Patrones de Cloud Native con MicroProfile

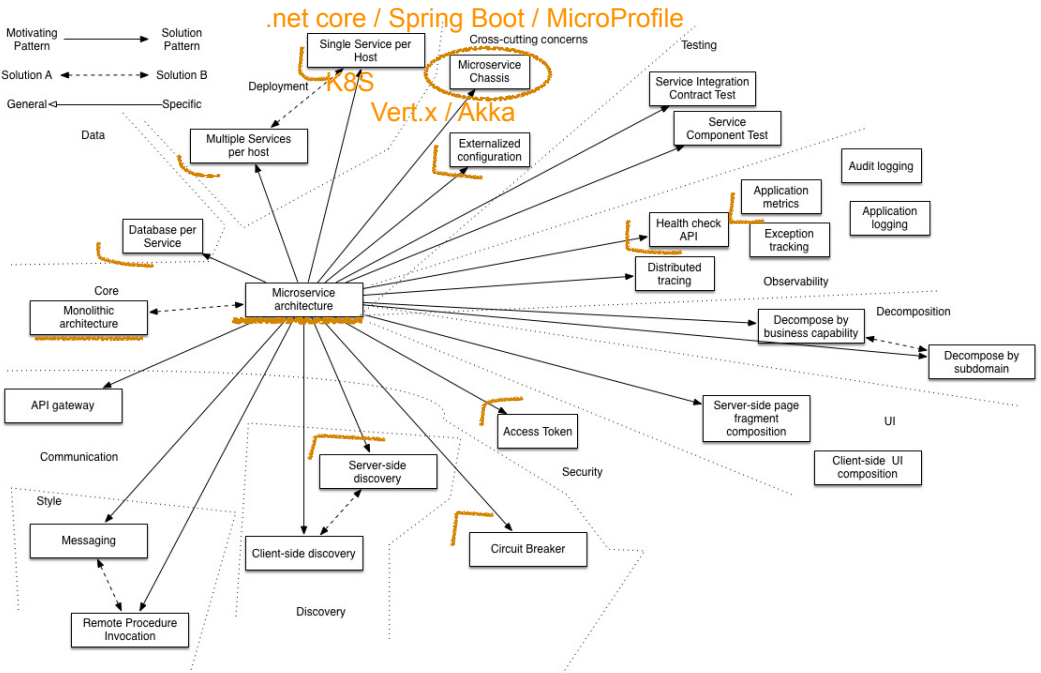
Víctor Orozco

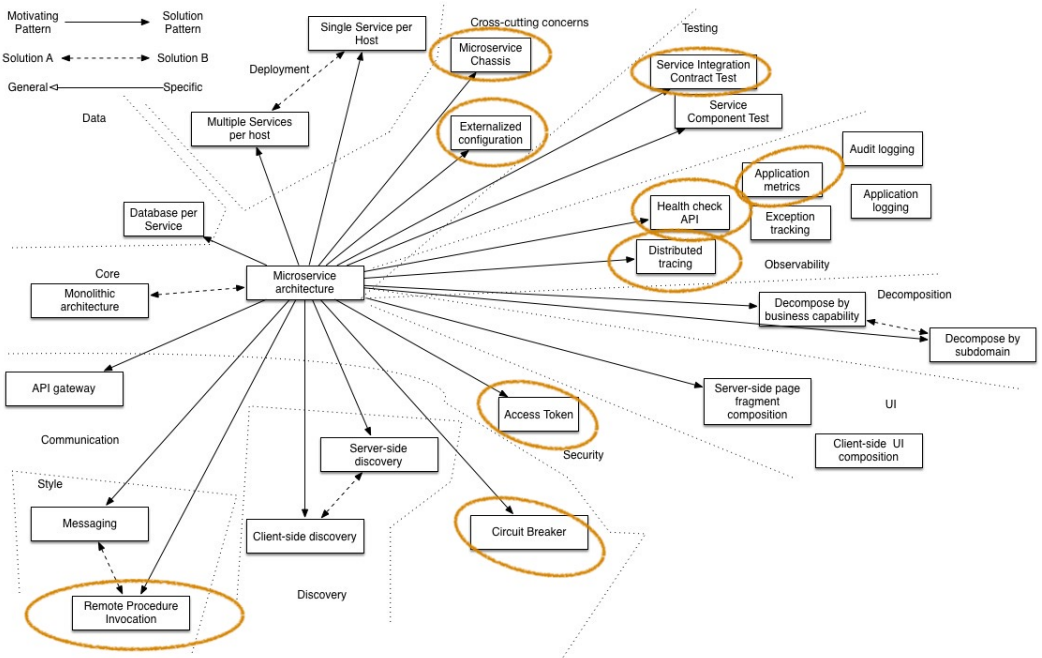
19 de noviembre de 2020

Nabenik

ACADEMIK

Patrones de microservicios


.net core / Spring Boot / MicroProfile
K8S
Vert.x / Akka
Motivating Pattern
Solution Pattern
Solution A
Solution B
General
Specific
Data
Deployment
Single Service per Host
Multiple Services per host
Database per Service
Cross-cutting concerns
Microservice Chassis
Externalized configuration
Testing
Service Integration Contract Test
Service Component Test
Audit logging
Application metrics
Application logging
Health check API
Exception tracking
Distributed tracing
Observability
Core
Monolithic architecture
Microservice architecture
Decomposition
Decompose by business capability
Decompose by subdomain
API gateway
Communication
Style
Messaging
Remote Procedure Invocation
Server-side discovery
Client-side discovery
Discovery
Access Token
Circuit Breaker
Security
