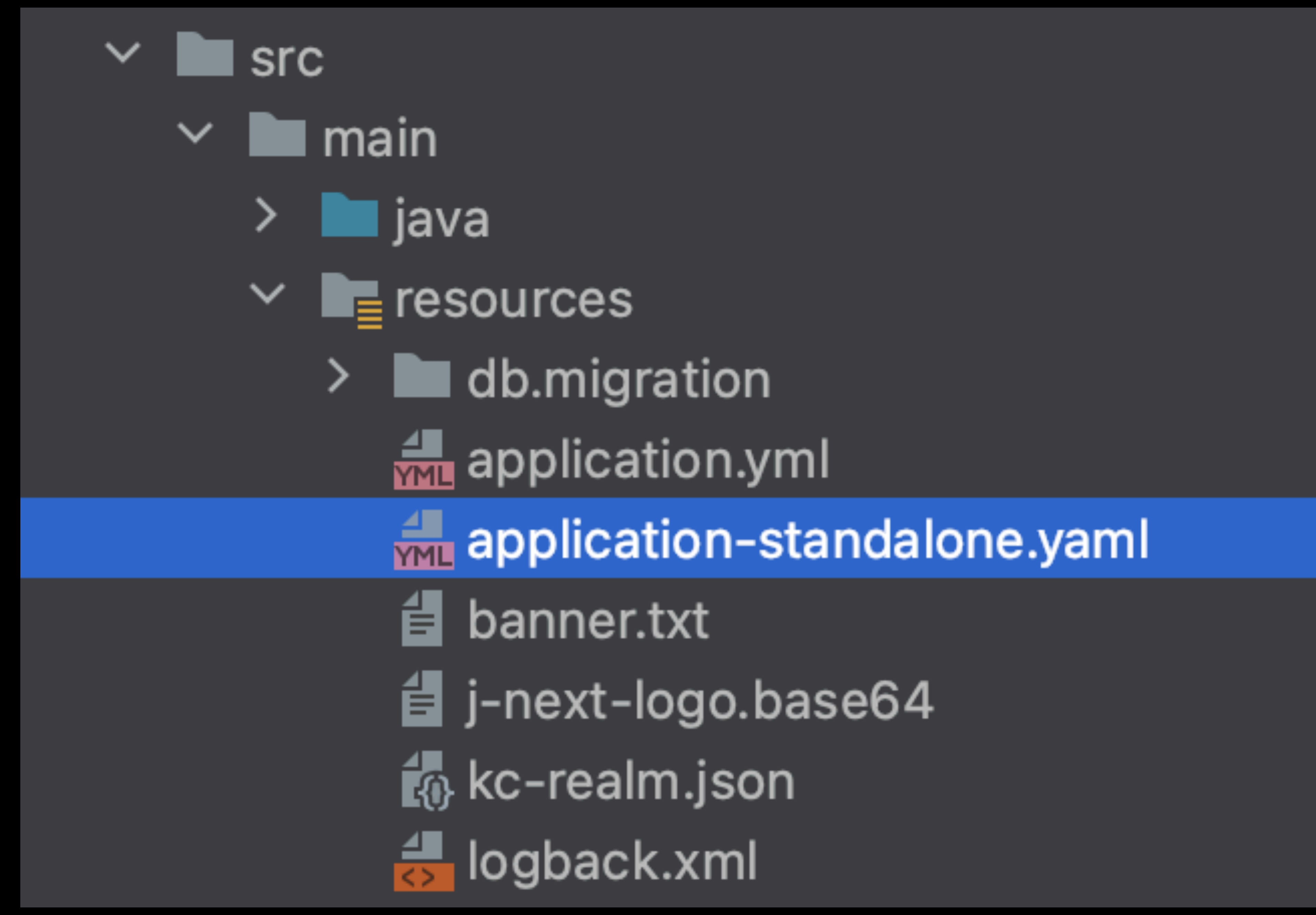
Lab 1.

```
$ git clone https://github.com/bastoker/testcontainers-workshop.git
```

🧪 Lab 1.

