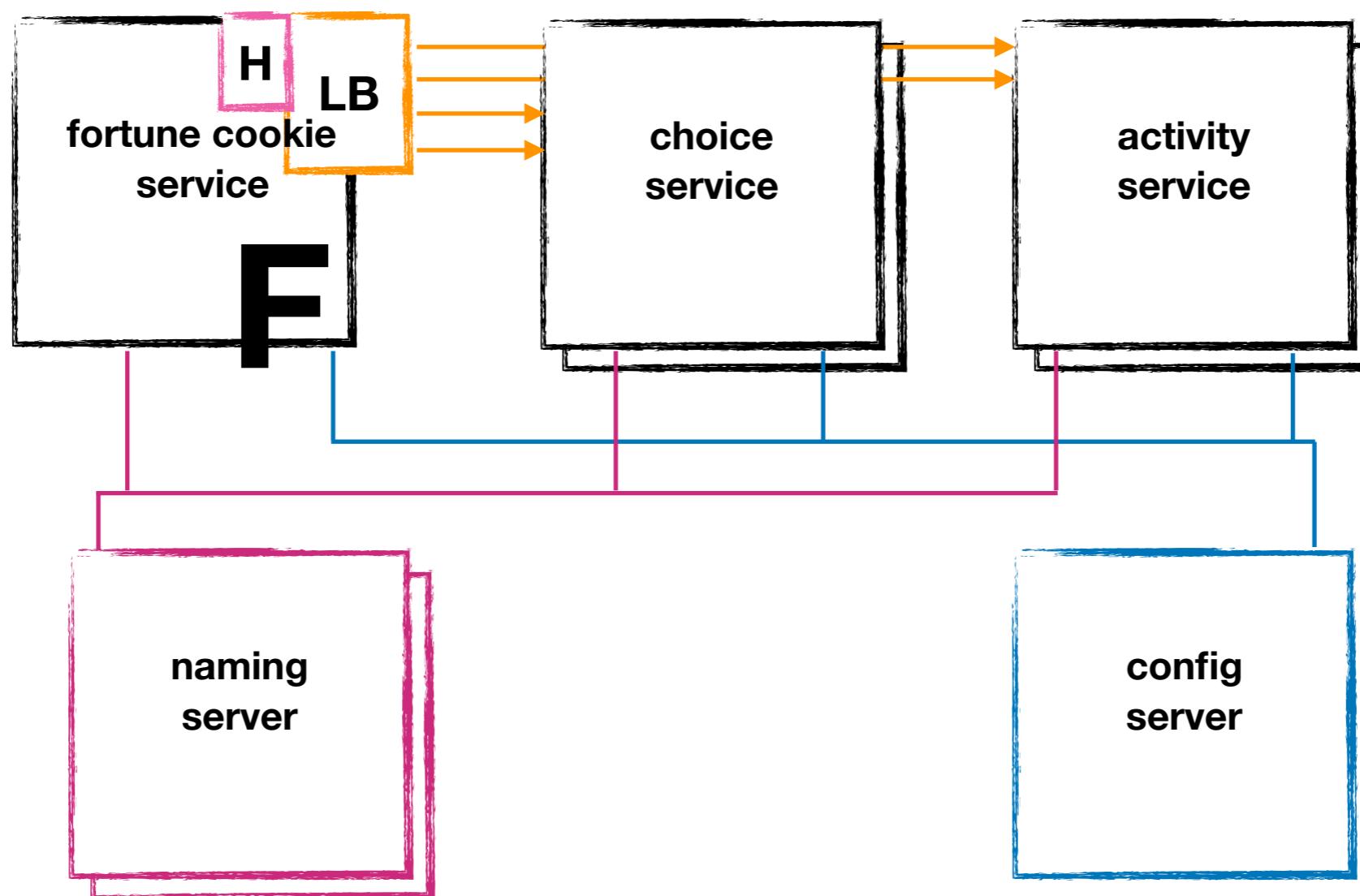


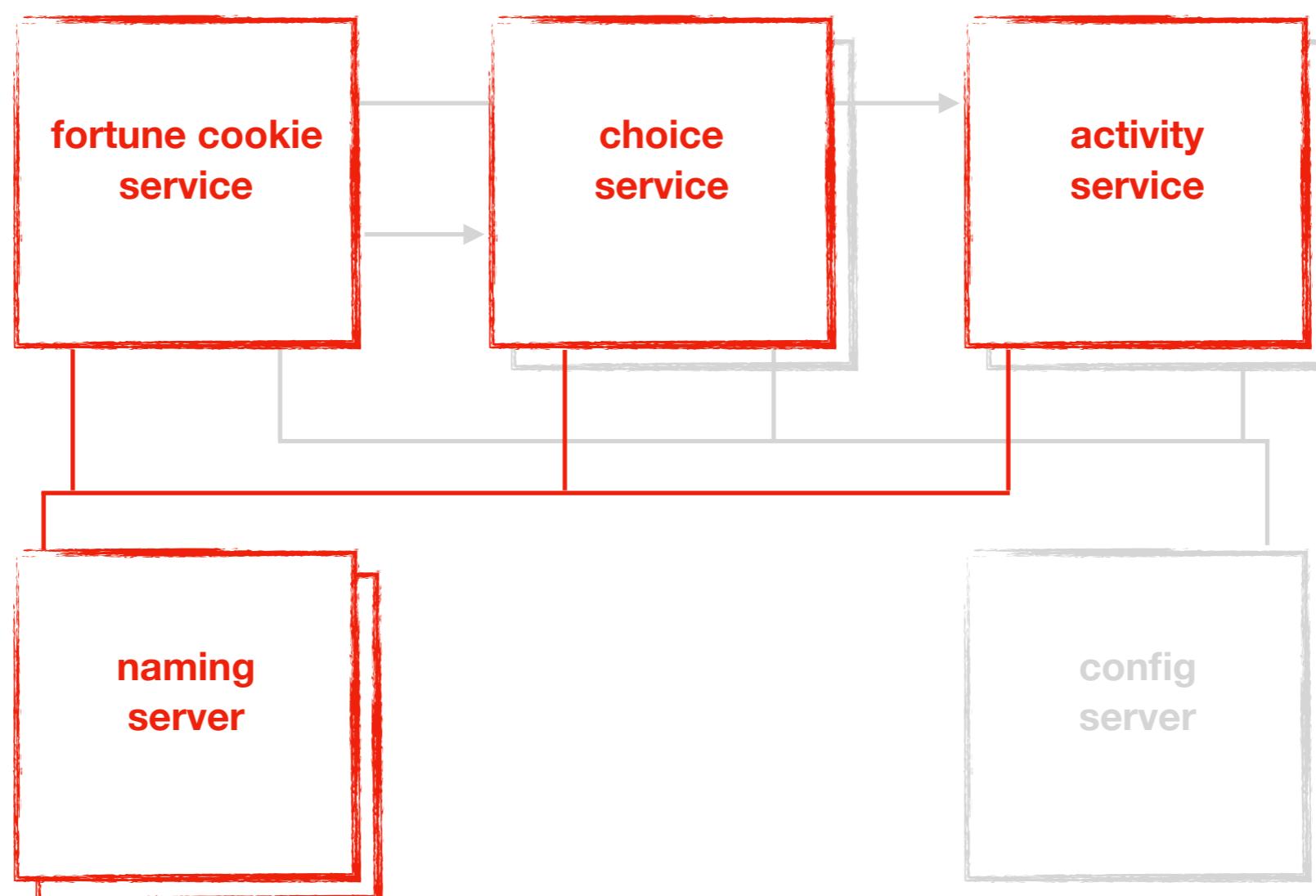


**Spring  
Cloud**

# Fortune



# I Serwisy i Eureka



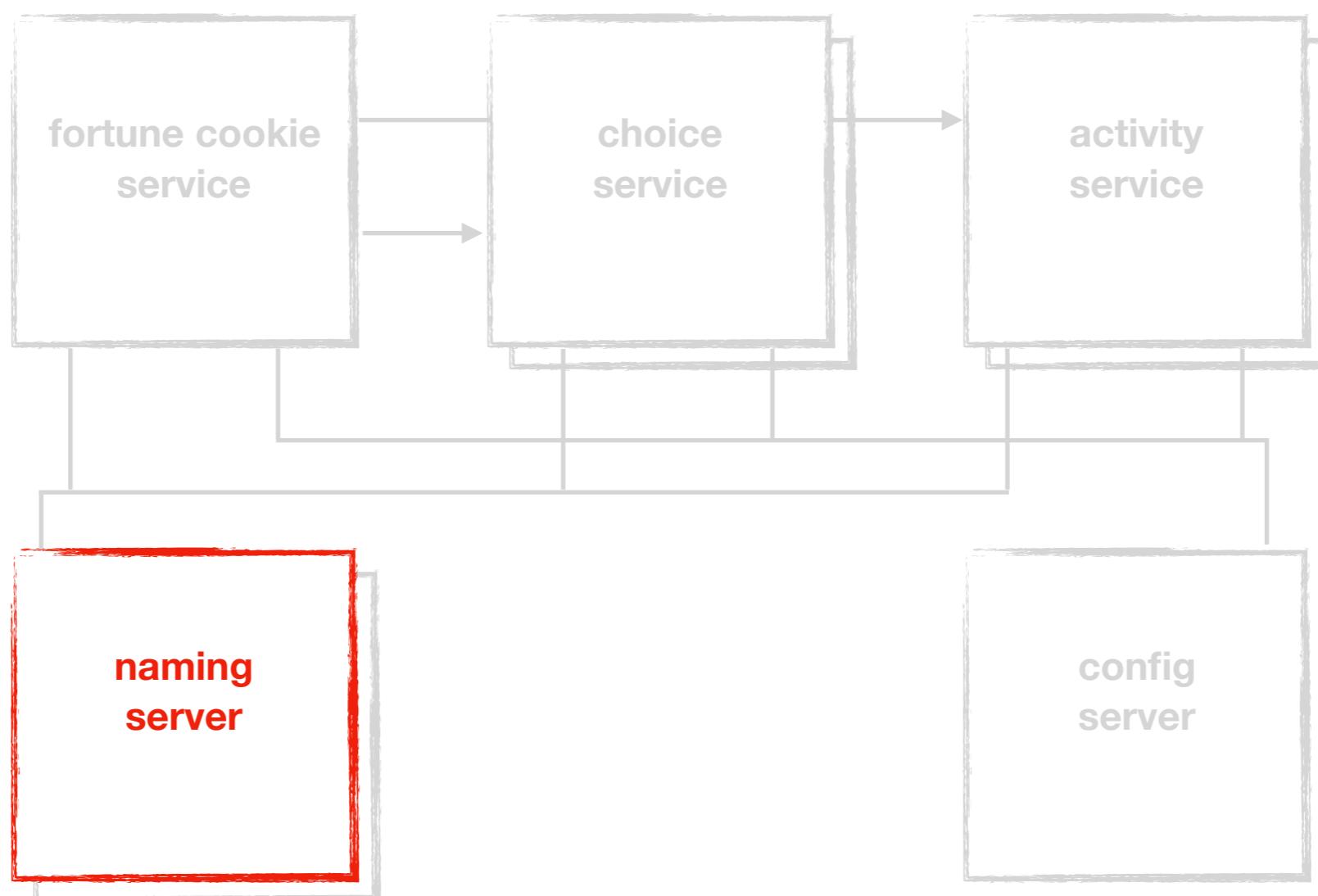
# I Serwisy i Eureka

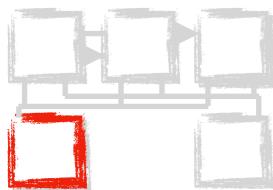
- serwer Eureka
- serwis fortune-cookie-service
  - jako klient Eureki
- zadanie - utworzenie serwisów, klientów Eureki
  - decision-service
  - activity-service
- refaktor fortune-cookie-service
  - korzysta z decision-service i activity-service
- refaktor Eureka - 2 instancje

# I Serwisy i Eureka

- **serwer Eureka**
- serwis fortune-cookie-service
  - jako klient Eureki
- zadanie - utworzenie serwisów, klientów Eureki
  - decision-service
  - activity-service
- refaktor - fortune-cookie-service
  - korzysta z decision-service i activity-service
- refaktor Eureka - 2 instancje

# Serwer Eureka





# szkielet projektu

SPRING INITIALIZR bootstrap your application now

Generate a  with  and Spring Boot

**Project Metadata**

Artifact coordinates

Group

Artifact

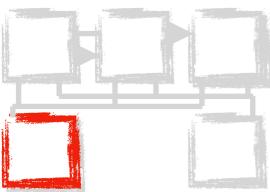
**Dependencies**

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

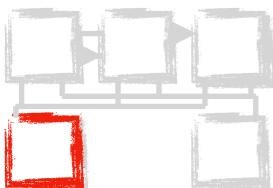
Don't know what to look for? Want more options? [Switch to the full version.](#)



# pliki konfiguracyjne

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "eureka-server". It contains a ".idea" folder, a ".mvn" folder, and a "src" directory. The "src" directory has "main" and "resources" subfolders. The "resources" folder contains "application.yml" and "bootstrap.yml".
- application.yml:** This file is currently open in the editor. It defines a "server" section with a "port" set to 8761.
- bootstrap.yml:** This file is also visible in the editor. It defines a "spring" section with an "application" section containing a "name" set to "eureka-server".

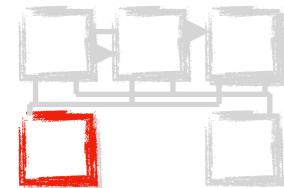


# dodanie @

The screenshot shows a Java application structure in an IDE. The project is named "eureka-server" located at "/Fortune/\_1\_serwer\_eu". The code editor displays the main class, `EurekaServerApplication.java`, which contains the following code:

```
1 package pl.altkom.eurekaserver;
2
3 import ...
4
5
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class EurekaServerApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(EurekaServerApplication.class, args);
13     }
14 }
15
```

The code editor highlights the annotations `@SpringBootApplication` and `@EnableEurekaServer`. The file is part of the `pl.altkom.eurekaserver` package. The `resources` folder contains `application.yml` and `bootstrap.yml`. Other files in the project include `.gitignore`, `eureka-server.iml`, `mvnw`, `mvnw.cmd`, and `pom.xml`.

