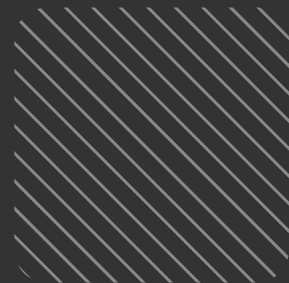


03-03-2021

Meli Toolkits



“Set de **librerías** de desarrollo **colaborativo** destinadas a consumir los servicios de la plataforma”.

IT BOARDING

BOOTCAMP

Índice



01 Meli Toolkits
Initiative

02 RestClient

03 Timeouts, Retries &
Error Handling

04 Trazabilidad

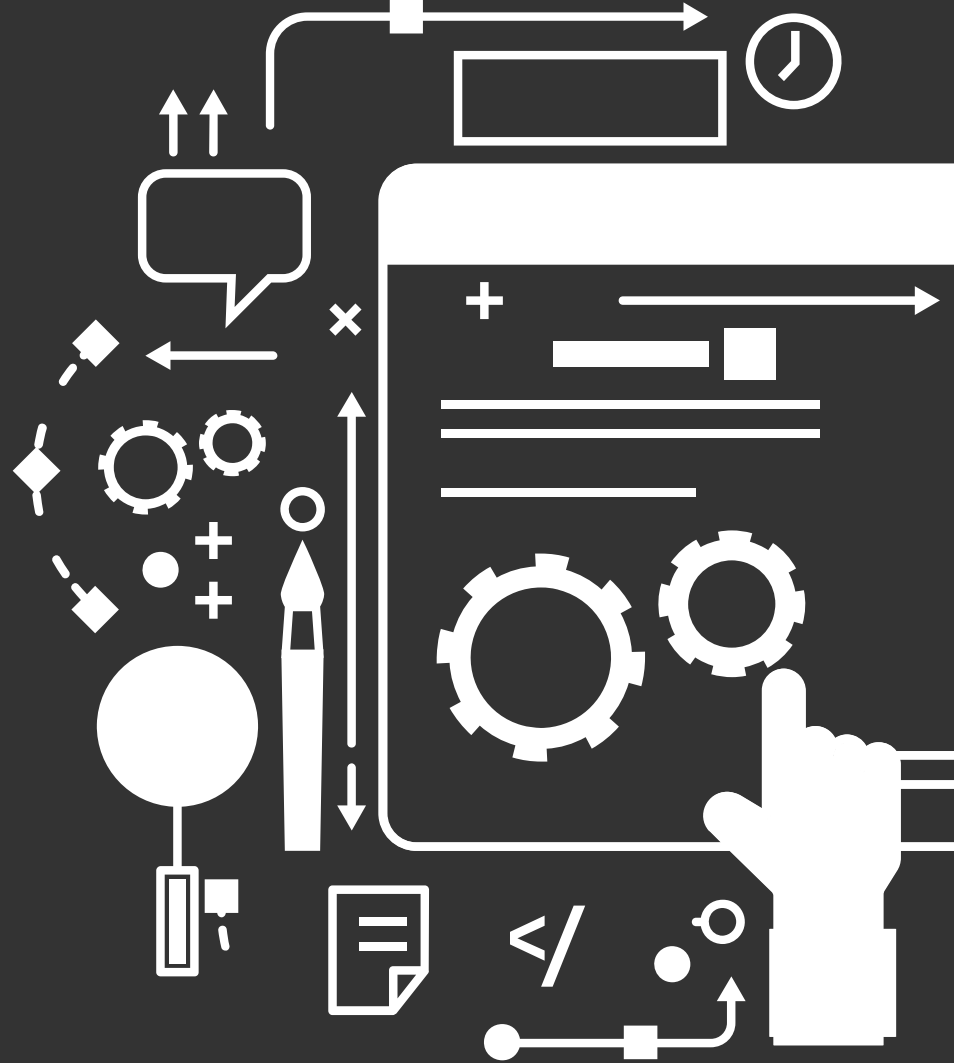
IT BOARDING

BOOTCAMP

Meli Toolkits

IT BOARDING

BOOTCAMP



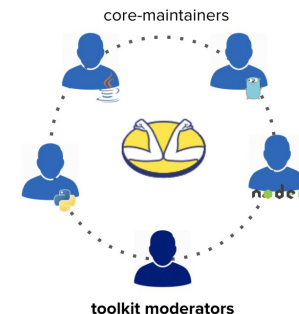
Qué conforman los meli-toolkits?