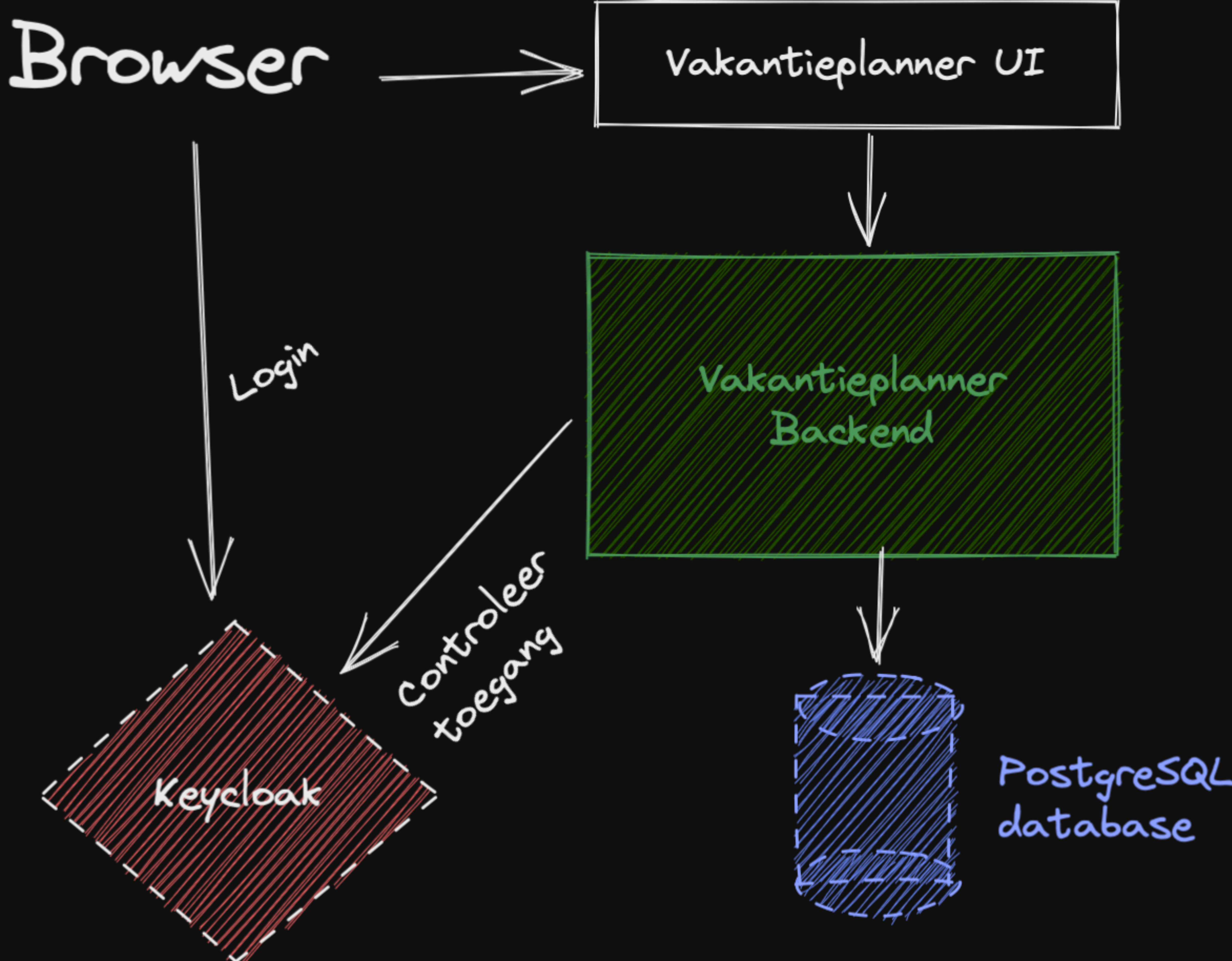
- testcontainers-workshop **[vakantieplanner]** ~/Workspace/testco
  - .idea
  - .misc
  - .mvn
  - src
    - main
      - java
        - helper
        - nl.jnext.workshop.testcontainers.vakantieplanner
        - standalone
          - StartApplicationStandalone
          - StartDatabaseStandalone
          - StartFrontendStandalone
          - StartKeycloakStandalone

Lab 1.