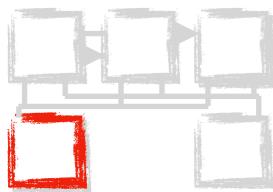


# niechęć do samotności ;)

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "eureka-server". The "resources" folder contains "application.yml" and "bootstrap.yml".
- Code Editor:** The "application.yml" file is open, showing configuration for a Eureka server with port 8761 and a client.
- Logs:** The "Console" tab displays application logs. It includes error messages from the "DiscoveryClient" and "InstanceInfoReplicator" classes, indicating a problem with the instance info replicator. It also includes informational log entries from "EurekaServerBootstrap" and "EurekaServerApplication" classes.





# UI Eureka

Screenshot of the Spring Eureka UI interface:

The browser address bar shows `localhost:8761`.

The header includes the Spring Eureka logo and navigation links for **HOME** and **LAST 1000 SINCE STARTUP**.

### System Status

Environment	test	Current time	2018-06-02T16:55:39 +0200
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	0
		Renews (last min)	0

### DS Replicas

localhost

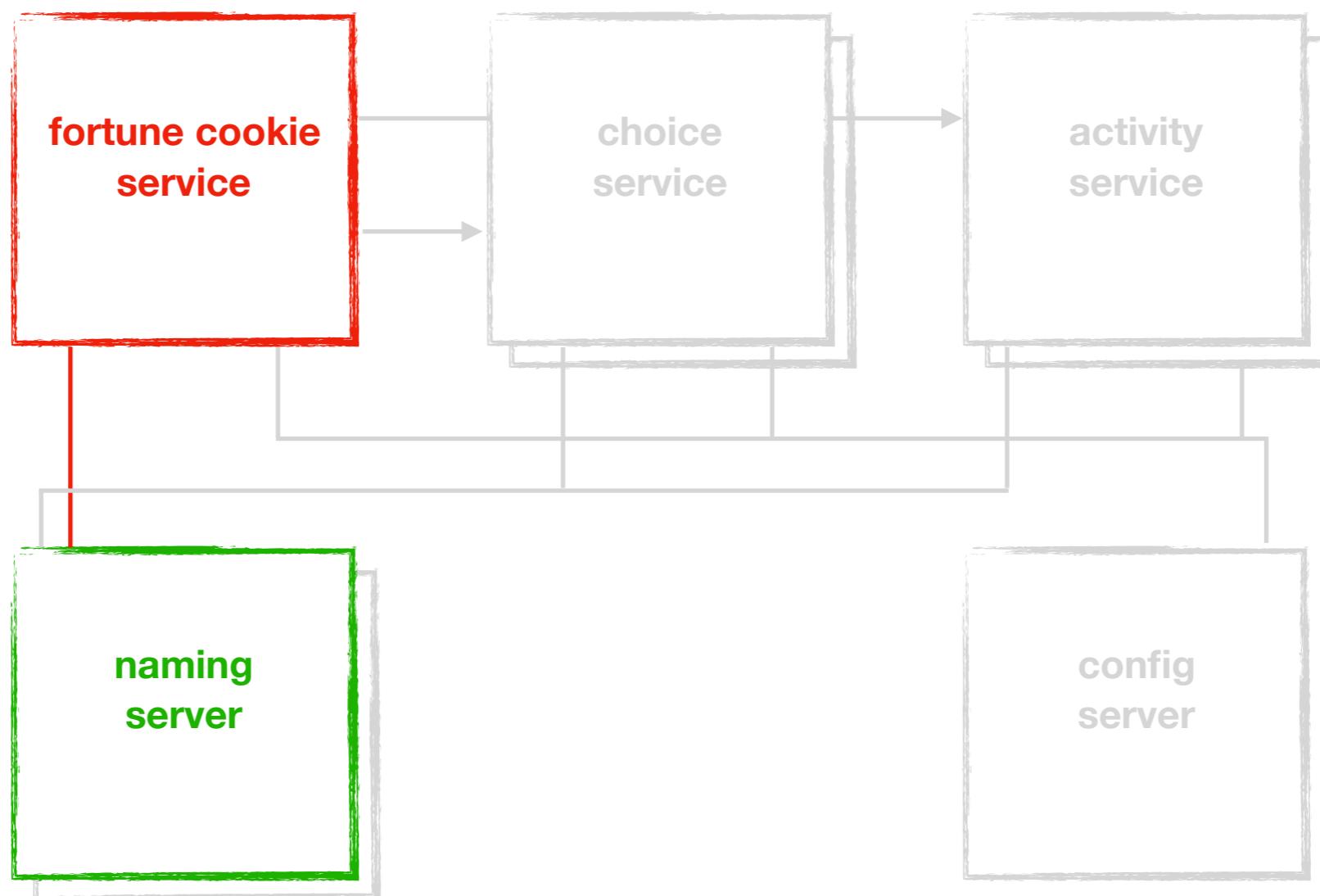
### Instances currently registered with Eureka

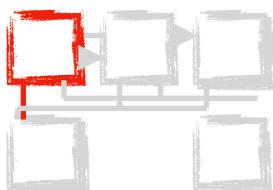
Application	AMIs	Availability Zones	Status
No instances available			

# I Serwisy i Eureka

- serwer Eureka
- **serwis fortune-cookie-service**
  - **jako klient Eureki**
- zadanie - utworzenie serwisów, klientów Eureki
  - decision-service
  - activity-service
- refaktor - fortune-cookie-service
  - korzysta z decision-service i activity-service
- refaktor Eureka - 2 instancje

# fortune-cookie-service





# szkielet projektu

SPRING INITIALIZR bootstrap your application now

Generate a **Maven Project** with **Java** and **Spring Boot 2.0.2**

**Project Metadata**

Artifact coordinates

**Group**: pl.altkom

**Artifact**: fortune-cookie-service

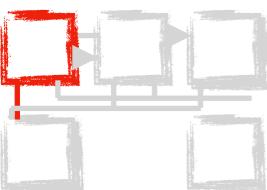
**Dependencies**

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Web, Eureka Discovery, Actuator

Generate Project \*



# pliki konfiguracyjne

The screenshot shows an IDE interface with two configuration files open:

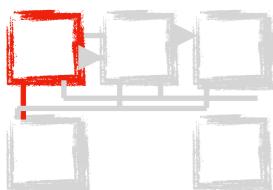
- application.yml** (top tab):

```
spring:
  application:
    name: fortune-cookie-service
```
- bootstrap.yml** (bottom tab):

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
  server:
    port: 8080
  fortunes: It's a good day for a cookie!, Take a nap!
```

The project structure on the left includes:

- Project: fortune-cookie-service (~/Fortune/\_fortune\_cookie\_service/fortune-cookie-service)
- src:
  - main:
    - java:
      - pl.altkom.fortunecookieService:
        - controller:
          - FortuneController
        - FortuneCookieServiceApplication
    - resources:
      - application.yml
      - bootstrap.yml
  - test
  - target
  - .gitignore
  - fortune-cookie-service.iml
  - mvnw
  - mvnw.cmd
  - pom.xml
  - External Libraries
  - watcher and Console

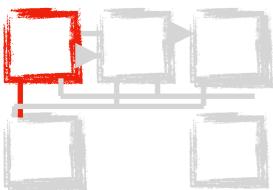


# zależność i @

The screenshot shows an IDE interface with two main panes. The left pane is the Project Structure view, showing the project tree for 'fortune-cookie-service'. It includes folders for .idea, .mvn, src (with main/java and resources), test, target, .gitignore, pom.xml, External Libraries, and Scratches and Consoles. The pom.xml file is currently selected. The right pane contains two code editors. The top editor shows the 'FortuneCookieServiceApplication.java' file, which starts with annotations: '@SpringBootApplication' and '@EnableDiscoveryClient'. Below these are the class definition and a main method. The bottom editor shows the 'pom.xml' file, specifically the dependency section. It lists several dependencies, including ones for Spring Cloud (Eureka client) and Spring Boot (test dependency).

```
pom.xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

```
FortuneCookieServiceApplication.java
@SpringBootApplication
@EnableDiscoveryClient
public class FortuneCookieServiceApplication {
    public static void main(String[] args) { SpringApplication.run(FortuneCookieServiceApplication.class, args); }
}
```



# kontroler

```
@RestController
public class FortuneController {

    @Value("${fortunes}")
    private String fortunes;

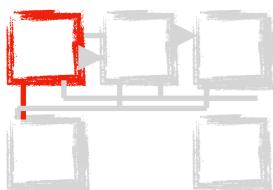
    @GetMapping("/fortune")
    @ResponseBody // automatic conversion to JSON
    public Fortune fortune() { return getFortune(); }

    private Fortune getFortune() {
        String[] sentences = fortunes.split( regex: "," );
        int i = (int) Math.round( Math.random() * (sentences.length - 1));
        return new Fortune(sentences[i]);
    }

    class Fortune {
        private String fortune;

        public Fortune(String fortune) { this.fortune = fortune; }

        public String getFortune() { return fortune; }
    }
}
```



# test serwisu

Normal Basic Auth Digest Auth OAuth 1.0 No environment ▾

http://localhost:8080/fortune GET ▾

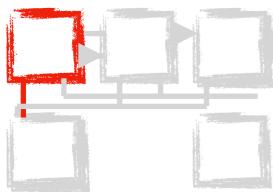
Header Value Manage presets

Send Preview Add to collection

Body Headers (3) STATUS 200 TIME 154 ms

Pretty Raw Preview JSON XML

```
1 {  
2   "fortune": "It's a good day for a cookie!"  
3 }
```



# serwis w Eurece

localhost:8080/fortune

Eureka

spring Eureka

HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2018-06-02T18:57:10 +0200
Data center	default	Uptime	00:05
		Lease expiration enabled	false
		Renews threshold	0
		Renews (last min)	5

### DS Replicas

localhost

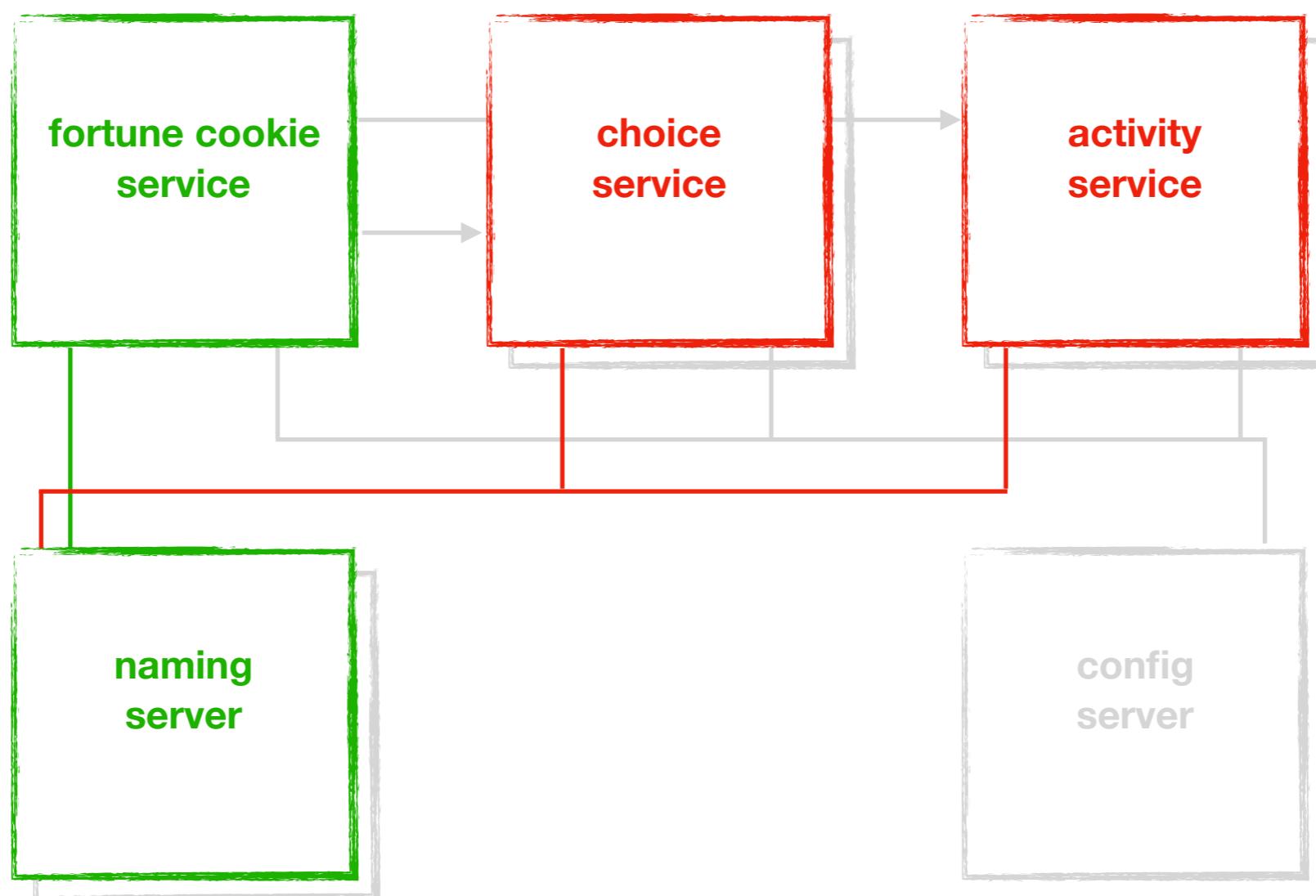
### Instances currently registered with Eureka

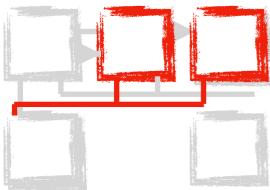
Application	AMIs	Availability Zones	Status
FORTUNE-COOKIE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:fortune-cookie-service:8080
UNKNOWN	n/a (1)	(1)	UP (1) - 192.168.8.102:8761

# I Serwisy i Eureka

- serwer Eureka
- **serwis fortune-cookie-service**
  - jako klient Eureki
- **zadanie - utworzenie serwisów, klientów Eureki**
  - **decision-service**
  - **activity-service**
- refaktor - fortune-cookie-service
  - korzysta z decision-service i activity-service
- refaktor Eureka - 2 instancje

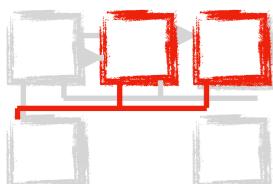
# zadanie - implementacja





# zadanie - implementacja

- analogicznie do **fortune-cookie-service**
- decision-service
  - 8000
  - **GetMapping("/decision")**
  - It's a good idea, It's a bad day etc.
- activity-service
  - 8010
  - **GetMapping("/activity")**
  - to run a marathon, for cookies etc.