Server-side page fragment composition
Client-side UI composition
UI


Motivating Pattern
Solution Pattern
Solution A
Solution B
General
Specific
Data
Deployment
Single Service per Host
Multiple Services per host
Database per Service
Cross-cutting concerns
Microservice Chassis
Externalized configuration
Testing
Service Integration Contract Test
Service Component Test
Audit logging
Application metrics
Application logging
Health check API
Exception tracking
Distributed tracing
Observability
Core
Monolithic architecture
Microservice architecture
Decomposition
Decompose by business capability
Decompose by subdomain
API gateway
Server-side page fragment composition
UI
Client-side UI composition
Access Token
Security
Communication
Server-side discovery
Style
Messaging
Client-side discovery
Circuit Breaker
Remote Procedure Invocation
Discovery

Eclipse MicroProfile

Eclipse MicroProfile

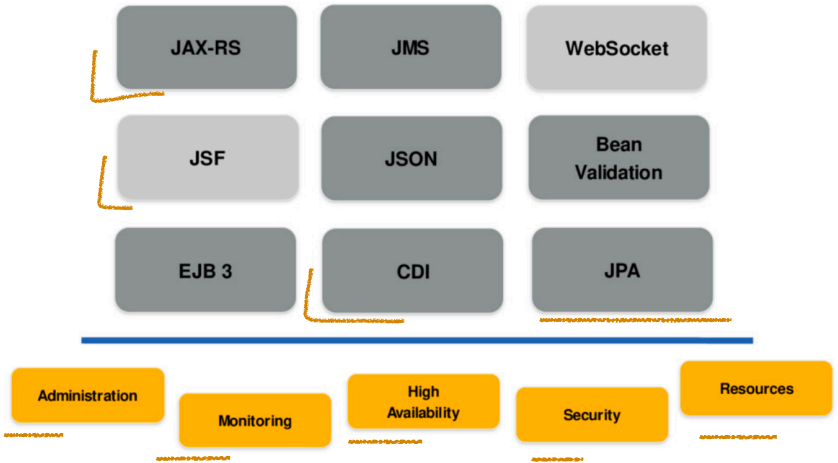
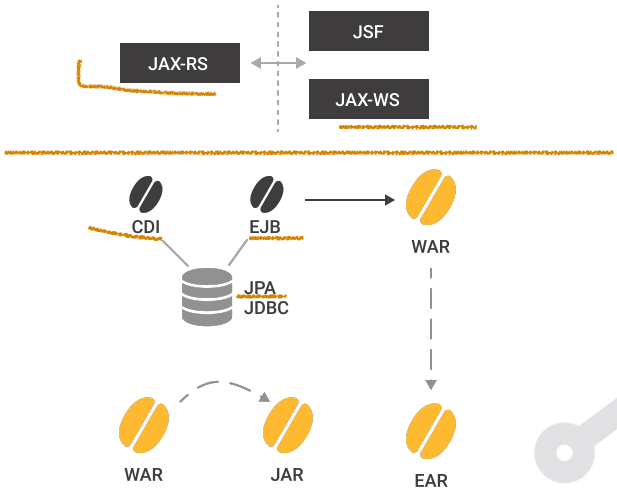