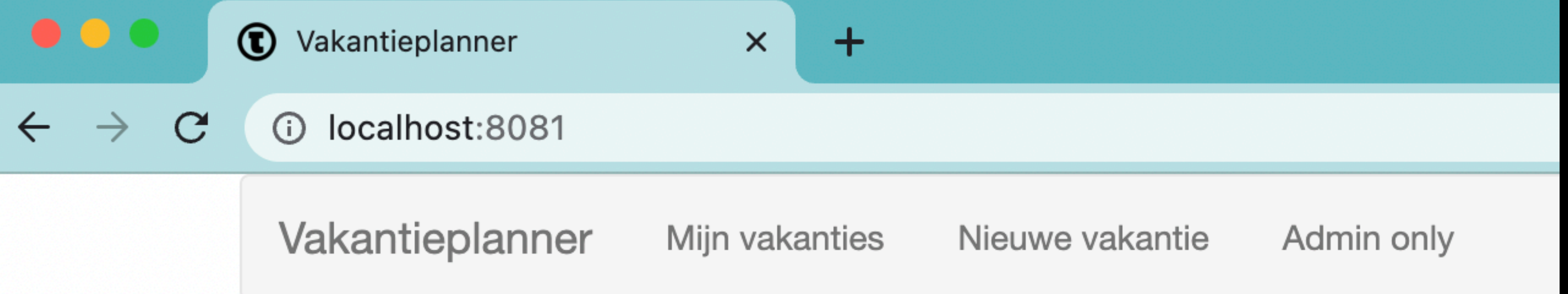


🧪 Lab 1.

```yaml
  datasource:
    url: 'jdbc:postgresql://localhost:53580/standalone-db?loggerLevel=OFF'
    username: 'admin'
    password: 'admin'
keycloak:
  auth-server-url: 'http://localhost:53538/'
  realm: vakantieplanner
  resource: spring-app
  use-resource-role-mappings: false
  bearer-only: true
```



Browser
Vakantieplanner UI
Login
Vakantieplanner
Backend
Controleer
toegang
Keycloak
PostgreSQL
database