# test decision-service

Normal Basic Auth Digest Auth OAuth 1.0 No environment ▾

http://localhost:8000/decision GET ▾

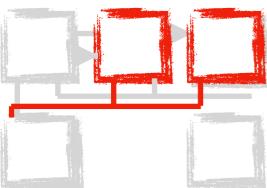
Header Value Manage presets

Send Preview Add to collection

Body Headers (3) STATUS 200 TIME 9 ms

Pretty Raw Preview JSON XML

```
1 {  
2   "decision": "It's time"  
3 }
```



# test activity-service

Normal Basic Auth Digest Auth OAuth 1.0 No environment ▾

http://localhost:8010/activity GET ▾

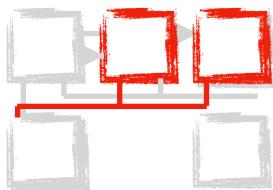
Header Value Manage presets

Send Preview Add to collection

Body Headers (3) STATUS 200 TIME 136 ms

Pretty Raw Preview JSON XML

```
1 {  
2   "activity": "run a marathon"  
3 }
```



# serwisy w Eurece

 **spring Eureka**

HOME LAST 1000 SINCE STARTUP

## System Status

Environment	test	Current time	2018-06-02T19:45:12 +0200
Data center	default	Uptime	00:53
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	16

## DS Replicas

localhost

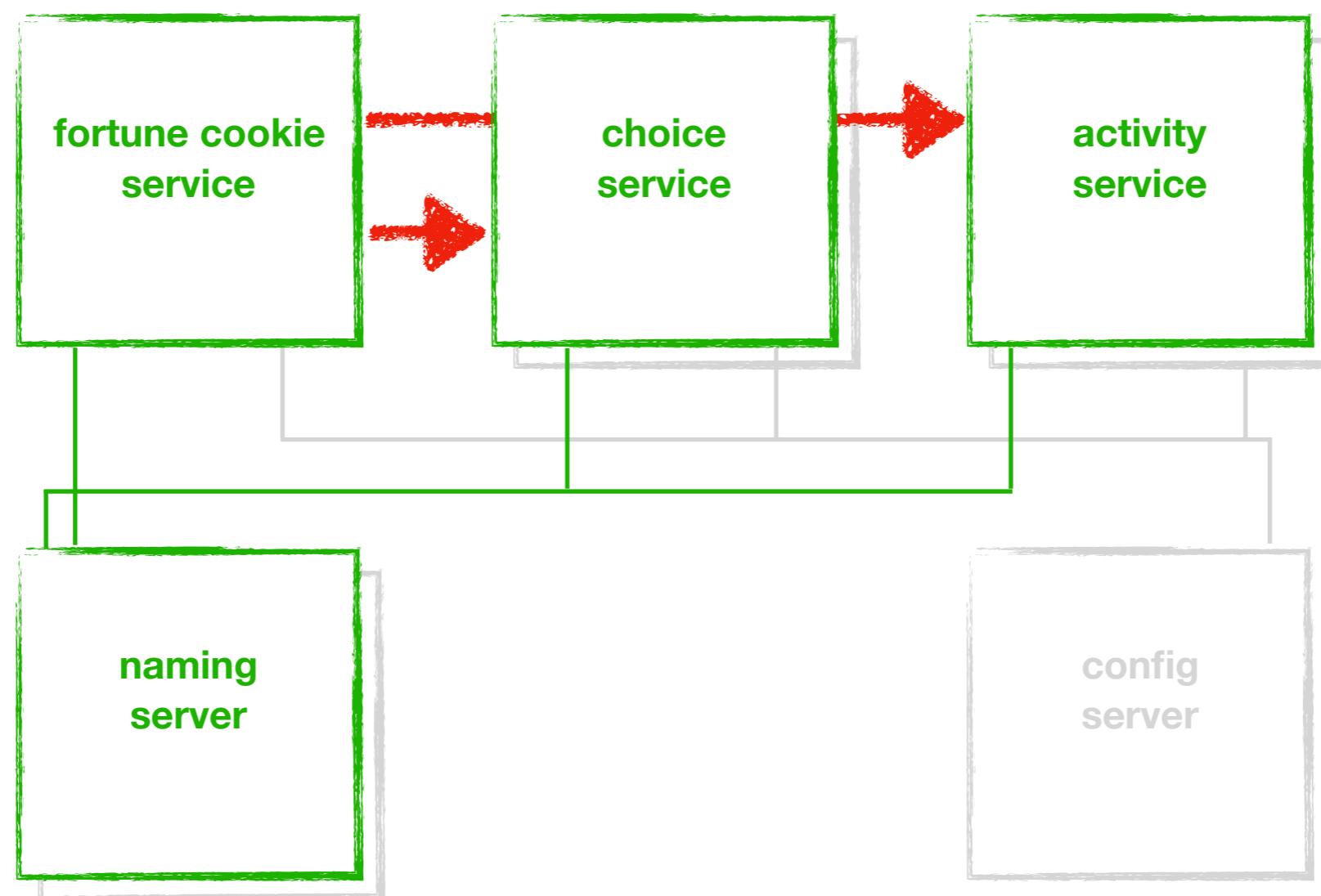
## Instances currently registered with Eureka

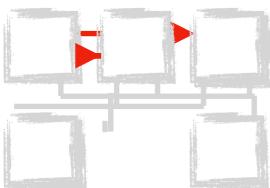
Application	AMIs	Availability Zones	Status
ACTIVITY-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:activity-service:8010
DECISION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:decision-service:8000
FORTUNE-COOKIE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:fortune-cookie-service:8080
UNKNOWN	n/a (1)	(1)	UP (1) - 192.168.8.102:8761

# I Serwisy i Eureka

- serwer Eureka
- **serwis fortune-cookie-service**
  - jako klient Eureki
- zadanie - utworzenie serwisów, klientów Eureki
  - decision-service
  - activity-service
- **refaktor - fortune-cookie-service**
  - **korzysta z decision-service i activity-service**
- refaktor Eureka - 2 instancje

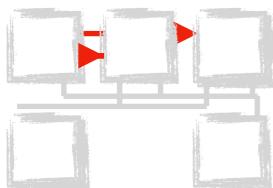
# refaktor fortune-cookie-service





# konroller

```
@Autowired  
private org.springframework.cloud.client.discovery.DiscoveryClient discoveryClient;  
  
@GetMapping("/fortune")  
@ResponseBody // automatic conversion to JSON  
public Fortune fortune() {  
    String fortune =  
        getDataFromService("DECISION-SERVICE") + " " + getDataFromService("ACTIVITY-SERVICE");  
    return new Fortune(fortune);  
}  
  
public String getDataFromService(String service) {  
    List<ServiceInstance> list = discoveryClient.getInstances(service);  
    if (list != null && list.size() > 0) {  
        URI uri = list.get(0).getUri();  
        if (uri != null) {  
            String responseInJSON = (new RestTemplate()).getForObject(uri, String.class);  
            return convertJSONResponseToSentence(responseInJSON);  
        }  
    }  
    return null;  
}
```

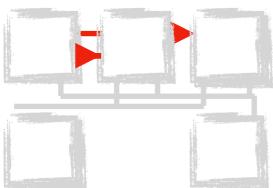


# konwersja JSON'a

```
private String convertJSONResponseToSentence(String responseInJSON) {  
    JSONArray dataFromService = JsonPath.read(responseInJSON, jsonPath: "$..*")  
    return (String) dataFromService.get(0);  
}
```

controller > Fortune

```
<dependency>  
    <groupId>com.jayway.jsonpath</groupId>  
    <artifactId>json-path</artifactId>  
    <version>2.4.0</version>  
</dependency>
```



# test

http://localhost:8080/fortune| GET

http://localhost:8080/fortune

header	value

Manage presets

**Send** Preview Add to collection

Body Headers (3) STATUS 200 TIME 83 ms

Pretty Raw Preview   JSON XML

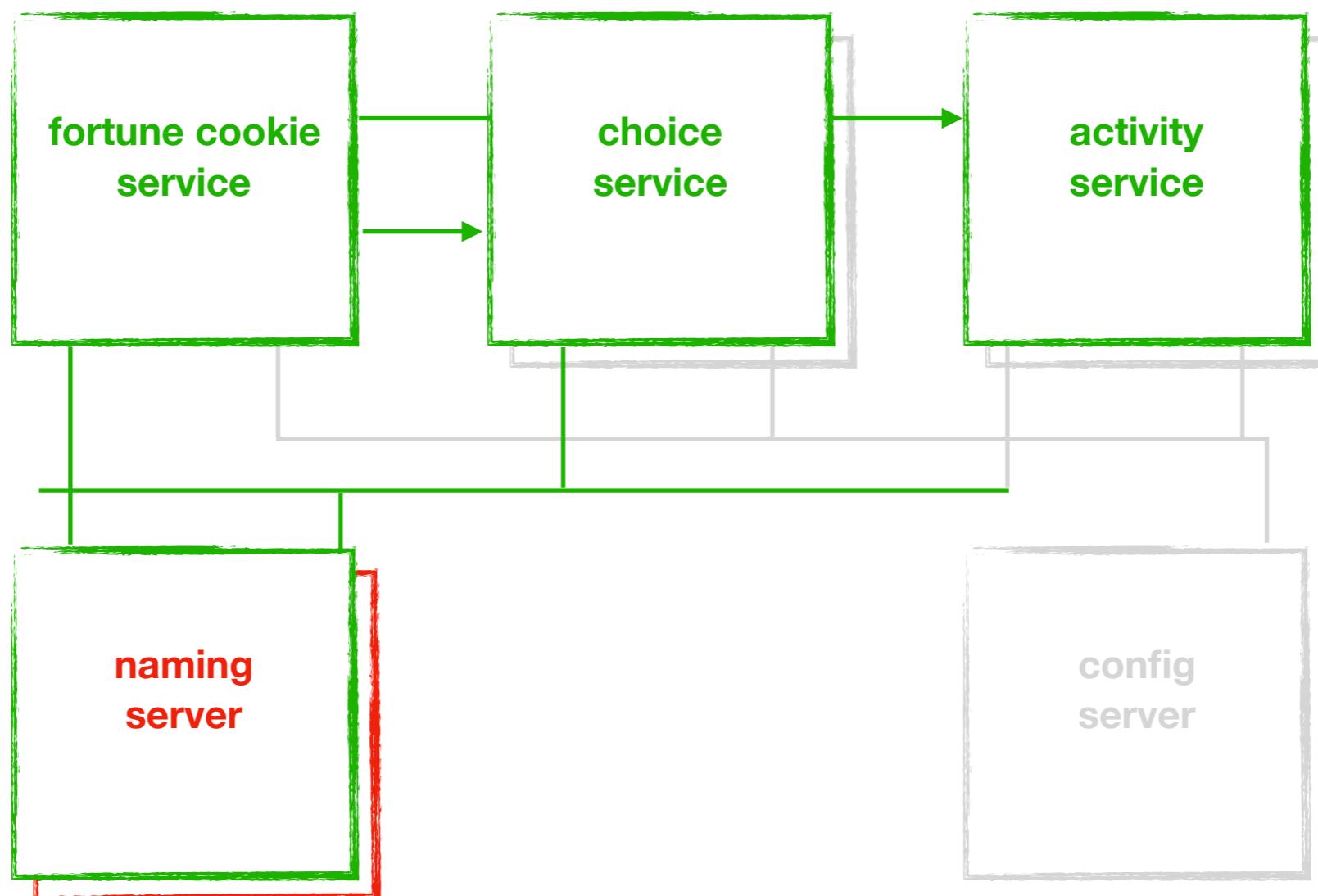
```
1 {  
2   "fortune": "Not a good idea to run a marathon"  
3 }
```

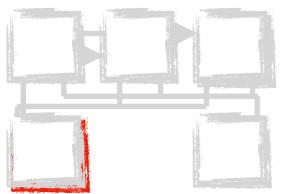
\* uwaga, będzie bug - co trzeba poprawić? : )

# I Serwisy i Eureka

- serwer Eureka
- **serwis fortune-cookie-service**
  - jako klient Eureki
- zadanie - utworzenie serwisów, klientów Eureki
  - decision-service
  - activity-service
- refaktor - fortune-cookie-service
  - korzysta z decision-service i activity-service
- **refaktor Eureka - 2 instancje**

# refaktor Eureka

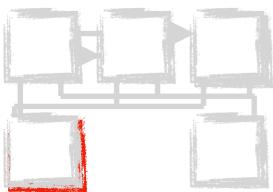




# dodanie wpisu

c:\WINDOWS\system32\drivers\etc\hosts

```
# START Eureka
127.0.0.1    eureka-primary
127.0.0.1    eureka-secondary
# END Eureka
```

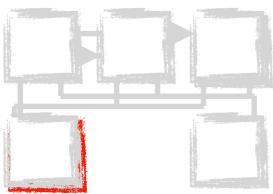


# plik konfiguracyjny - eureka

The screenshot shows the project structure of a Spring Cloud application named "eureka-server". The "application.yml" file is selected in the resources folder under the "src/main" directory. The code defines two profiles: "eureka-primary" and "eureka-secondary".

```
---  
spring:  
  profiles: eureka-primary  
  server:  
    port: 8761  
  eureka:  
    instance:  
      hostname: eureka-primary  
    client:  
      registerWithEureka: true  
      fetchRegistry: true  
      serviceUrl:  
        defaultZone: http://eureka-secondary:8762/eureka  
---  
spring:  
  profiles: eureka-secondary  
  server:  
    port: 8762  
  eureka:  
    instance:  
      hostname: eureka-secondary  
    client:  
      registerWithEureka: true  
      fetchRegistry: true  
      serviceUrl:  
        defaultZone: http://eureka-primary:8761/eureka
```

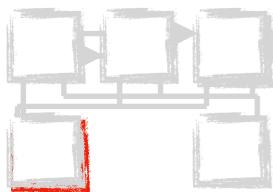




# plik konfiguracyjny - serwisy

**we wszystkich serwisach:**

```
eureka:  
  client:  
    serviceUrl:  
      defaultZone: http://eureka-primary:8761/eureka,http://eureka-secondary:8762/eureka
```

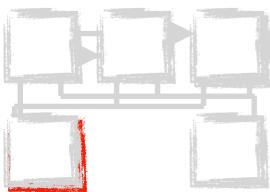


# użycie Multirun - 2 instancje

The screenshot shows the 'Run/Debug Configurations' dialog with the following details:

- Name:** TwoEurekas
- Choose configurations to run:**
  - Run 'EurekaServerApplicationPrimary'
  - Run 'EurekaServerApplicationSecondary'
- Checkboxes:**
  - Start configurations one by one
  - Mark the tab of failed configuration
  - Close tab of successfully completed configuration (and leave only tabs of failed configurations)
  - Re-use tab to run configuration (except the tab of failed configurations)
- Before launch: Activate tool window**:
  - There are no tasks to run before launch.
  - Show this page  Activate tool window
- Buttons:** Cancel, Apply, OK