Son el conjunto de librerías que en cada stack nos facilitan consumir los **Servicios de Fury** e integrarnos con los **Microservicios** del ecosistema **Meli**.

- **Fury Services Clients**
 - Kvs, DS, Audits, Cache, ObjectStorage
 - BigQueue, Lock, Session, Secrets, etc
- **MicroServices & Resilience**
 - Rest Client, Rate Limiter, Circuit Breaker
- **Security**
 - mlauth: Oauth access token, API Keys
 - ssid resolution (for frontends)
- **Observability**
 - Datadog Telemetry, logging, Traceability - Platform Headers Forwarding
- **Common Purpose Tools**, i18n

Inner Source?



Prácticas tomadas de la cultura Open Source aplicada al mantenimiento y evolución del software interno.



- **Open Collaboration & Communication**

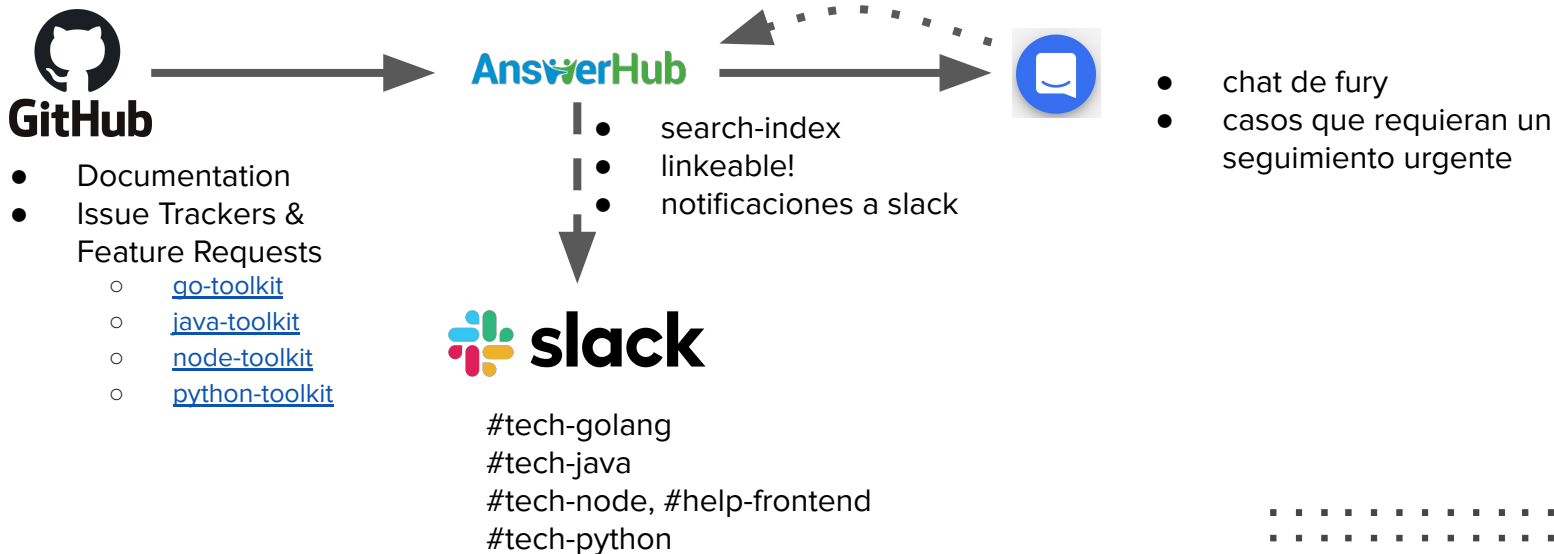
- **Documentation**
- **Issue Tracking & Feature Requests. Pull Requests**
 - [go-toolkit](#)
 - [java-toolkit](#)
 - [node-toolkit](#)
 - [python-toolkit](#)

- **Quality Assurance**

-  **core-maintainers:** Colaboradores activos pertenecientes a cualquier BU, que velan por la mantenibilidad de alguno de los toolkits y la **evolución a largo plazo**.
-  **toolkits-moderator:** Coordinan los esfuerzos para que la evolución de los toolkits de cada una de las tecnologías soportadas sea pareja. A su vez gestionan el roadmap de las mismas.