Figura 1: Credito: Reza Rahman

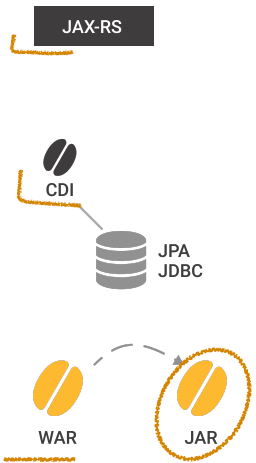
Eclipse MicroProfile



ACADEMIK

JSON-P

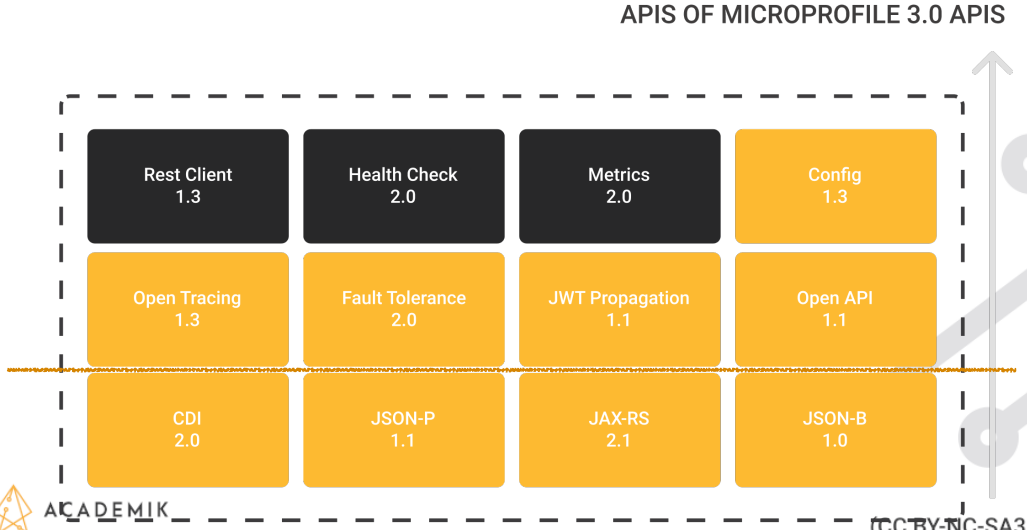
Eclipse MicroProfile



JSON-P

ACADEMIK

Eclipse MicroProfile



ACADEMIK

Eclipse MicroProfile - Implementaciones

Bibliotecas

- SmallRye (Red Hat)
- Hammock
- Apache Geronimo
- Fujitsu Launcher

JEAS - Fat Jar

- Dropwizard
- KumuluzEE
- Helidon (Oracle)
- Open Liberty (IBM)
- Quarkus (Red Hat)

ACADEMIK

Eclipse MicroProfile - Implementaciones

Micro server

- Payara Micro
- TomEE MicroProfile

Full server

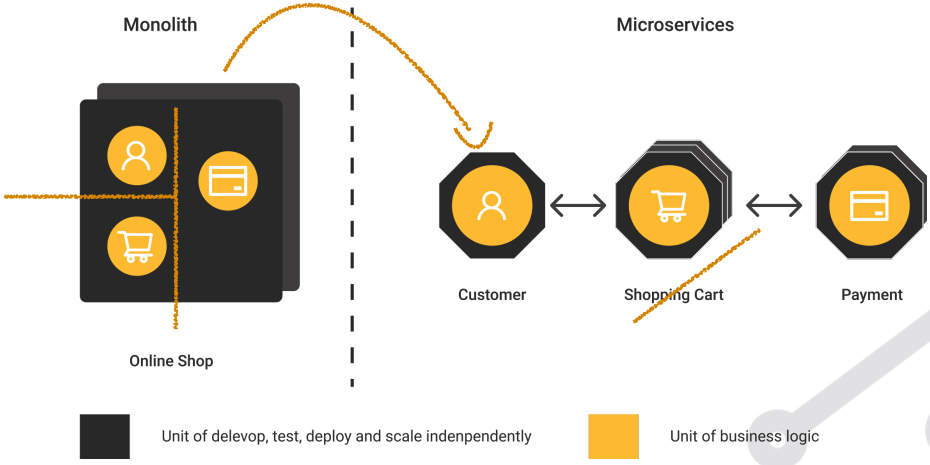
- Payara Application Server
- Apache TomEE
- JBoss Application Server / Wildfly Application Server
- WebSphere Liberty (IBM)