# UI Eureka

 **spring Eureka**      HOME      LAST 1000 SINCE STARTUP

## System Status

Environment	test	Current time	2018-06-03T09:33:08 +0200
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	0

## DS Replicas

eureka-secondary
------------------

## Instances currently registered with Eureka

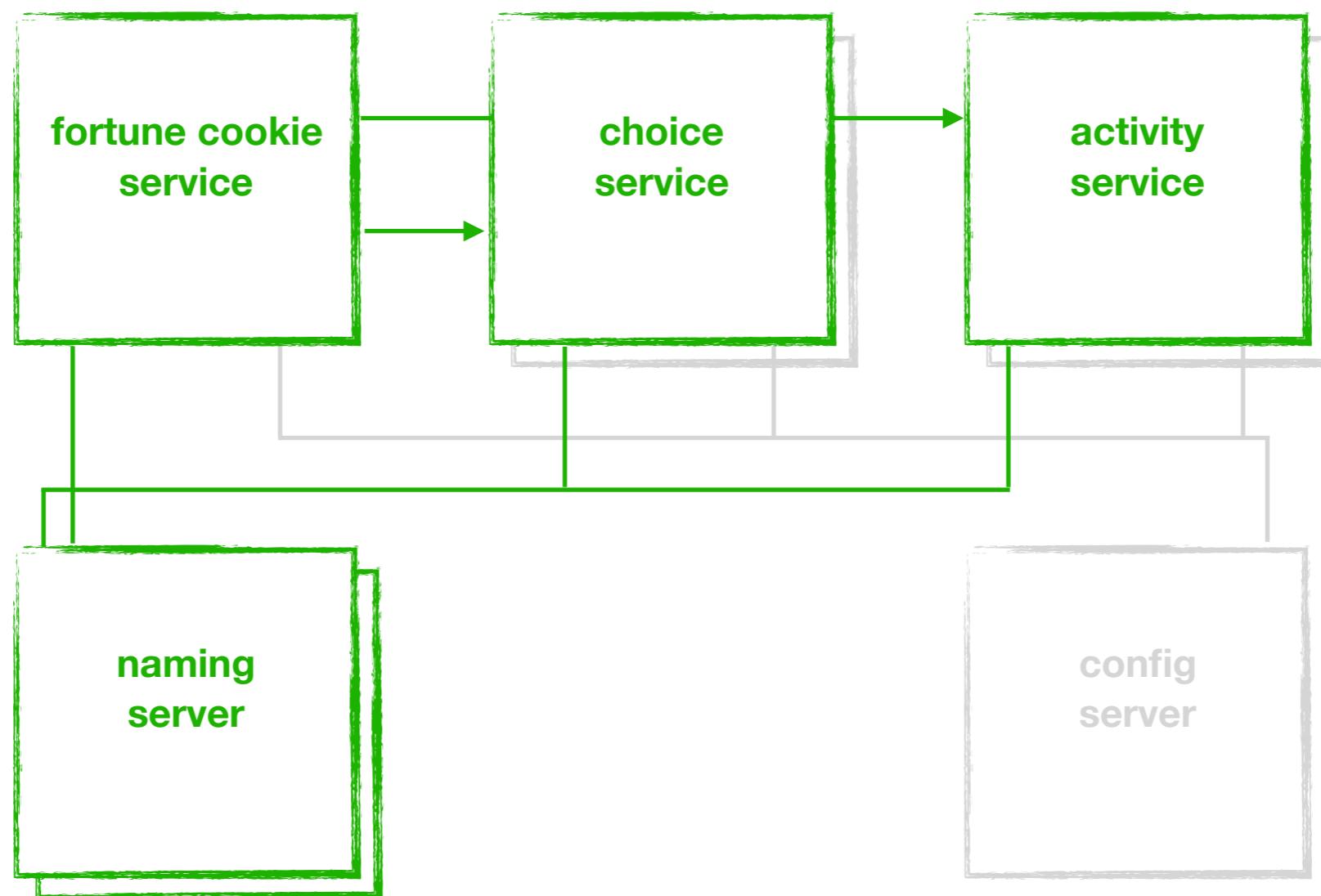
Application	AMIs	Availability Zones	Status
ACTIVITY-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:activity-service:8010
DECISION-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:decision-service:8000
FORTUNE-COOKIE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.8.102:fortune-cookie-service:8080
UNKNOWN	n/a (2)	(2)	UP (2) - 192.168.8.102:8762 , 192.168.8.102:8761

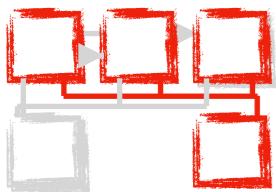
## General Info

37

  
Spring  
Cloud

# Serwisy i Eureka - podsumowanie

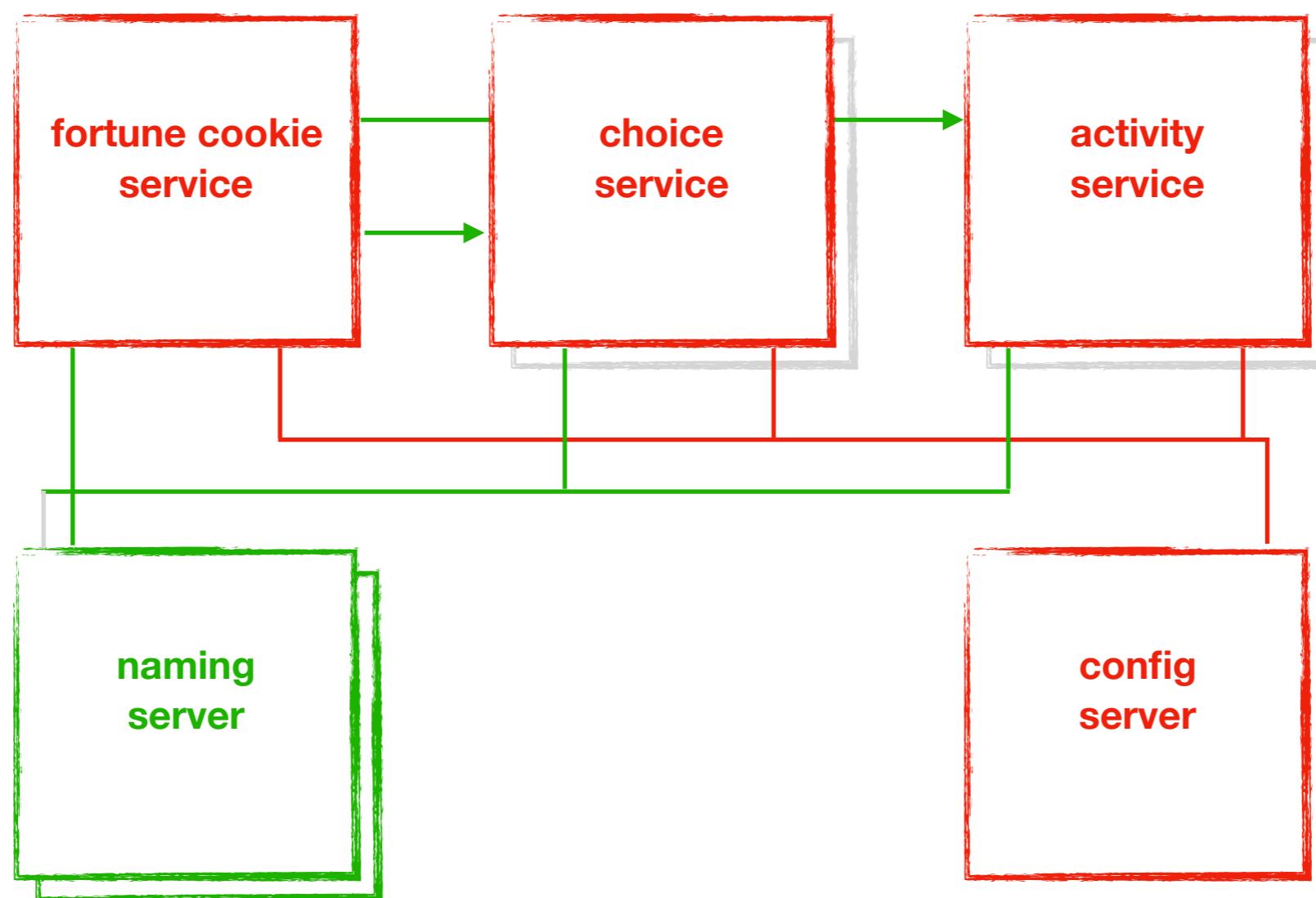


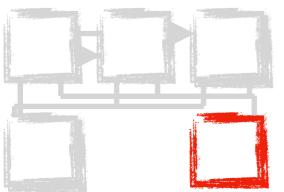


# II Serwisy i Config Server

- utworzenie repo Git, pierwszy commit
- komponent Config Server
- refactor / podłączenie activity-service do Config Server
  - profil default
  - profil dev
- zadanie - podłączenie pozostałych serwisów
  - analogicznie jak activity-service
    - decision-service
    - fortune-cookie-service

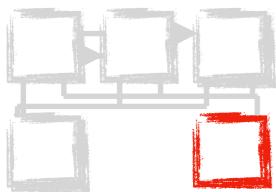
# II Serwisy i Config Server





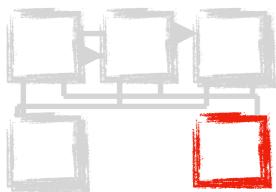
# II Serwisy i Config Server

- **utworzenie repo Git, pierwszy commit**
  - komponent Config Server
  - refactor / podłączenie activity-service do Config Server
    - profil default
    - profil dev
  - zadanie - podłączenie pozostałych serwisów
    - analogicznie jak activity-service
      - decision-service
      - fortune-cookie-service



# GIT i pierwszy commit

- utworzenie repo
  - git init
- utworzenie pliku konfiguracyjnego (vi lub edytor)
  - vi activity-service.yml
    - **zawartość pliku na kolejnym slajdzie**
- przygotowanie i commit pliku
  - git add -A && git commit -m 'add activity-service config, profiles: default & dev'



# GIT i pierwszy commit

eureka:

  client:

    serviceUrl:

      defaultZone: http://eureka-primary:8761/eureka,http://eureka-secondary:8762/eureka

---

spring:

  profiles: lazy

activities: to grab a bear,to eat a cookie,to take a nap

---

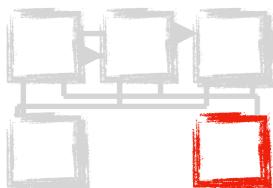
spring:

  profiles: crazy

activities: to run a marathon, to go hiking

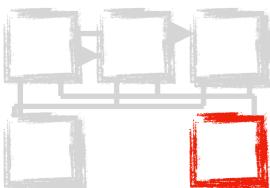
# II Serwisy i Config Server

- utworzenie repo Git, pierwszy commit
- **komponent Config Server**
- refactor / podłączenie activity-service do Config Server
  - profil default
  - profil dev
- zadanie - podłączenie pozostałych serwisów
  - analogicznie jak activity-service
    - decision-service
    - fortune-cookie-service



# szkielet projektu

The screenshot shows the Spring Initializer interface at [start.spring.io](https://start.spring.io). The top navigation bar includes icons for back, forward, and search, followed by a logo and the URL. The main header reads "SPRING INITIALIZER bootstrap your application now". Below the header, a large button says "Generate a [Maven Project] with [Java] and Spring Boot 2.0.2".  
  
The left side of the interface is labeled "Project Metadata" and contains fields for "Artifact coordinates", "Group" (set to "pl.altkom"), and "Artifact" (set to "config-server").  
  
The right side is labeled "Dependencies" and includes a "Search for dependencies" field containing "Web, Security, JPA, Actuator, Devtools...". A "Selected Dependencies" section shows "Config Server" with a green "X" button to remove it. A large green "Generate Project" button is at the bottom.  
  
At the bottom of the page, there's a note: "Don't know what to look for? Want more options? [Switch to the full version.](#)"



# konfig i @

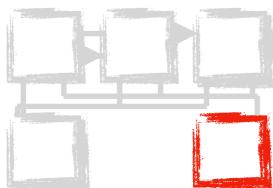
The screenshot shows the IntelliJ IDEA interface with a project named "config-server". The left sidebar displays the project structure, including ".idea", ".mvn", "src" (containing "main" and "java" packages), and "resources" (containing "application.yml"). The "target" folder is highlighted in yellow. The right side shows two code editors: "application.yml" and "ConfigServerApplication.java".