Pipeline Support



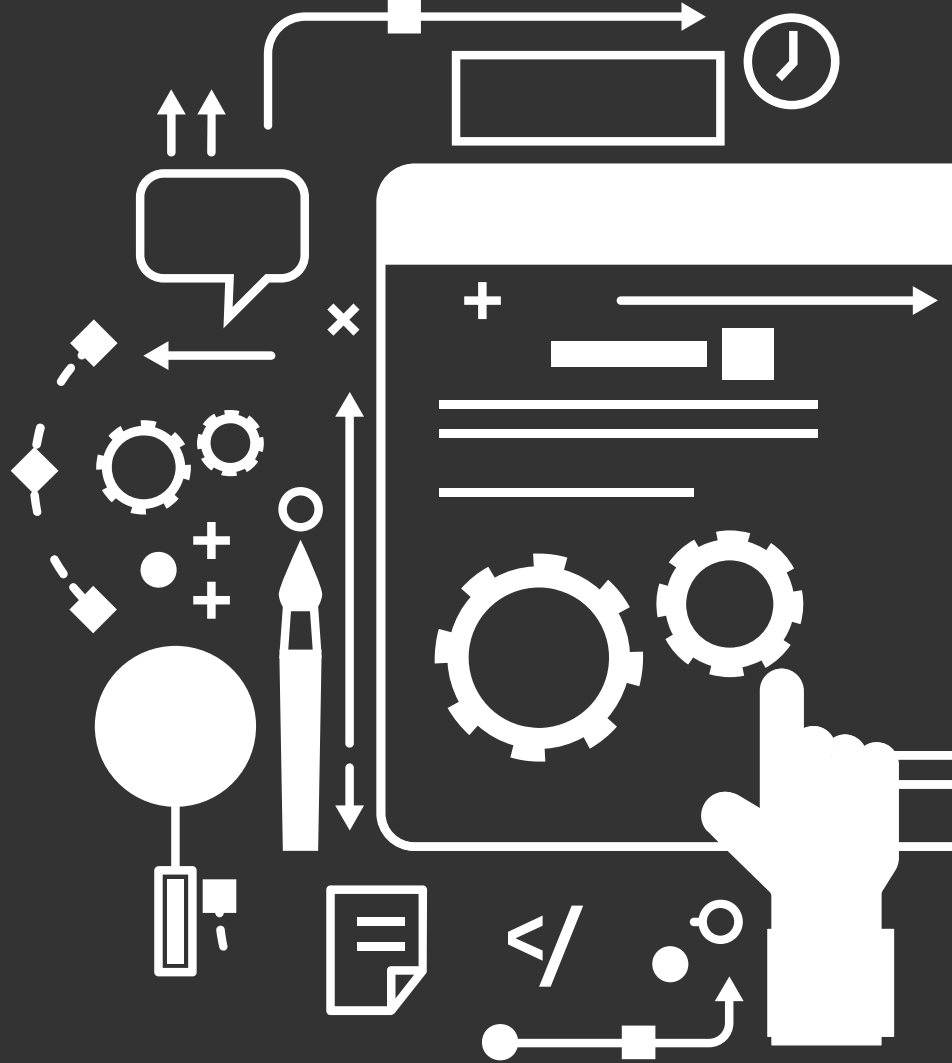
IT BOARDING

BOOTCAMP

HTTP RestClient

IT BOARDING

BOOTCAMP



Cross SDK

RestClient

2021



RestClient

Cliente **HTTP** adaptado para consumir **Rest APIs**

- **Responsabilidades**
 - Manejo de timeouts, retries
 - Marshalling de json
 - Reutilización de conexiones HTTP
 - Observabilidad y métricas
 - New Relic
 - Datadog
 - Trazabilidad

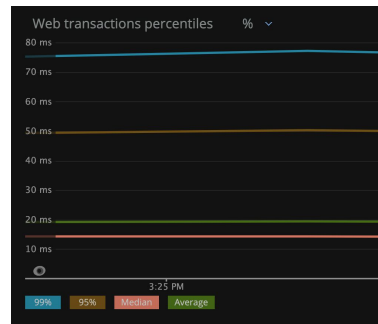


IT BOARDING

BOOTCAMP

// Timeouts

Para instanciar un cliente, debemos configurarlo dependiendo a qué API nos conectemos



- **Response Time**
 - Percentiles (**P95, P99**)
 - No Average ni Max
- **Criticidad**
 - Seremos tan lentos como nuestra dependencia más lenta
 - El timeout no debería estar cerca o por debajo del Average
- **Contention Scenarios**
 - Si un request demora más de x tiempo, podemos entender que nunca volverá.