https://wiki.eclipse.org/MicroProfile/Implementation

Eclipse MicroProfile - Objetivos

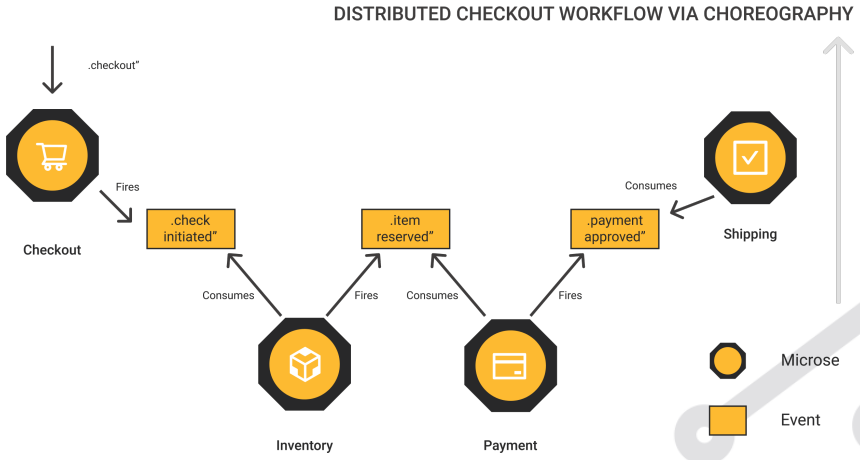
Demo MicroProfile Starter

Eclipse MicroProfile



ACADEMIK

Eclipse MicroProfile - Coreografia



ACADEMIK

Eclipse MicroProfile - Orquestación

DISTRIBUTED CHECKOUT WORKFLOW VIA ORCHESTRATION

Microservice

Command

ACADEMIK

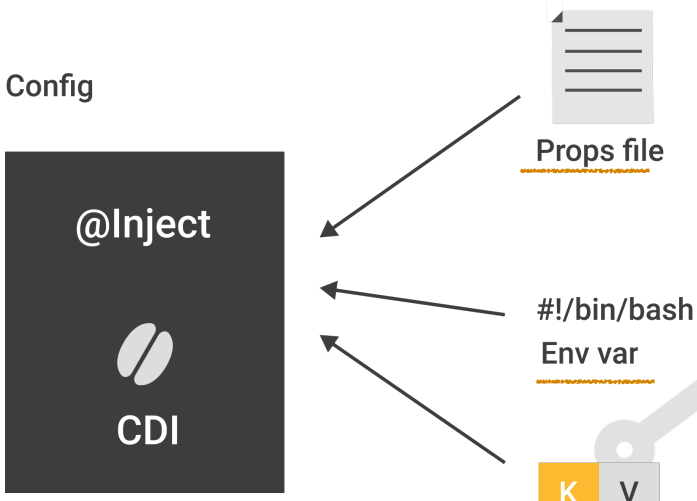
Eclipse MicroProfile - Cross-cutting concerns

- Health checks & Metrics
- Resilence & Fault Tolerance
- Configuration
- Authentication & Authorization
- Standarized documentation
- Tracing