**application.yml** content:

```
1 spring:
2   cloud:
3     config:
4       server:
5         git:
6           uri: file:///Users/lukasz/Fortune/_6_config_server_activity/localrepo
7         application:
8           name: config-server
9       server:
10      port: 8888
```

**ConfigServerApplication.java** content:

```
1 package pl.altkom.configserver;
2
3 import ...
4
5 @SpringBootApplication
6 @EnableConfigServer
7 public class ConfigServerApplication {
8
9   public static void main(String[] args) { SpringApplication.run(ConfigServerAppl
10  )
11 }
12 }
```



# test profilu default

Normal Basic Auth Digest Auth OAuth 1.0 No environment ▾

http://localhost:8888/activity-service/default GET ↻

http://localhost:8888/activity-service/default

measured      venue

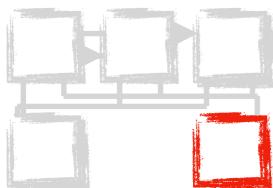
Manage presets

Send Preview Add to collection

Body Headers (3) STATUS 200 TIME 85 ms

Pretty Raw Preview [ ] JSON XML

```
1 {
  "name": "activity-service",
  "profiles": [
    "default"
  ],
  "label": null,
  "version": "2aba6913727daeabf290a871f710521efb846b93",
  "state": null,
  "propertySources": [
    {
      "name": "file:///Users/lukasz/Fortune/_6_config_server_activity/localrepo/activity-service.yml (document #0)",
      "source": {
        "eureka.client.serviceUrl.defaultZone": "http://eureka-primary:8761/eureka,http://eureka-secondary:8762/eureka"
      }
    }
  ]
}
```



# test profilu crazy

Normal Basic Auth Digest Auth OAuth 1.0 No environment -

http://localhost:8888/activity-service/crazy GET

Header Value Manage presets

Send Preview Add to collection

Body Headers (3) STATUS 200 TIME 60 ms

Pretty Raw Preview JSON XML

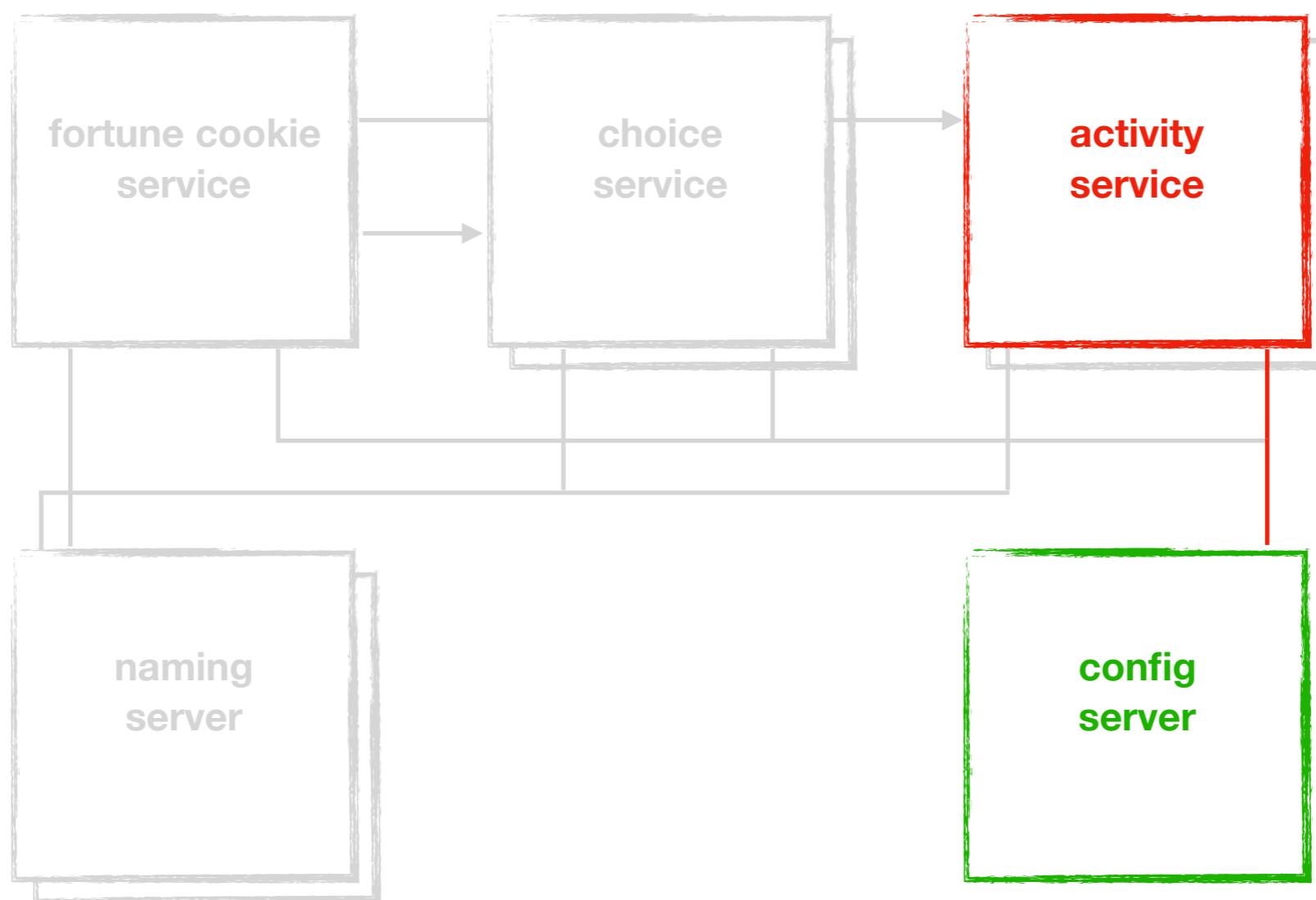
```
1 {
2     "name": "activity-service",
3     "profiles": [
4         "crazy"
5     ],
6     "label": null,
7     "version": "2aba6013727daeabf290a871f710521efb846b93",
8     "state": null,
9     "propertySources": [
10         {
11             "name": "file:///Users/lukasz/Fortune/_6_config_server_activity/localrepo/activity-service.yml (document #2)",
12             "source": {
13                 "spring.profiles": "crazy",
14                 "activities": "to run a marathon, to go hiking"
15             }
16         },
17         {
18             "name": "file:///Users/lukasz/Fortune/_6_config_server_activity/localrepo/activity-service.yml (document #0)",
19             "source": {
20                 "eureka.client.serviceUrl.defaultZone": "http://eureka-primary:8761/eureka,http://eureka-secondary:8762/eureka"
21             }
22         }
23     ]
24 }
```

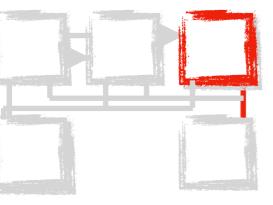


# II Serwisy i Config Server

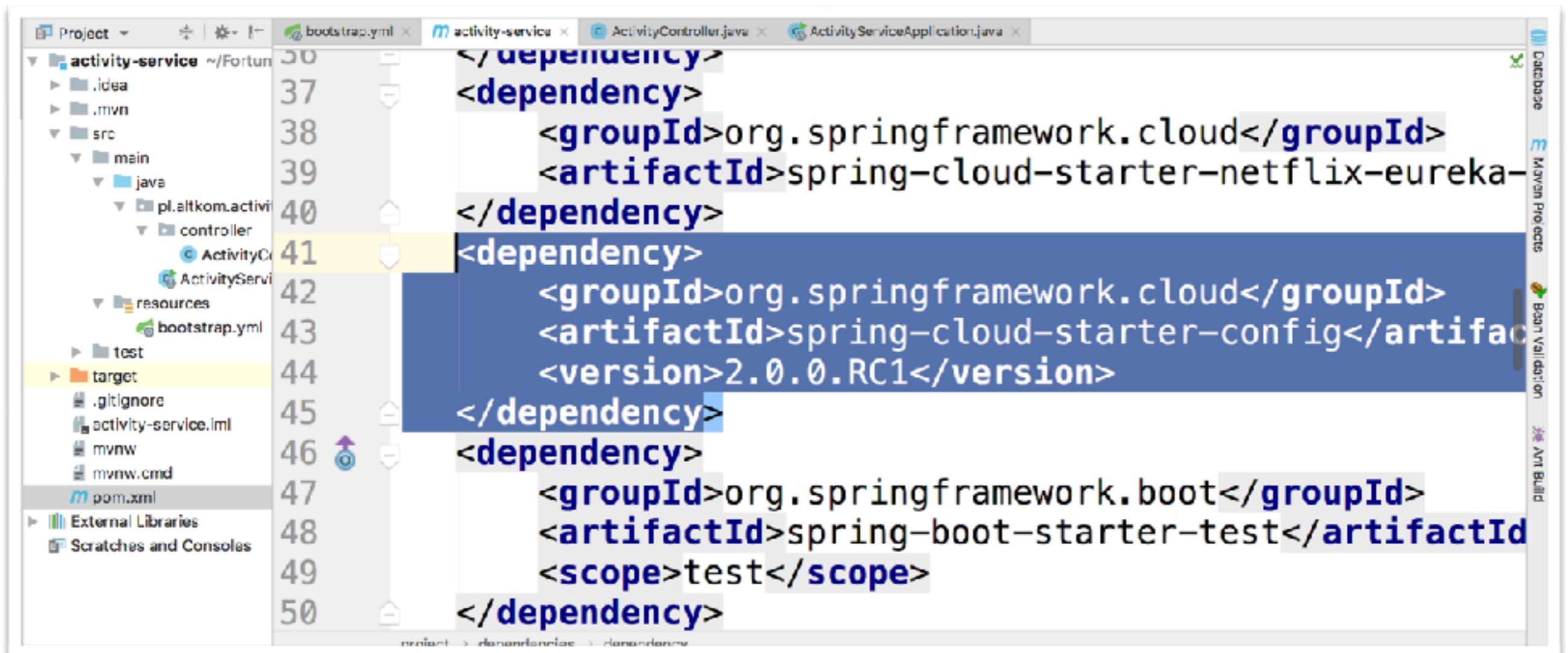
- utworzenie repo Git, pierwszy commit
- komponent Config Server
- **refactor / podłączenie activity-service do Config Server**
  - **profil lazy**
  - **profil crazy**
- zadanie - podłączenie pozostałych serwisów
  - analogicznie jak activity-service
    - decision-service
    - fortune-cookie-service

# refaktor activity-service





# dodanie zależności

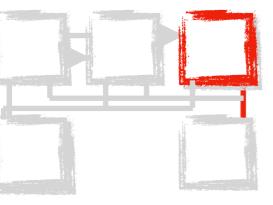


```
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-
```

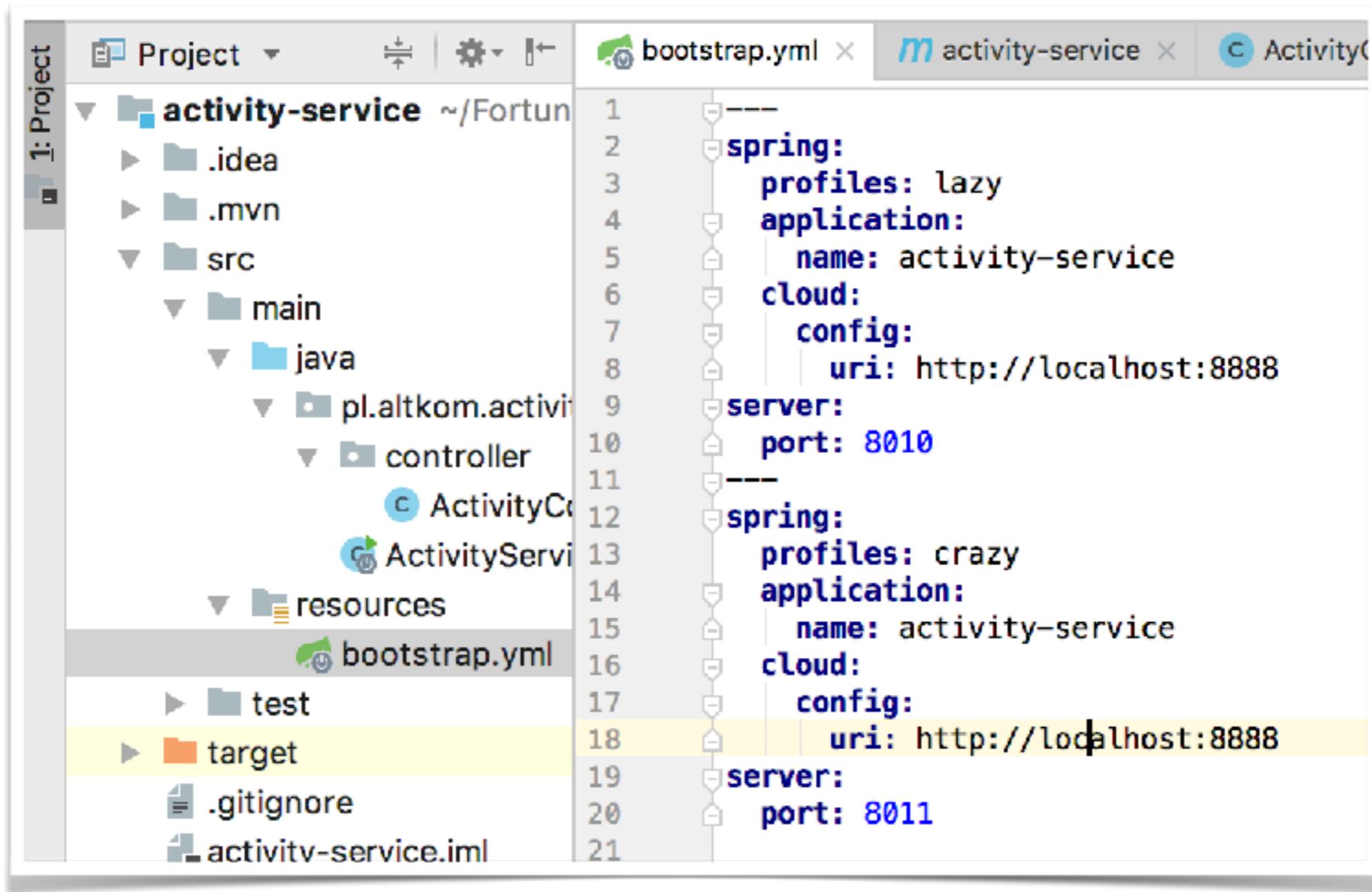
The screenshot shows the IntelliJ IDEA interface with the project 'activity-service' selected. The 'pom.xml' file is open in the central editor area, displaying the following XML code:

```
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-
```

The code block for adding the 'spring-cloud-starter-config' dependency is highlighted with a blue background. The XML structure includes dependencies for Eureka, Config, and Spring Boot Test.

