**3 MULTIPLE MATCHES WIREMOCK**

**1- CREO MI POJO DE RESPONSE DE ESA API DE FRAUDCHECK**

package com**.**booking**.**gateway**;**

**import** com**.**fasterxml**.**jackson**.**annotation**.**JsonCreator**;**

**import** com**.**fasterxml**.**jackson**.**annotation**.**JsonProperty**;**

public class **PayBuddyFraudCheckResponse** **{**

private final **boolean** blacklisted**;**

**@JsonCreator**

public **PayBuddyFraudCheckResponse(@JsonProperty(**"blacklisted"**)** final boolean blacklisted**)** **{**

**this.**blacklisted **=** blacklisted**;**

**}**

public boolean **isBlacklisted()** **{**

**return** blacklisted**;**

**}**

**}**

**===**

**2- IMPLEMENTO EL METODO DE CHECKEO EN MI SERVICE**

package com**.**booking**.**gateway**;**

**import** com**.**booking**.**util**.**PayBuddyRestTemplate**;**

**import** java**.**math**.**BigDecimal**;**

**import** java**.**time**.**LocalDate**;**

public class **PayBuddyGateway** **{**

private final PayBuddyRestTemplate restTemplate **=** **new** PayBuddyRestTemplate**();**

private final String payBuddyBaseUrl**;**

public **PayBuddyGateway(**final String payBuddyHost**,** final int payBuddyPort**)** **{**

**this.**payBuddyBaseUrl **=** String**.**format**(**"http://%s:%s/"**,** payBuddyHost**,** payBuddyPort**);**

**}**

public PayBuddyPaymentResponse makePayment**(**String creditCardNumber**,** LocalDate creditCardExpiry**,**

BigDecimal amount**)** **{**

final PayBuddyPaymentRequest request **=** **new** **PayBuddyPaymentRequest(**

creditCardNumber**,** creditCardExpiry**,** amount**);**

**return** restTemplate**.**postForObject**(**payBuddyBaseUrl **+** "/payments"**,** request**,** PayBuddyPaymentResponse**.**class**);**

**}**

public PayBuddyFraudCheckResponse fraudCheck**(**String creditCardNumber**)** **{**

**return** restTemplate**.**getForObject**(**payBuddyBaseUrl **+** "/blacklisted-cards/" **+** creditCardNumber**,** PayBuddyFraudCheckResponse**.**class**);**

**}**

**}**

**===**

**3- REALIZO EL TEST CON LOS DOS CASOS, HAPPY & UNHAPPY FLOW**

package com**.**booking**.**service**;**

**import** com**.**booking**.**domain**.**BookingPayment**;**

**import** com**.**booking**.**domain**.**CreditCard**;**

**import** com**.**booking**.**gateway**.**PayBuddyGateway**;**

**import** com**.**github**.**tomakehurst**.**wiremock**.**junit**.**WireMockRule**;**

**import** org**.**junit**.**Before**;**

**import** org**.**junit**.**Rule**;**

**import** org**.**junit**.**Test**;**

**import** java**.**math**.**BigDecimal**;**

**import** java**.**time**.**LocalDate**;**

**import** static com**.**booking**.**service**.**BookingResponse**.**BookingResponseStatus**.**SUCCESS**;**

**import** static com**.**booking**.**service**.**BookingResponse**.**BookingResponseStatus**.**SUSPECTED\_FRAUD**;**

**import** static com**.**github**.**tomakehurst**.**wiremock**.**client**.**WireMock**.\*;**

**import** static org**.**assertj**.**core**.**api**.**Assertions**.**assertThat**;**

public class BookingServiceTest **{**

@Rule

public WireMockRule wireMockRule **=** **new** WireMockRule**();**

private BookingService bookingService**;**

@Before

public void setUp**()** **{**

bookingService **=** **new** BookingService**(new** PayBuddyGateway**(**"localhost"**,** wireMockRule**.**port**()));**

**}**

@Test

public void shouldSucceedToPayForBooking**()** **{**

// Given

stubFor**(**post**(**urlPathEqualTo**(**"/payments"**)).**withRequestBody**(**

equalToJson**(**"{" **+**

" \"creditCardNumber\": \"1234-1234-1234-1234\"," **+**

" \"creditCardExpiry\": \"2018-02-01\"," **+**

" \"amount\": 20.55" **+**

"}"**))**

**.**willReturn**(**

okJson**(**"{" **+**

" \"paymentId\": \"2222\"," **+**

" \"paymentResponseStatus\": \"SUCCESS\"" **+**

"}"**)));**

stubFor**(**get**(**urlPathEqualTo**(**"/blacklisted-cards/1234-1234-1234-1234"**))**

**.**willReturn**(**okJson**(**"{" **+**

" \"blacklisted\": \"false\"" **+**

"}"**)));**

// When

final BookingResponse bookingResponse **=** bookingService**.**payForBooking**(**