ACADEMIK

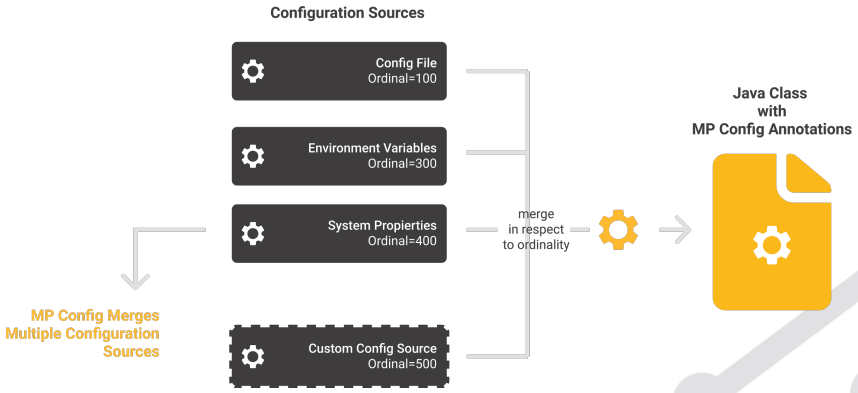
Eclipse MicroProfile - APIs

Config



ACADEMIK

Config



ACADEMIK

Config

```
@Inject
@ConfigProperty(name = .ºmdbservice.url")
String omdbDaemonServiceUrl;
```

Ext. de la configuración (VM, Docker, Kubernetes)

Config

```
@Inject
@ConfigProperty(name = "application.currency")
private String currency;

@Inject
@ConfigProperty(name = "application.list.maxSize",
        defaultValue="10")
private Integer maxSize;
```

Config

No CDI, no hay problema

```
final Config config = ConfigProvider.getConfig();
config.getValue("application.curreny", String.class);
config.getOptionalValue("application.list.maxSize",
        Integer.class);
```

Config

Propiedades dinámicas

```
@Inject @ConfigProperty(name="userId")
Provider<String> userId;
```

Config

Inyección global

```
@Inject Config config;
```

OpenAPI - Aplicación

APIs documentadas en formato estandard

- @APIResponses - Respuestas multiples de una API
- @APIResponse - Respuesta unica de una API
- @Content - Esquema y ejemplo
- @Schema - I/O data types
- @Operation - Describe la operación
- @Parameter - Describe el parámetro de una operación

OpenAPI - Aplicación

APIs documentadas en formato estandard

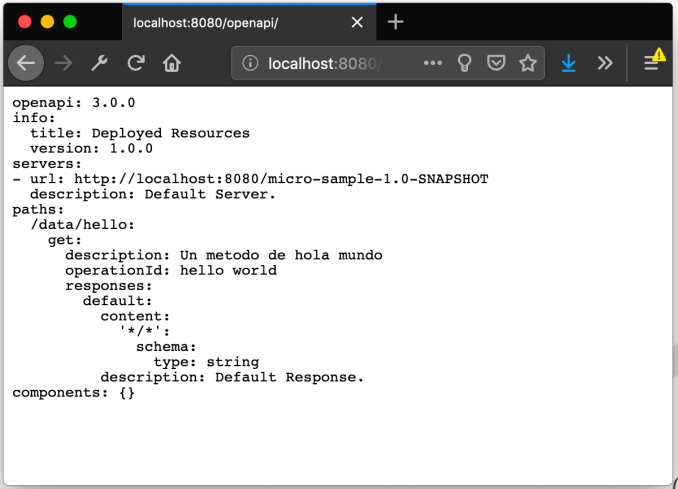
```
@ApplicationPath("/api")
@OpenAPIDefinition(info = @Info(
        title = "Example␣application",
        version = "1.0.0",
        contact = @Contact(
        name = "Victor␣Orozoc",
        email = "vorozco@nabenik.com",
        url = "http://vorozco.com")
        ),
        servers = {
                @Server(url = "/example",
                description = "localhost")
        }
)
public class ApplicationConfig extends Application {
```

ACADEMIK

APIs documentadas en formato estandard

```
@GET @Path("/{key}")
@Operation(description = "Get␣the␣value␣for␣this␣key")
@APIResponses({
        @APIResponse(responseCode = "200",
        description = "Successful,␣returning␣the␣value")
})
@Produces(MediaType.TEXT_PLAIN)
public Response getConfigValue(@PathParam("key") String key)
```