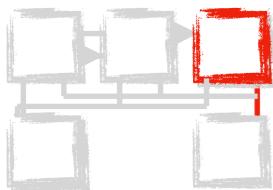


# plik konfiguracyjny



The screenshot shows a Java project structure in an IDE. The project is named "activity-service". The structure includes ".idea", ".mvn", "src" (containing "main" and "java" packages), "resources" (containing "bootstrap.yml"), "test", "target", ".gitignore", and "activity-service.iml". The "bootstrap.yml" file is open in the editor, displaying the following YAML configuration:

```
1  ----
2  spring:
3    profiles: lazy
4    application:
5      name: activity-service
6    cloud:
7      config:
8        uri: http://localhost:8888
9    server:
10   port: 8010
11  ----
12 spring:
13   profiles: crazy
14   application:
15     name: activity-service
16   cloud:
17     config:
18       uri: http://localhost:8888
19   server:
20     port: 8011
```



# uruchamianie z profilem

Name: ActivityServiceApplicationLazy  Share  Single instance only

Main class: pl.altkom.activityservice.ActivityServiceApplication

VM options: -Dspring.profiles.active=Lazy

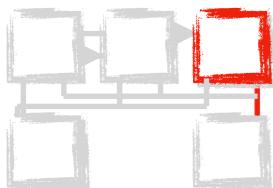
Program arguments:

Working directory:

Environment variables:

Use classpath of module: activity-service  Include dependencies with "Provided" scope

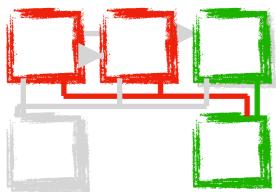
JRE: Default (1.8 - SDK of 'activity-service' module)



# test obuinstancji

The screenshot shows two parallel API requests in the Postman application:

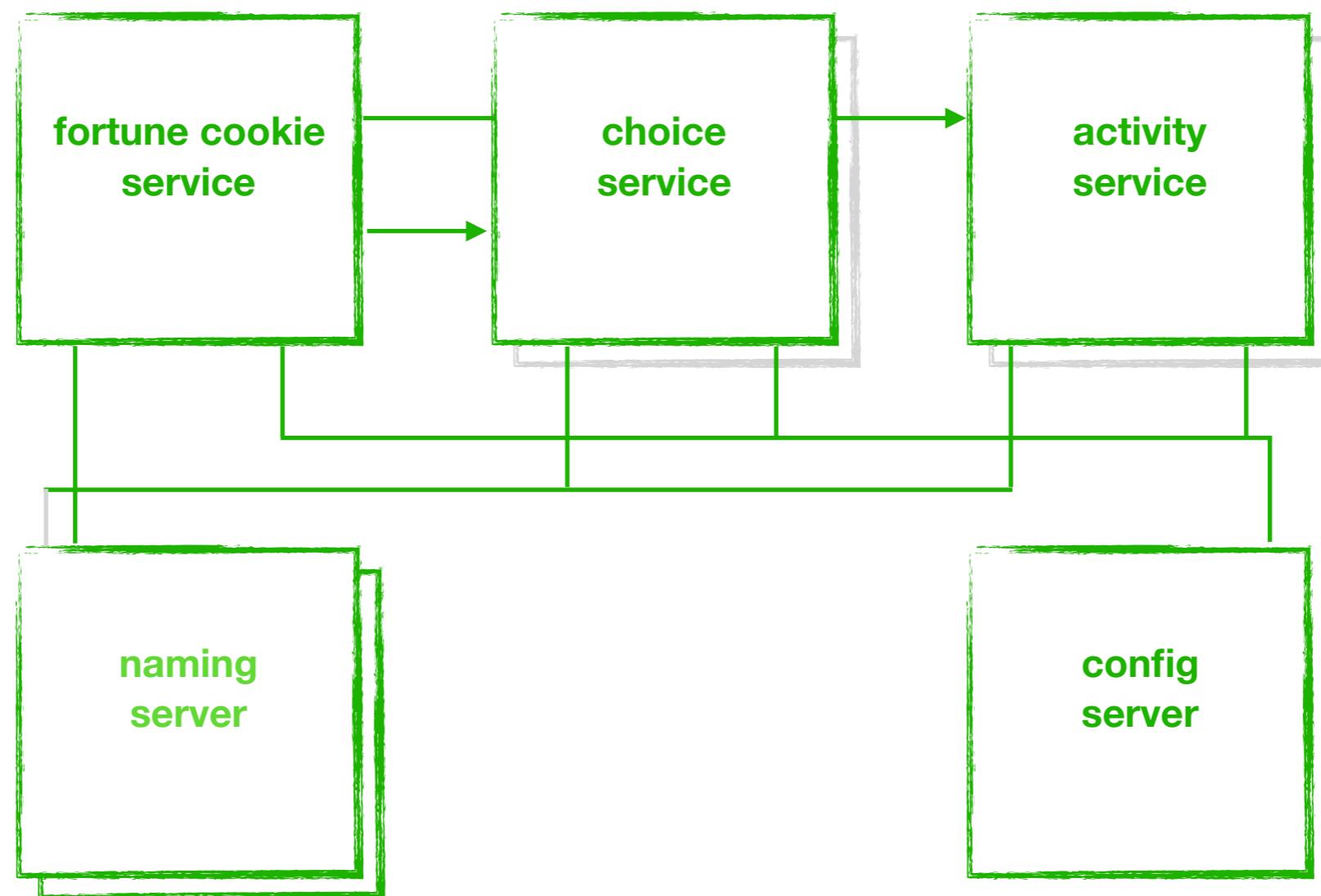
- Request 1 (Left):** URL: `http://localhost:8010/`, Method: GET. Status: 200, Time: 11 ms. Body: { "activity": "to grab a bear" }.
- Request 2 (Right):** URL: `http://localhost:8011/`, Method: GET. Status: 200, Time: 30 ms. Body: { "activity": "to run a marathon" }.



# zadanie

- utworzenie repo Git, pierwszy commit
- komponent Config Server
- refactor / podłączenie activity-service do Config Server
  - profil lazy
  - profil crazy
- **zadanie - podłączenie pozostałych serwisów**
  - **analogicznie jak activity-service**
    - **decision-service (profil polite, rough)**
    - **fortune-cookie-service (bez profilu)**

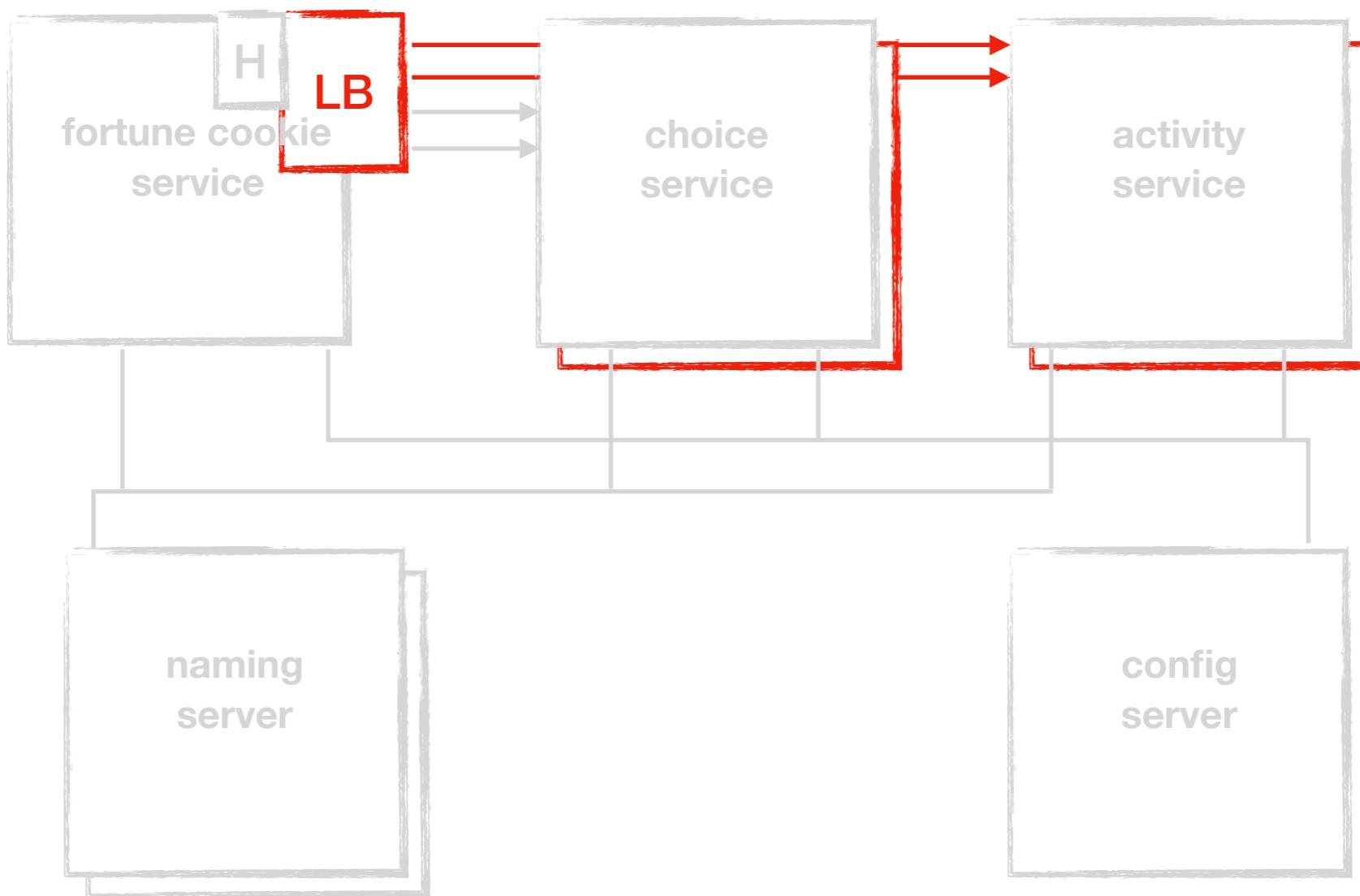
# Serwisy i Config Server - podsumowanie



# III Ribbon

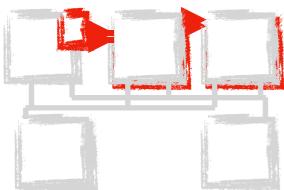
- refaktor fortune-cookie-service
  - użycie Ribbon
- test
  - uruchomienie podwójnych instancji
    - decision-service
      - profile polite i rough
    - activity-service
      - profile lazy i crazy
  - w logach fortune-cookie-service
    - powinno być widać instancje w current list of Servers
  - requesty do fortune-cookie-service
    - powinny zwracać mix 4 instancji (w configu różne dane dla każdej)

# III Ribbon



# III Ribbon

- **refaktor fortune-cookie-service**
  - **użycie Ribbon**
- test
  - uruchomienie podwójnych instancji
    - decision-service
      - profile polite i rough
    - activity-service
      - profile lazy i crazy
  - requesty do fortune-cookie-service
  - powinny zwracać mix 4 instancji (w configu różne dane dla każdej)
  - w logach fortune-cookie-service c
  - powinno być widać instancje w current list of Servers



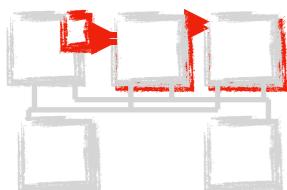
# refaktor - kontroler i główna

```
@Autowired  
@Qualifier("ribbonRestTemplate")  
private RestTemplate ribbonRestTemplate;  
  
@GetMapping("/fortune")  
@ResponseBody // automatic conversion to JSON  
public Fortune fortune() {  
    String fortune =  
        getDataFromService("DECISION-SERVICE") + " " + getDataFromService("ACTIVITY-SERVICE");  
    return new Fortune(fortune);  
}  
  
public String getDataFromService(String service) {  
    String responseInJSON = (ribbonRestTemplate).getForObject(url: "http://" + service, String.class);  
    return convertJSONResponseToSentence(responseInJSON);  
}  
FortuneController > ribbonRestTemplate  
CookieServiceApplication.java ✘  
  
import ...  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class FortuneCookieServiceApplication {  
  
    public static void main(String[] args) { SpringApplication.run(FortuneCookieServiceApplication.class, args); }  
  
    @Bean  
    @LoadBalanced  
    RestTemplate ribbonRestTemplate() {  
        return new RestTemplate();  
    }  
}
```

