Desafío MercadoLibre
Presentación

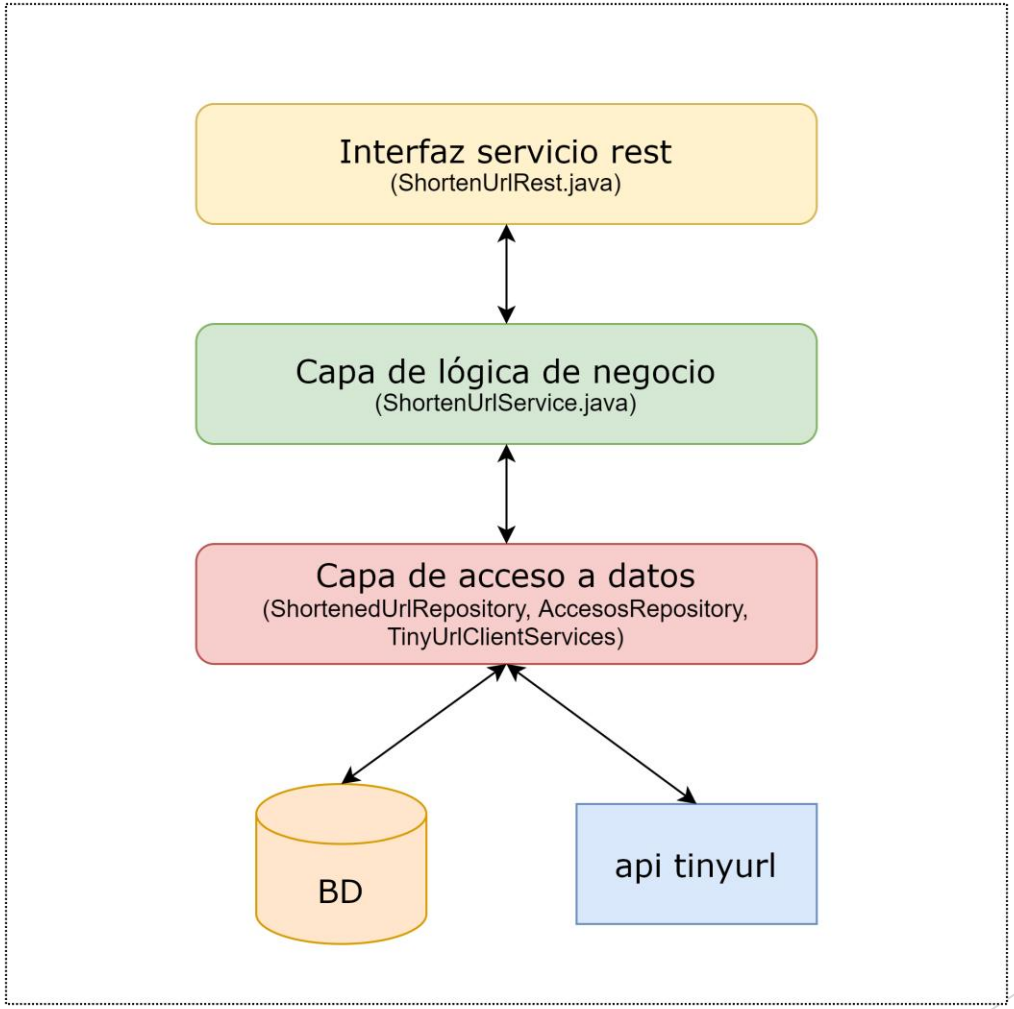
Creado por Manuel A. Ibañez

Tecnología usada

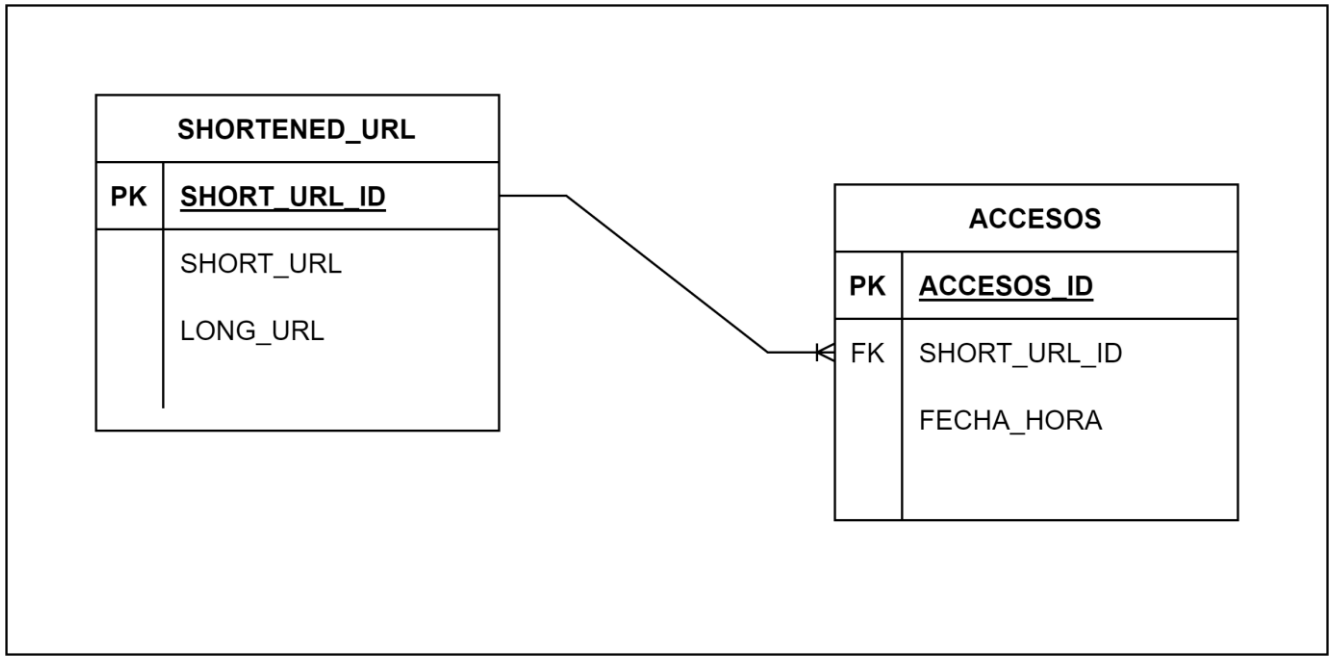
- Java
- Framework quarkus
- Base de datos H2 en memoria
- generador de urls api de tinyurl

Arquietectura servicio

`shorten-url-rest`



Modelo de datos



Método /create

Llamada a la api tinyurl.

```
try {
      CreateTinyUrlRequest request = new CreateTinyUrlRequest();
      request.setDomain(domain);
      request.setUrl(longUrl);

      response = client.create(request);
} catch (Exception e) {
      LOG.error(e);
      return new ResponseDTO(e.getMessage(), null);
}
```

Persiste la url corta y url larga en la tabla `shortened_url`.

```
try {
      String[] split = longUrl.split("//");
      String shortUrl = split[0] + "//" + domain + "/" +
      response.getData().getAlias();

      shortenedUrl = new ShortenedUrl();
      shortenedUrl.setLongUrl(longUrl);
      shortenedUrl.setShortUrl(shortUrl);

      saveShortenedUrl(shortenedUrl);

} catch (Exception e) {
      LOG.error(e);
      return new ResponseDTO(e.getMessage(), null);
}
```

Método /get-short-url

De la tabla shortened_url se obtiene una url corta a partir de una url larga.

```
try {
	ShortenedUrl shortenedUrl = dao.getByLongUrl(longUrl);

	if (shortenedUrl == null) {
		throw new MeliServiceException("No se encuentra short url
		para : " + longUrl);
	} else {
		return new ResponseDTO(null, shortenedUrl.getShortUrl());
	}
} catch (Exception e) {
	LOG.error(e);
	return new ResponseDTO(e.getMessage(), null);
}
```

Método /get-long-url