OpenAPI

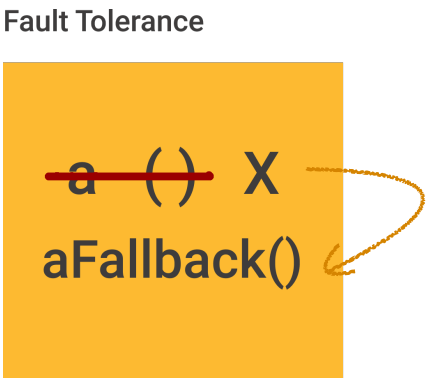


ACADEMIK

OpenAPI - Swagger UI

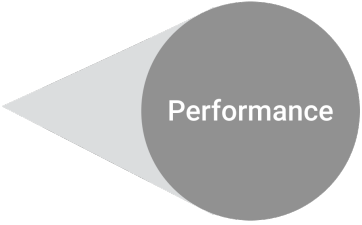
APIs documentadas en formato estandard

```
<dependency>
        <groupId>org.microprofile-ext.openapi-ext</groupId>
        <artifactId>swagger-ui</artifactId>
        <version>1.0.2</version>
</dependency>
```



Metrics

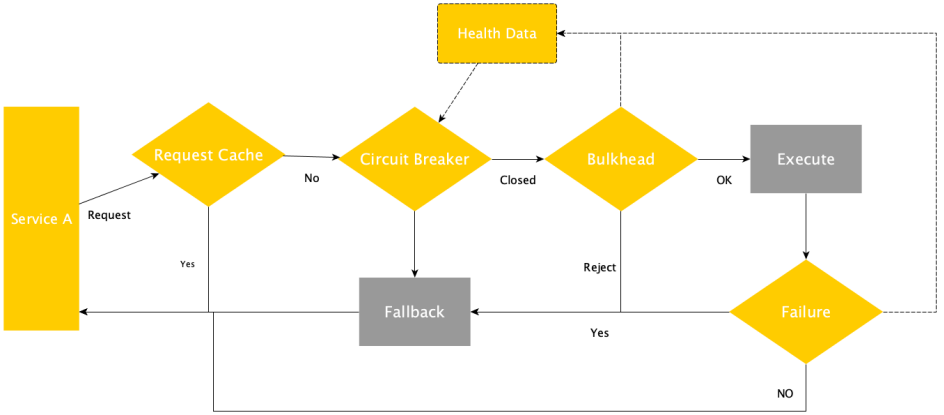
Metrics



ACADEMIK

Fault Tolerance + Metrics

- *Fault Tolerance* depende de la existencia de metricas, las metricas se exponen mediante *Metrics*



Fault tolerance

Reglas de evaluación y alternativas

- Circuit Breaker
- Bulkhead
- Retry
- Timeout
- Fallback

ACADEMIK

Fault tolerance - Retry

```
@Retry(delay = 400, maxDuration= 3200, jitter= 400, maxRetries = 10)
public Connection serviceA() {
        ...
}

@Retry(retryOn = {IOException.class})
public void serviceB() {
        ...
}
```

Fault tolerance - CircuitBreaker

```
@CircuitBreaker(successThreshold = 10,
        requestVolumeThreshold = 4,
        failureRatio=0.75,
        delay = 1000)
public Connection serviceA() {
        Connection conn = null;
        conn = connectionService();
        return conn;
}
```

Fault tolerance - Bulkhead

```
@Bulkhead(5)
public Connection serviceA() {
        Connection conn = null;
        conn = connectionService();
        return conn;
}

@Asynchronous
@Bulkhead(value = 5, waitingTaskQueue = 8)
public Future<Connection> serviceA() {
        Connection conn = null;
        conn = connectionService();
        return CompletableFuture.completedFuture(conn);
}
```

Fault tolerance - Fallback, Timeout

```
@GET
@Path("/{id:[a-z]*[0-9][0-9]*}")
@Fallback(fallbackMethod = "findByIdFallBack")
@Timeout(TIMEOUT)
public Response findById(@PathParam("id")
final String imdbId) {
...
}

public Response findByIdFallBack(@PathParam("id")
final String imdbId) {
...
}
```

Fault tolerance - Fallback Handler, Timeout

```
@GET
@Path("/{id:[a-z]*[0-9][0-9]*}")
@Fallback(MovieFindAllFallbackHandler.class)
@Timeout(TIMEOUT)
public Response findById(@PathParam("id")
final String imdbId) {
...
}

public class MovieFindAllFallbackHandler
        implements FallbackHandler<List> {
        @Override
        public List handle(final ExecutionContext context) {
                return Stream.of("Star␣Wars",
                "The␣Matrix", "Cantinflas").collect(toList());
        }
}
```

Métricas

- JSON or OpenMetrics (Prometheus)
- Vendor
- Base
- Application

¿Cuales?