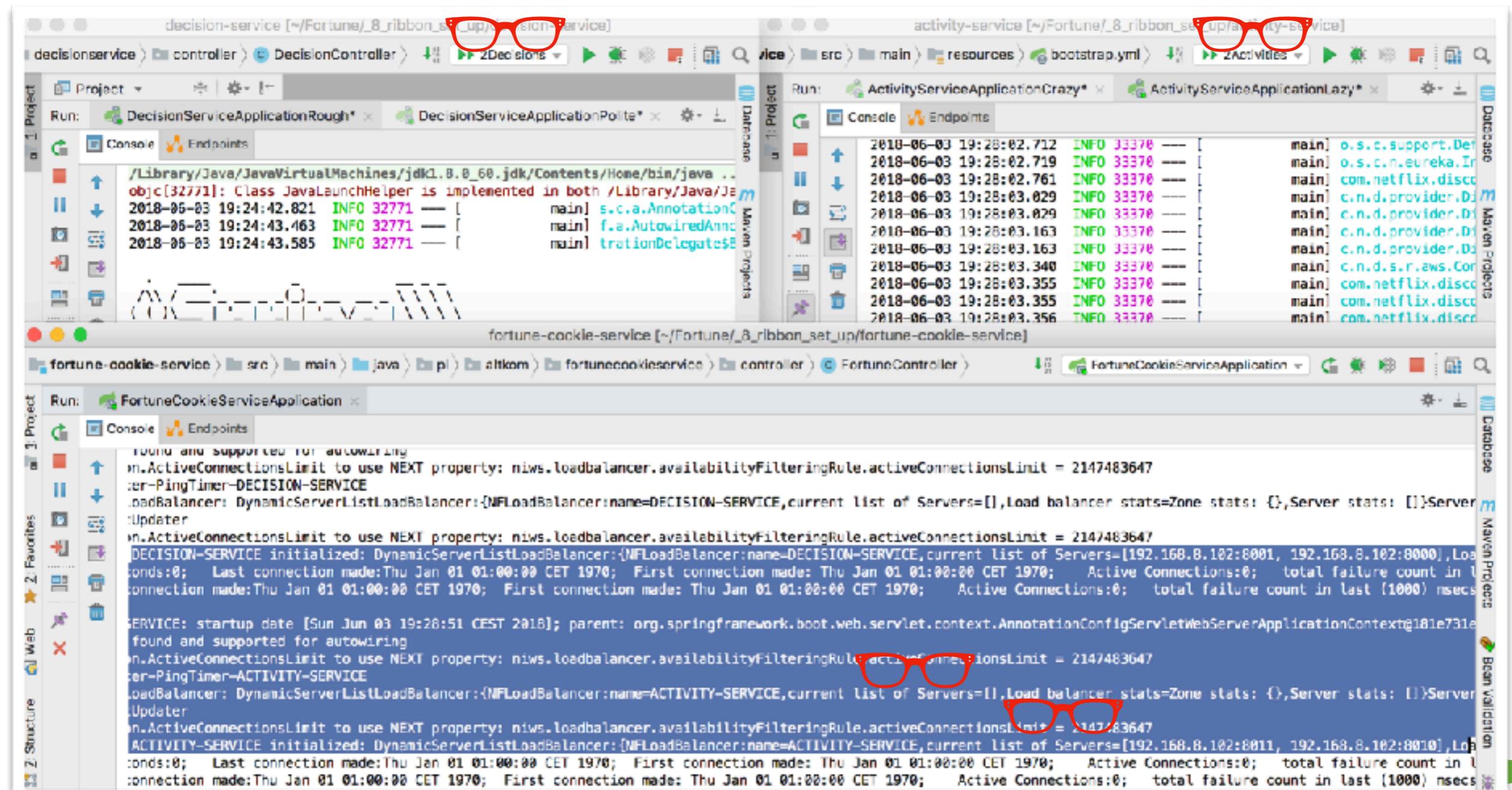


# III Ribbon

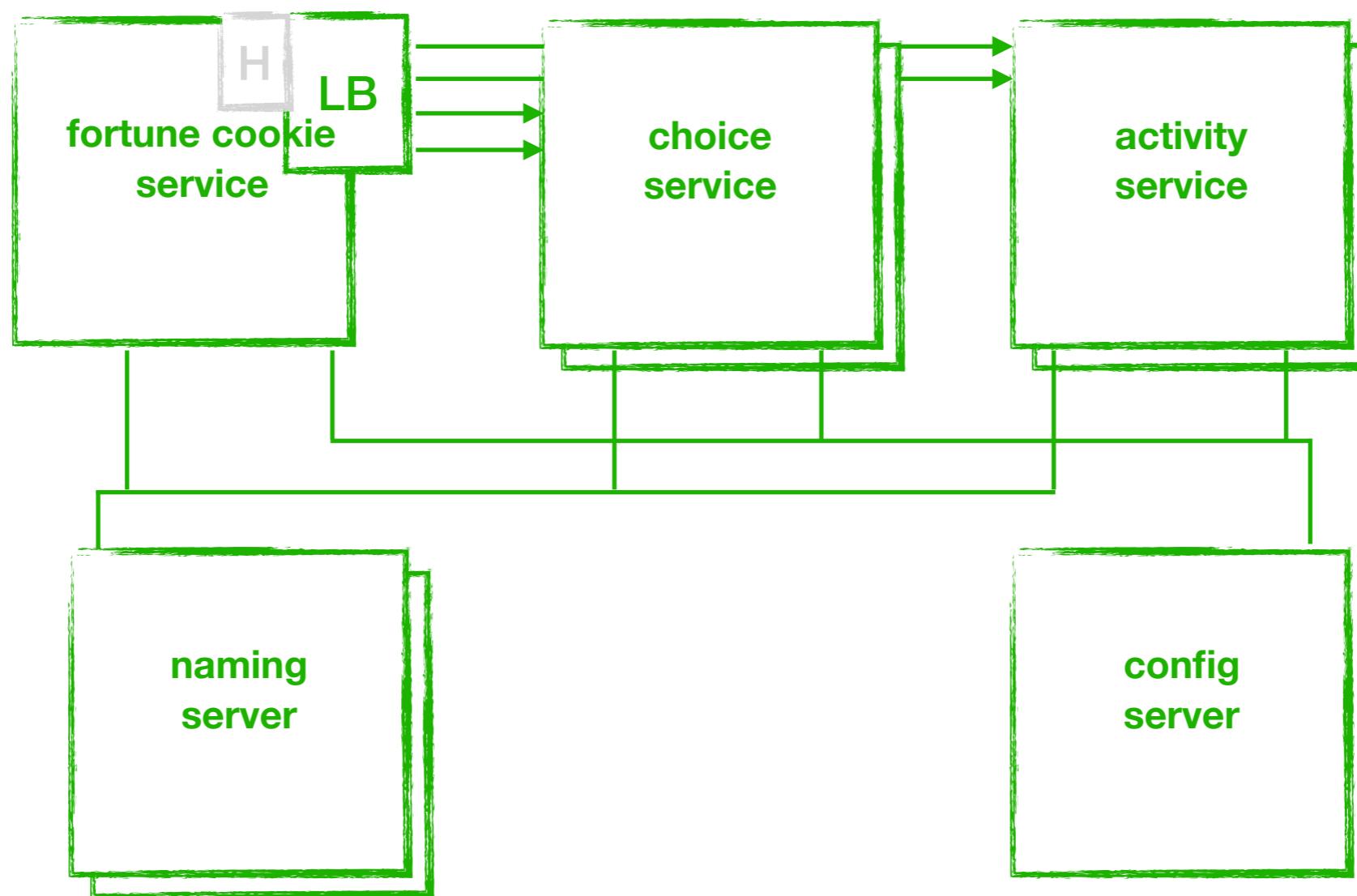
- refaktor fortune-cookie-service
  - użycie Ribbon
- **test**
  - **uruchomienie podwójnych instancji**
    - decision-service
      - profile polite i rough
    - activity-service
      - profile lazy i crazy
  - **w logach** fortune-cookie-service
    - powinno być widać instancje w **current list of Servers**
  - **requesty** do fortune-cookie-service
    - **powinny zwracać mix 4** instancji (w configu różne dane dla każdej)

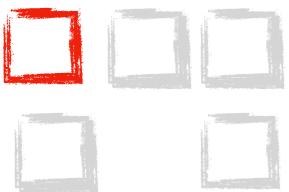


test



# Ribbon - podsumowanie



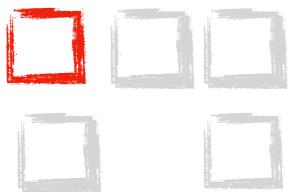


# IV Feign

- refaktor fortune-cookie-service
  - zależności maven
  - dodanie interfejsów
  - `@EnableFeignClients`
  - użycie w kontrolerze

# IV Feign

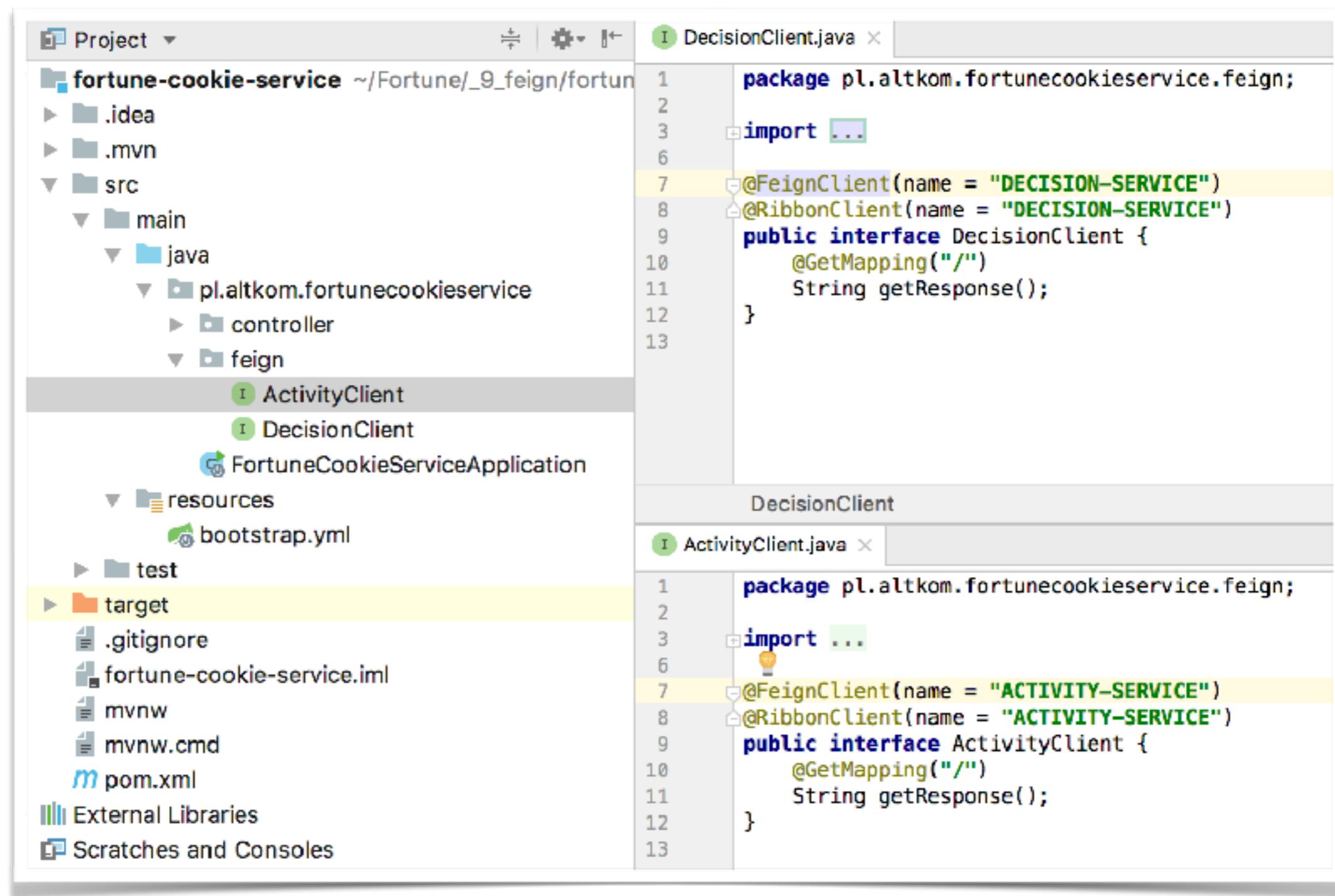




# zależności maven

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

# dodanie interfejsów



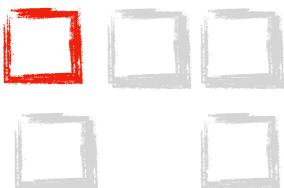
The screenshot shows a Java project structure in an IDE. The project is named "fortune-cookie-service". The source code is organized into packages: "pl.altkom.fortunecookieservice" (containing "controller" and "feign" sub-packages) and "pl.altkom.fortunecookieservice.feign" (containing "ActivityClient" and "DecisionClient"). The "target" folder is highlighted in yellow.

**DecisionClient.java:**

```
1 package pl.altkom.fortunecookieservice.feign;
2
3 import ...
4
5
6
7 @FeignClient(name = "DECISION-SERVICE")
8 @RibbonClient(name = "DECISION-SERVICE")
9 public interface DecisionClient {
10     @GetMapping("/")
11     String getResponse();
12 }
13
```

**ActivityClient.java:**

```
1 package pl.altkom.fortunecookieservice.feign;
2
3 import ...
4
5
6
7 @FeignClient(name = "ACTIVITY-SERVICE")
8 @RibbonClient(name = "ACTIVITY-SERVICE")
9 public interface ActivityClient {
10     @GetMapping("/")
11     String getResponse();
12 }
13
```



# użycie w kontrolerze

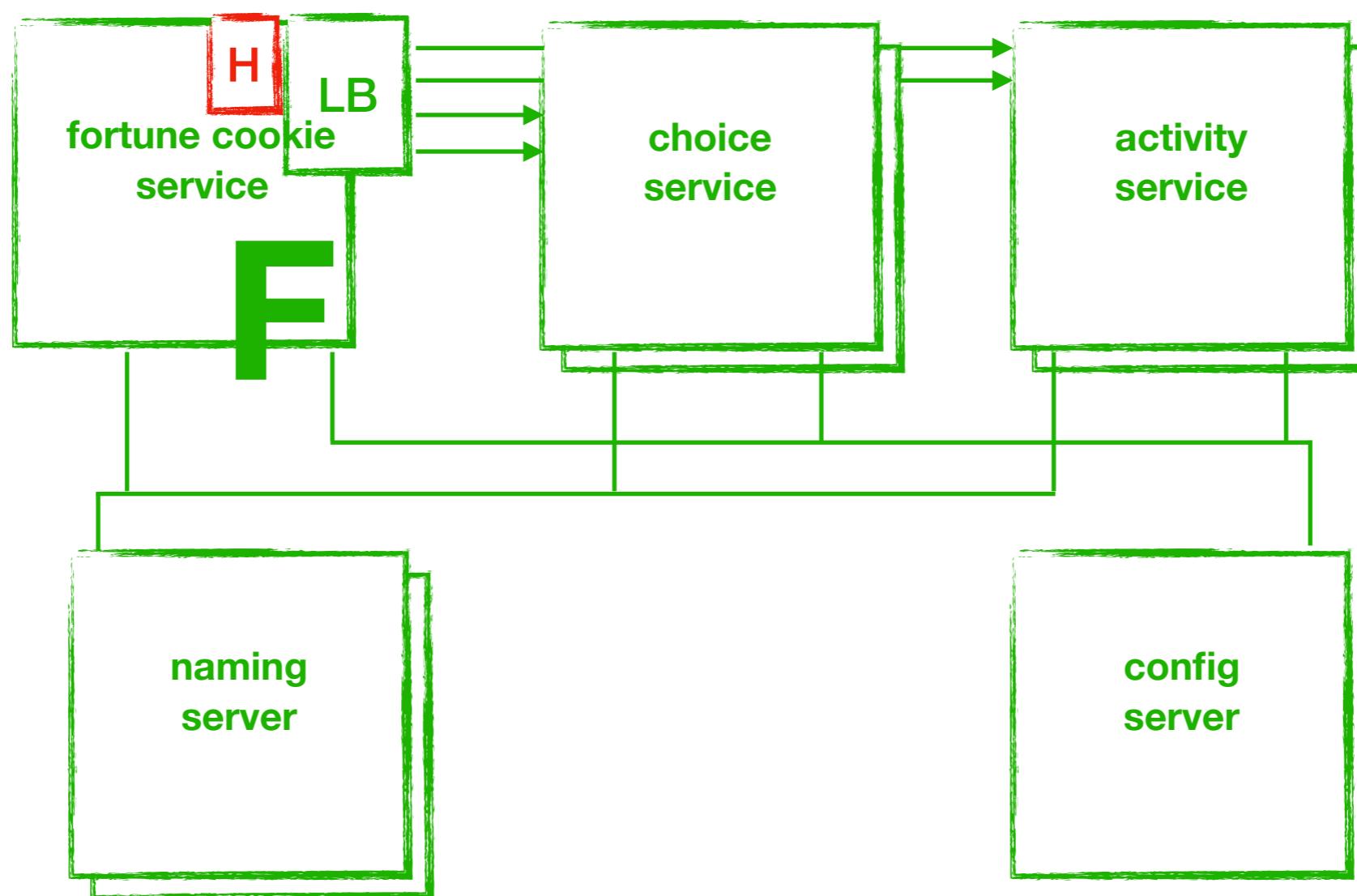
```
@RestController
public class FortuneController {

    @Autowired
    private ActivityClient activityClient;
    @Autowired
    private DecisionClient decisionClient;

    @GetMapping("/fortune")
    @ResponseBody
    public Fortune fortune() {
        String fortune =
            convertJSONResponseToSentence(decisionClient.getResponse()) + " " +
            convertJSONResponseToSentence(activityClient.getResponse());
        return new Fortune(fortune);
    }

    private String convertJSONResponseToSentence(String responseInJSON) {
        JSONArray dataFromService = JsonPath.read(responseInJSON, jsonPath: "$..*");
        return (String) dataFromService.get(0);
    }
}
```