// Retries

Para instanciar un cliente, debemos configurarlo dependiendo a qué API nos conectemos

- **Default Retriable Methods**

- GET
- HEAD
- OPTIONS

- **Retriable Status Codes**

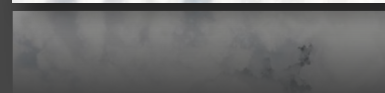
- 5xx

- **Strategies**

- Simple Strategy
- Exponential Backoff Strategy

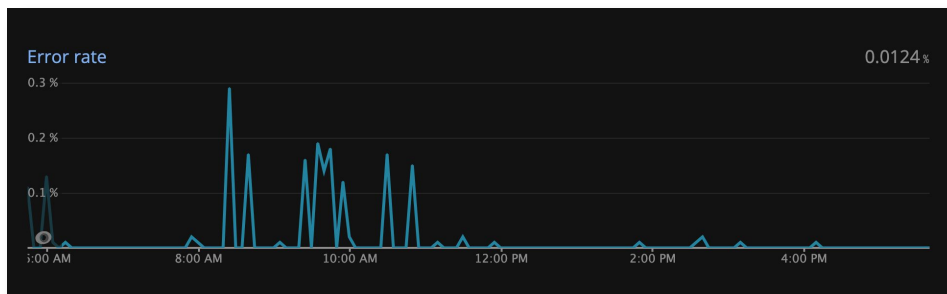
- **Contention Scenarios**

- Los reintentos no sólo hace que esperemos más tiempo, sino que pueden generar más carga hacia nuestra dependencia



// Error Handling

Los microservicios fallan!



- **Criticidad**

- SLA: Qué tan crítico es que un request falle? *Era un intento de compra? Era una búsqueda?*
- Seremos tan resilientes como nuestra dependencia menos resiliente!
- Reintentos
 - Pueden salvar los errores de transporte (*Timeouts*) como los del microservicio que consultamos (**5xx**).
 - Si recibimos **429**, considerar que un reintento seguramente falle.

- **Trade off**

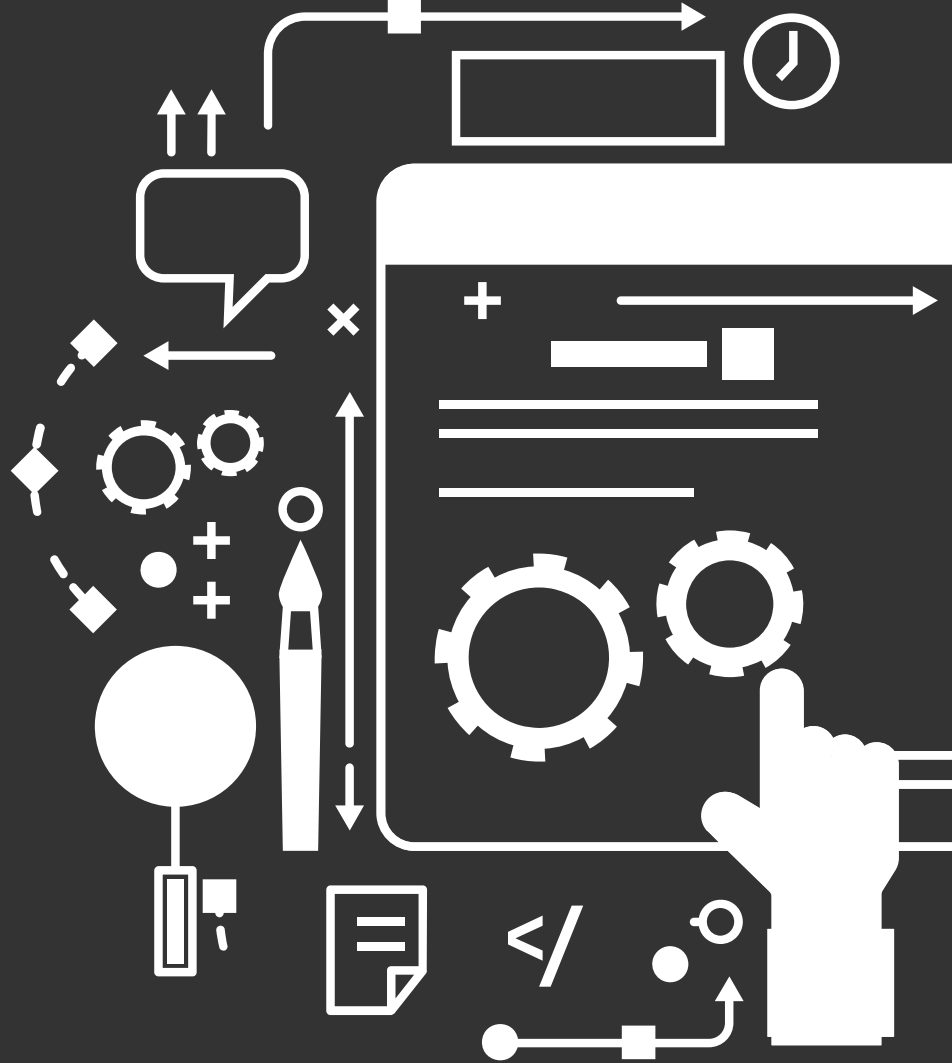
- Más tiempo espero, más cpu consumo.
 - Los request siguen llegando! Los retries generan una carga exponencial a nuestras dependencias.
- Es mejor que un request falle “felizmente” que generar contención y afectar al 100% del tráfico que recibamos.
 - Circuit Breakers