- Counted
- Gauge
- Metered
- Timed
- Histogram

ACADEMIK

Metrics - Counted

```
@Inject
@Metric
Counter failedQueries;

@GET
@Path("/{id:[a-z]*[0-9][0-9]*}")
@Fallback(fallbackMethod = "findByIdFallBack")
@Timeout(TIMEOUT)
public Response findById(@PathParam("id")
final String imdbId) {
...
}

public Response findByIdFallBack(@PathParam("id")
final String imdbId) {
        failedQueries.inc();
}
```

Metrics - Gauge

Inc-dec en tiempo real

```
@Gauge(unit = .ExternalDatabases",name = "movieDatabases", absolute = true)
public long getDatabases() {
        return 99; //Any value
}

/metrics/application/movieDatabases
```

ACADEMIK

Metrics - Metered

Events rate

```
@Metered(name = "moviesRetrieved",
        unit = MetricUnits.MINUTES,
        description = "Metrics␣to␣monitor␣movies",
        absolute = true)
public Response findExpandedById(
        @PathParam("id") final Long id)
```

```
/metrics/application/movieDatabases
```

Metrics- Timed

Desempeño y retraso

```
@Timed(name = "moviesDelay",
        description = "Time␣to␣retrieve␣a␣movie",
        unit = MetricUnits.MINUTES,
        absolute = true)
public Response findExpandedById(
        @PathParam("id") final Long id)

/metrics/application/moviesDelay
```

ACADEMIK

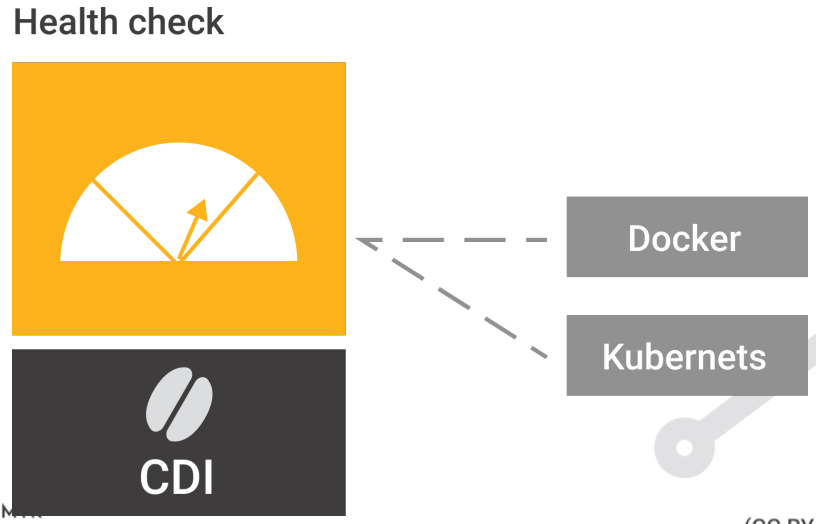
Metrics - Histogram

Distribuciones

```
@Inject
MetricRegistry registry;

@POST
@Path("/add/{attendees}")
public Response addAttendees(
        @PathParam("attendees") Long attendees) {
        Metadata metadata =
                new Metadata("matrix␣attendees",
                        MetricType.HISTOGRAM);
        Histogram histogram =
                registry.histogram(metadata);
        histogram.update(attendees);
        return Response.ok().build();
}
```

Health Check



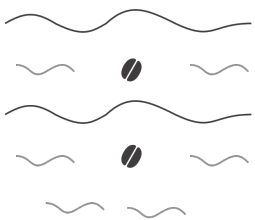
Health Check

¿Estas vivo?

```
@Override
public HealthCheckResponse call() {
        return HealthCheckResponse.named("TaVivoAinda")
                .withData("key1", "val1")
                .withData("key2", "val2")
                .up()
                .build();

}
```