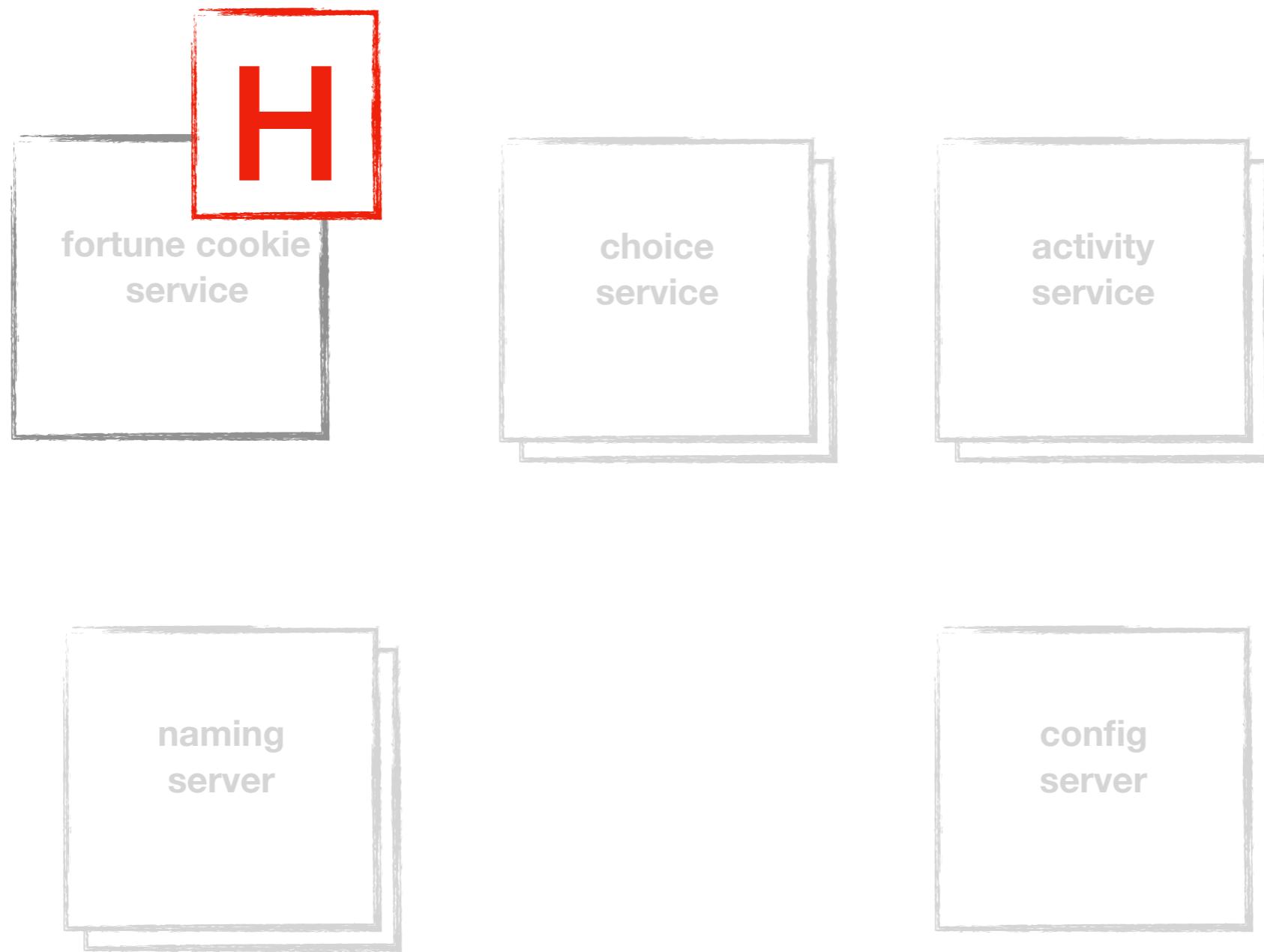
# V Hystrix

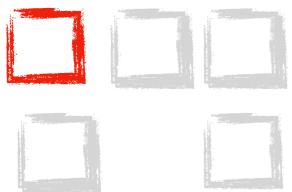


# V Hystrix

- refaktor fortune-cookie-service
  - dodaj zależność w pom
  - dodaj adnotacjw w głównej
  - w adnotacjach @FeignClient ustaw fallback'i
  - zaimplementuj fallback'i

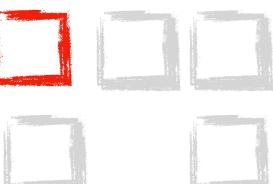
# V Hystrix



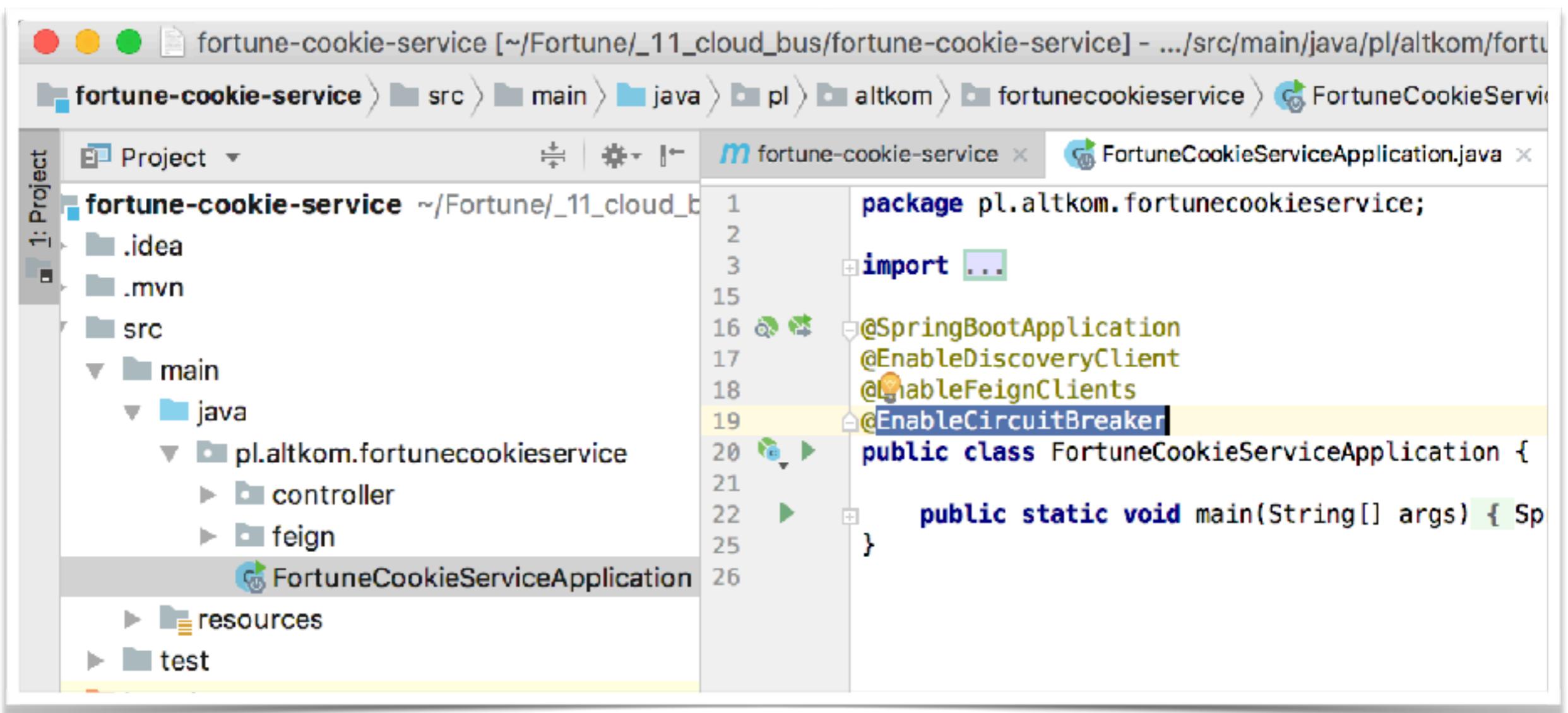


# dodaj zależność w pom

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hystrix</artifactId>
    <version>1.4.4.RELEASE</version>
</dependency>
```



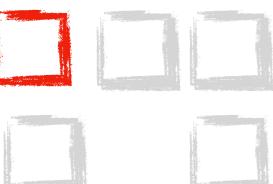
# dodaj adnotację w głównej



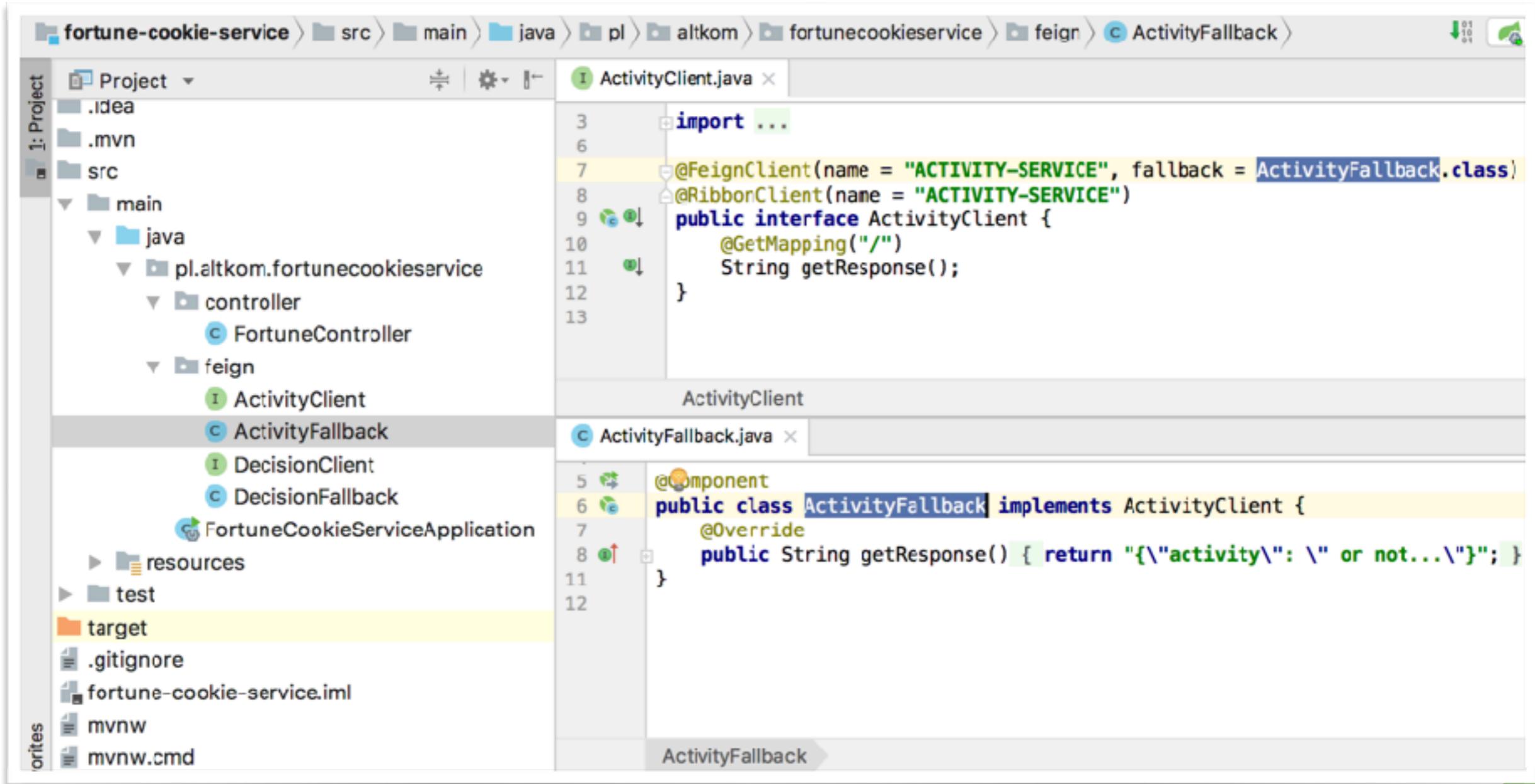
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** fortune-cookie-service [~/Fortune/\_11\_cloud\_bus/fortune-cookie-service] - .../src/main/java/pl/altkom/fortunecookie/service
- Breadcrumbs:** fortune-cookie-service > src > main > java > pl > altkom > fortunecookieservice > FortuneCookieServiceApplication.java
- Toolbars:** Project, View, Editor, Run, Build, Tools, Help.
- Code Editor:** The file FortuneCookieServiceApplication.java contains the following code:

```
1 package pl.altkom.fortunecookieeservice;
2
3 import ...
4
5 @SpringBootApplication
6 @EnableDiscoveryClient
7 @EnableFeignClients
8 @EnableCircuitBreaker
9
10 public class FortuneCookieServiceApplication {
11
12     public static void main(String[] args) { Sp
13 }
14 }
```
- Project Tree:** Shows the project structure with folders .idea, .mvn, src, main, java, pl.altkom.fortunecookieeservice, controller, feign, and resources.



# fallback: activity-service



The screenshot shows the IntelliJ IDEA interface with the project structure and code editor for a Spring Cloud application.

**Project Structure:**

- fortune-cookie-service
- src
- main
- java
- pl.altkom.fortunecookieservice
- controller
- feign

  - ActivityClient
  - ActivityFallback
  - DecisionClient
  - DecisionFallback

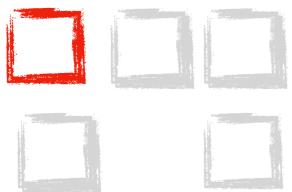
- FortuneCookieServiceApplication
- resources
- test
- target
- .gitignore
- fortune-cookie-service.iml
- mvnw
- mvnw.cmd

**Code Editor (ActivityClient.java):**

```
import ...  
@FeignClient(name = "ACTIVITY-SERVICE", fallback = ActivityFallback.class)  
@RibbonClient(name = "ACTIVITY-SERVICE")  
public interface ActivityClient {  
    @GetMapping("/")  
    String getResponse();  
}
```

**Code Editor (ActivityFallback.java):**

```
@Component  
public class ActivityFallback implements ActivityClient {  
    @Override  
    public String getResponse() { return "{\"activity\": \"or not...\"}"; }  
}
```



# zadania

- implementacja fallback dla DecisionClient
- test
  - po wyłączeniu activity-service
  - po wyłączeniu decision-service

# Hystrix - podsumowanie