Trazabilidad

//Reenvío Dinámico de Headers

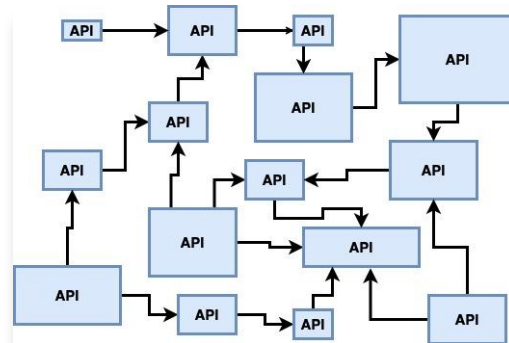
IT BOARDING

BOOTCAMP





Los microservicios fallan!



- Implementación One Shot por cada app
- Features as a Service

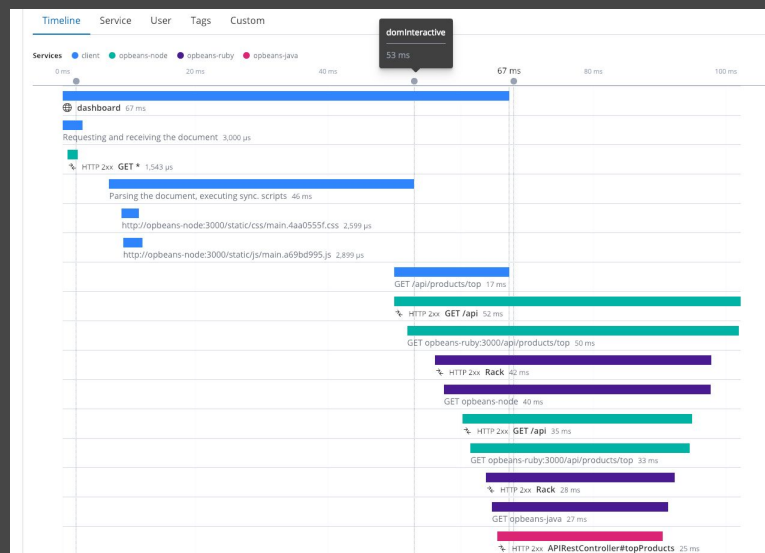
- Autenticación y Autorización
- Control Dinámico de Tráfico
- Distributed Tracing