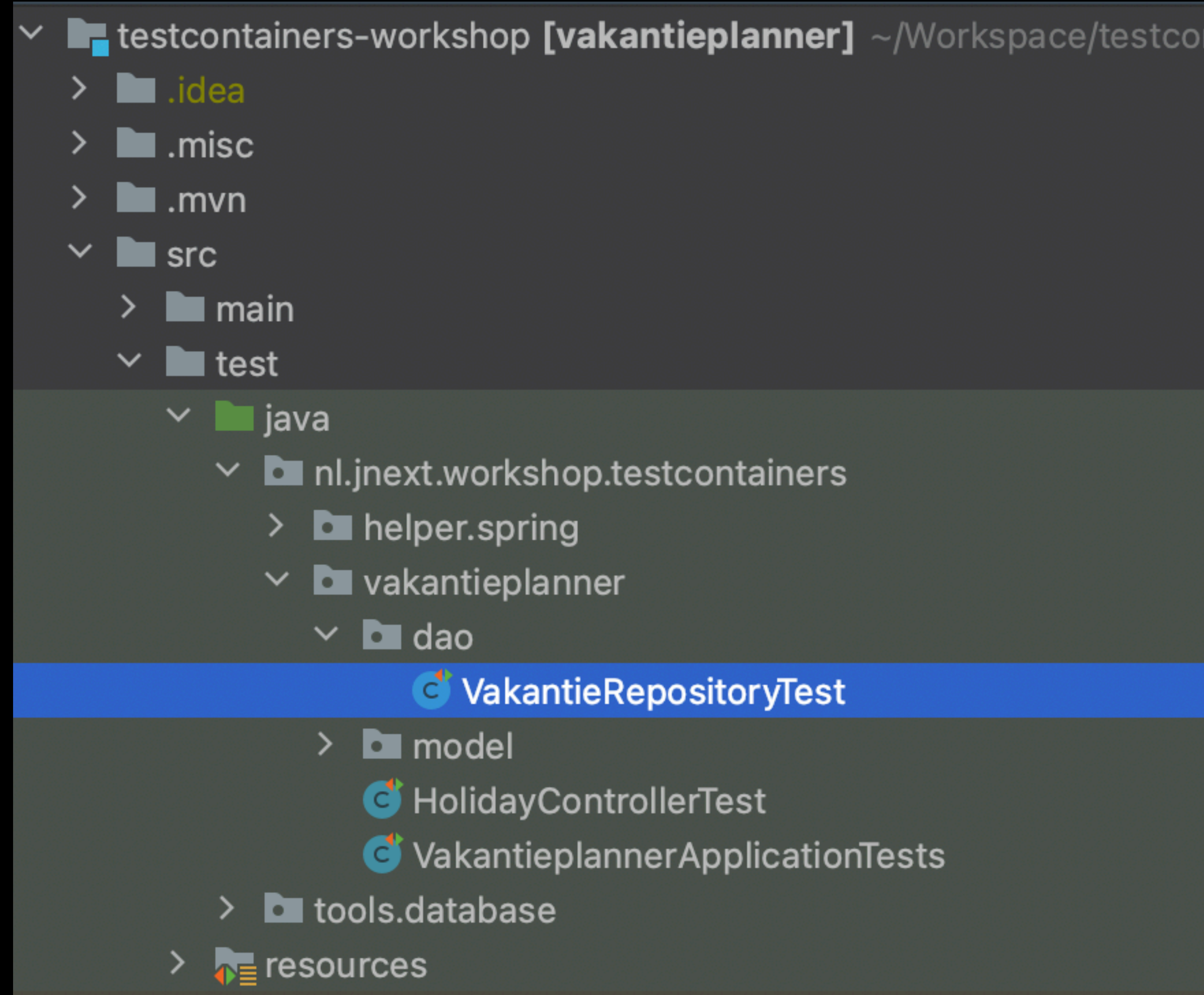

Vakantieplanner
localhost:8081
Vakantieplanner
Mijn vakanties
Nieuwe vakantie
Admin only
Overzicht
Omschrijving
id
Van
Kerstvakantie2
1
22 November 2022

Food :-)

Lab 2.

- Goal: Test database access layer
  - Using JUnit test lifecycle
  - Using PostgreSQL Testcontainer
  - Using Spring Injection of properties dynamically

Lab 2.



Lab 2.

```java
@DynamicPropertySource
static void configureContext(DynamicPropertyRegistry registry) {
    // Doe hier iets:

    // En maak de Spring configuratie af:
    registry.add( name: "spring.datasource.url", () -> ???);
    registry.add( name: "spring.datasource.username", () -> ???);
    registry.add( name: "spring.datasource.password", () -> ???);
}
```

Hint:

datasource-url wordt ook wel JDBC-url genoemd soms

Lab 2.