JWT



@Inject
Principal

@Inject
Realm

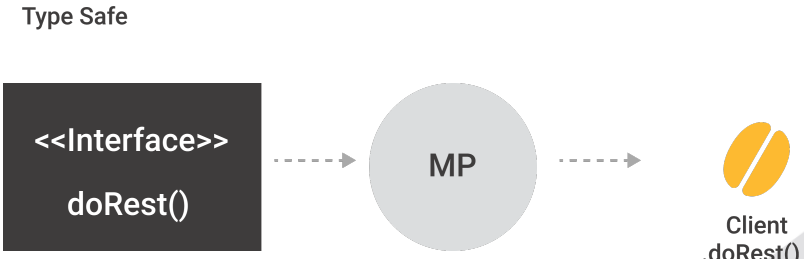
ACADEMIK

```
@LoginConfig(authMethod = "MP-JWT")
public class ApplicationConfig extends Application {
}

@Inject
private JsonWebToken jwtPrincipal;

@Inject
@Claim(.email")
private String email;
```

TypeSafe



ACADEMIK

TypeSafe

```
@Path("/playlist")
@Consumes("application/json")
public interface MusicPlaylistService {

        @GET
        List<String> getPlaylistNames();


        @PUT
        @Path("/{playlistName}")
        long updatePlayList(@PathParam("playlistName")
                String name,
                List<Song> playlist)
                throws UnknownPlaylistException;
}
```

ACADEMIK

Demo

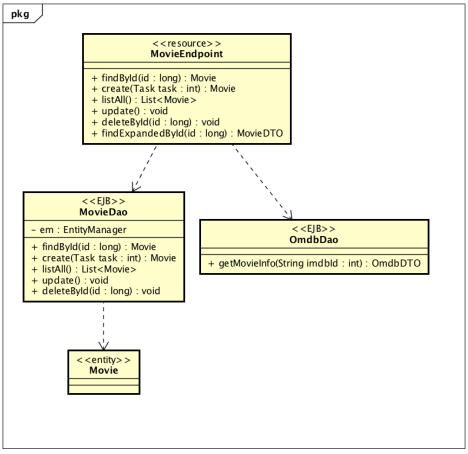
ACADEMIK

Java 11, JAX-RS, CDI, EJB, MicroProfile

https://github.com/tuxtor/payara-demo
https://github.com/tuxtor/omdb-demo

Stacks tradicionales

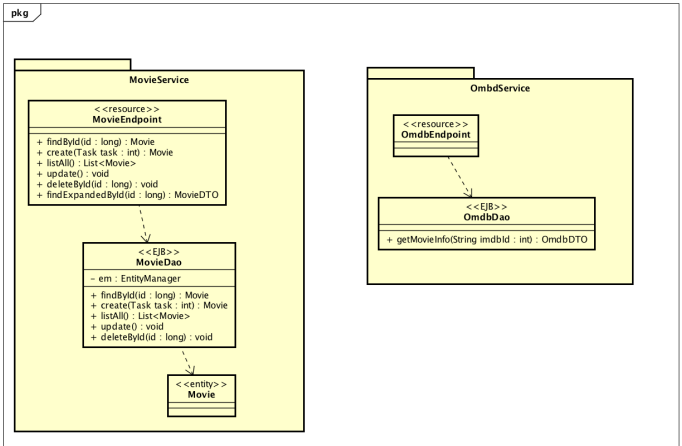
- EJB
- **JTA**
- JAX-RS
- CDI



ACADEMIK

EE + MicroProfile - Demo

MicroProfile: JAX-RS, CDI, Config, Fault Tolerance, Metrics
Payara Micro: EJB, JTA
Fatores externos: Location, Deployment, Orchestation, Balancing, Consistency



ACADEMIK

12 factores cloud native (Heroku)

Microprofile

- Config
- Backing service
- Disposability

Cloud

- Codebase (Git-Flow)
- Dependencies (Maven)
- Build, Release, Run
- Processes (Pipelines)
- Port binding
- Concurrency (Docker - k8s)
- Dev / Prod parity
- Logs
- Admin process

ACADEMIK

ACADEMIK

Escríbenos a cursos@academik.io

www.academik.io