De la tabla shortened_url se obtiene una url larga a partir de una url corta.

```
try {
	ShortenedUrl shortenedUrl = dao.getByShortUrl(shortUrl);

	if (shortenedUrl == null) {
		throw new MeliServiceException("No se encuentra long url
		para : " + shortUrl);
	} else {
		return new ResponseDTO(null, shortenedUrl.getLongUrl());
	}
} catch (Exception e) {
	LOG.error(e);
	return new ResponseDTO(e.getMessage(), null);
}
```

Método /get-long-url

Interceptor que se llama al final del método para registrar la llamada en la tabla accesos.

La persistencia se hace en un thread aparte para no penalizar la llamada.

```
@Logged
@Priority(2020)
@Interceptor
public class LoggingInterceptor {

	@Inject
	ShortenUrlService shortenUrlService;

	@AroundInvoke
	Object logInvocation(InvocationContext context) throws Exception {

		Object ret = context.proceed();

		Thread t1 = new Thread(new Runnable() {
		public void run() {
			shortenUrlService.log((String) context.getParameters()[0]);
		}
		});
		t1.start();

		return ret;
	}
}
```

Método /delete

Elimina el registro de la tabla shortened_url que corresponde a una url corta.

Además borra los registros de la tabla accesos que corresponden a la url corta eliminada.

```
try {
	ShortenedUrl entity = dao.getByShortUrlForUpdate(shortUrl);

	if (entity==null) {
		throw new MeliServiceException("No existe la short url que
		desea borrar : " + shortUrl);
	} else {
		dao.delete(entity);
		return new ResponseDTO(null, shortUrl);
	}
} catch (Exception e) {
	LOG.error(e);
	return new ResponseDTO(e.getMessage(), null);
}
```

Muchas gracias