○ Observabilidad

- Reconstrucción de la cadena de llamadas
- Diagnóstico y Forense

IT BOARDING

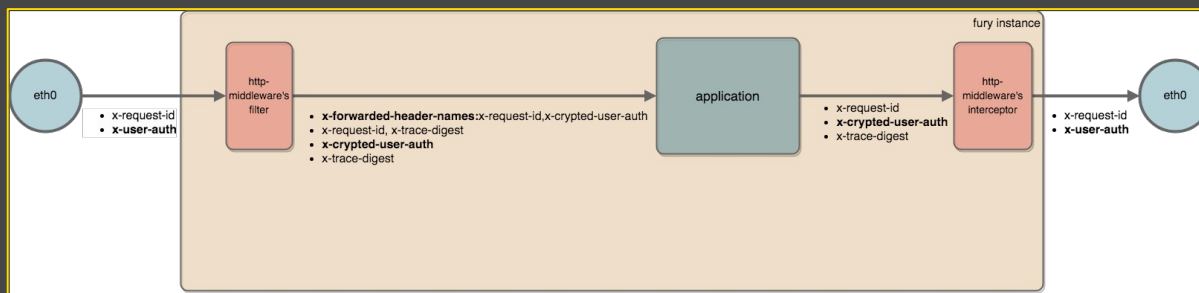
BOOTCAMP





Protocolo

X-Forwarded-Header-Names: header csv que recibe la aplicación con los nombres de los headers que las sdk deben forwardear (best effort, si vienen, la sdk los transporta al buildear el context basándose en los headers recibidos) value ej: **"X-Request-Id,X-D2ID,X-Detached-Id"**.





MeliContext

Nos abstrae del protocolo:

- Lo construimos con los headers recibidos en **cada request**
- Tenemos que **hacerlo llegar** hasta las llamadas del restclient.

```
func controller(request,response)
  ctx = buildMeliContext(request)
  service.exec(ctx, ids)
```

```
public class Service{
  public void exec(MeliContext ctx, List<Long>
  ids){
    serviceDAO.exec(ctx, ids, "blah");
  }
}
```

```
public class ServiceDAO{
  public void exec(MeliContext ctx, List<Long> ids,
  String otherParam){
    meliRestClient.get(uri, headers, ctx);
  }
}
```





MeliContext

Java [meli-restclient](#)

- `meliContext` = `MeliContextBuilder.build`(`HttpServletRequest request`)
- `meliContext` = `MeliContextBuilder.build`(`Map<String, String> headers`)
- `response` = `MeliRestClient.get`("http://internal.mercadolibre.com/items/MLA123", `meliContext`);

- **Construye** un **contexto** en base a los **headers** del protocolo de trazabilidad (**X-Forwarded-Header-Names** y otros).
 - Sus valores son distintos por cada request recibido!
- Siempre que nuestra API reciba un request, debemos **propagar** dicho context hasta los restclient que utilicemos (ya sea de manera sync o async).
 - **WebServices** (backend o frontend)
 - **BigQueue Consumer** Scopes
 - **Fury Job** Scopes





MeliContext

Java [meli-restclient](#)

- `meliContext` = `MeliContextBuilder.buildFlowStarterContext()`
- `response` = `MeliRestClient.get("http://internal.mercadolibre.com/items/MLA123", meliContext);`

- **Construye un contexto** para iniciar un request sin trazabilidad
- Se creó para los escenarios donde una API tiene que hacer un request planificado periódicamente, y no es iniciado por un request entrante.
 - Última opción. Evitar su uso a toda costa! Para flujos que arrancan sin un estímulo externo (tu api hace llamadas sin recibir un request). Ejemplo: Scheduled Tasks, Refresh de caches async.
 - No utilizar en:
 - **frontends**. Los frontends reciben requests e información de traza que deben propagar.
 - **fury jobs**. Se reciben headers de trazabilidad en cada ejecución.
 - **bigqueue** consumers. El servicio de consumers es de tipo push, recibiendo requests con información contextual
 - Los flujos marcados como **flow starter** son catalogados aparte y tendrán más **fricciones y limitaciones**.



// SLA Actual de Trazabilidad en Fury



Alcanza todo el tráfico que generes hacia
`internal.mercado[libre|pago|shops].com`



Actualmente no es necesario hacia los
servicios (*KVS, DS, BigQ, etc*), ni hacia fuera
del cloud de fury (*integradores*)



Gracias.

Alejandro Perez Albiero

IT BOARDING

BOOTCAMP