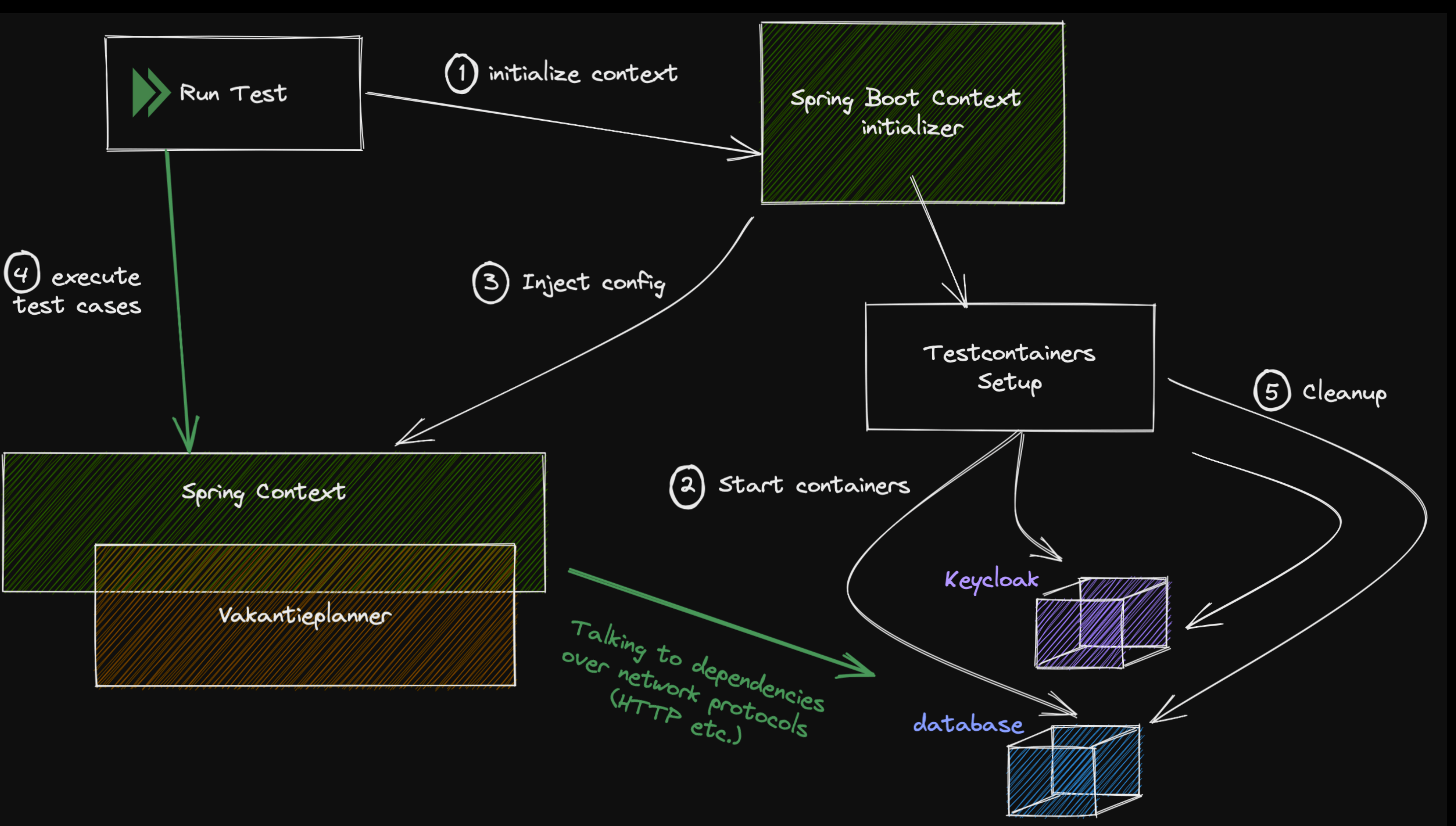
- Solution
- Extra nice things:
  - convenience annotations
- Questions

Lab 3.

- Goal: Test entire app using Testcontainers
- With enabled authentication
- Happy Duck, eh, Bug hunt!

🧪 Lab 3.

- src
  - main
  - test
    - java
      - nl.jnext.workshop.testcontainers
        - helper.spring
        - vakantieplanner
          - dao
          - model
          - VakantieplannerApiTests


Run Test
1 initialize context
Spring Boot Context initializer
4 execute test cases
3 Inject config
Testcontainers Setup
5 Cleanup
Spring Context
2 Start containers
Vakantieplanner
Keycloak
Talking to dependencies over network protocols (HTTP etc.)
database

Bedankt voor jullie aanwezigheid!