**new** BookingPayment**(**

"1111"**,**

**new** BigDecimal**(**"20.55"**),**

**new** CreditCard**(**"1234-1234-1234-1234"**,** LocalDate**.**of**(**2018**,** 2**,** 1**))));**

// Then

assertThat**(**bookingResponse**).**isEqualTo**(new** BookingResponse**(**"1111"**,** "2222"**,** SUCCESS**));**

**}**

@Test

public void shouldFailToPayForBookingDueToFraud**()** **{**

// Given

stubFor**(**get**(**urlPathEqualTo**(**"/blacklisted-cards/1234-1234-1234-1234"**))**

**.**willReturn**(**okJson**(**"{" **+**

" \"blacklisted\": \"true\"" **+**

"}"**)));**

// When

final BookingResponse bookingResponse **=** bookingService**.**payForBooking**(**

**new** BookingPayment**(**

"1111"**,**

**new** BigDecimal**(**"20.55"**),**

**new** CreditCard**(**"1234-1234-1234-1234"**,**

LocalDate**.**of**(**2018**,** 2**,** 1**))));**

// Then

assertThat**(**bookingResponse**)**

**.**isEqualTo**(new** BookingResponse**(**

"1111"**,**

**null,**

**SUSPECTED\_FRAUD));**

**}**

**}**

**4- METO EL NUEVO STATUS EN EL BOOKING RESPONSE STATUS!!!**

package com**.**booking**.**service**;**

**import** org**.**apache**.**commons**.**lang3**.**builder**.**EqualsBuilder**;**

**import** org**.**apache**.**commons**.**lang3**.**builder**.**HashCodeBuilder**;**

**import** org**.**apache**.**commons**.**lang3**.**builder**.**ToStringBuilder**;**

public class **BookingResponse** **{**

private final String bookingId**;**

private final String paymentId**;**

private final BookingResponseStatus bookingResponseStatus**;**

public **BookingResponse(**String bookingId**,** String paymentId**,** BookingResponseStatus bookingResponseStatus**)** **{**

**this.**bookingId **=** bookingId**;**

**this.**paymentId **=** paymentId**;**

**this.**bookingResponseStatus **=** bookingResponseStatus**;**

**}**

public enum **BookingResponseStatus** **{**

SUCCESS**,** REJECTED**,** **SUSPECTED\_FRAUD**

**}**

public String **getBookingId()** **{**

**return** bookingId**;**

**}**

public String **getPaymentId()** **{**

**return** paymentId**;**

**}**

public BookingResponseStatus **getBookingResponseStatus()** **{**

**return** bookingResponseStatus**;**

**}**

**@Override**

public boolean **equals(**Object obj**){**

**return** **EqualsBuilder.**reflectionEquals**(this,** obj**);**

**}**

**@Override**

public int **hashCode(){**

**return** **HashCodeBuilder.**reflectionHashCode**(this);**

**}**

**@Override**

public String **toString(){**

**return** **ToStringBuilder.**reflectionToString**(this);**

**}**

**}**

**5- PRIORITY**

package com**.**booking**.**service**;**

**import** com**.**booking**.**domain**.**BookingPayment**;**

**import** com**.**booking**.**domain**.**CreditCard**;**

**import** com**.**booking**.**gateway**.**PayBuddyFraudCheckResponse**;**

**import** com**.**booking**.**gateway**.**PayBuddyGateway**;**

**import** com**.**booking**.**gateway**.**PayBuddyPaymentResponse**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Set**;**

**import** java**.**util**.**UUID**;**

**import** static java**.**util**.**stream**.**Collectors**.**toSet**;**

public class BookingService **{**

private final PayBuddyGateway payBuddyGateway**;**

public BookingService**(**PayBuddyGateway payBuddyGateway**)** **{**

**this.**payBuddyGateway **=** payBuddyGateway**;**

**}**

public BookingResponse payForBooking**(**final BookingPayment bookingPayment**)** **{**

final CreditCard creditCard **=** bookingPayment**.**getCreditCard**();**

final PayBuddyFraudCheckResponse payBuddyFraudCheckResponse **=** payBuddyGateway**.**fraudCheck**(**creditCard**.**getNumber**());**

**if** **(**payBuddyFraudCheckResponse**.**isBlacklisted**())** **{**

**return** **new** BookingResponse**(**bookingPayment**.**getBookingId**(),** BookingResponse**.**BookingResponseStatus**.**SUSPECTED\_FRAUD**);**

**}**

final String paymentId **=** UUID**.**randomUUID**().**toString**();**

final PayBuddyPaymentResponse payBuddyPaymentResponse **=** payBuddyGateway**.**makePayment**(**paymentId**,** creditCard**.**getNumber**(),** creditCard**.**getExpiry**(),** bookingPayment**.**getAmount**());**

**if** **(**payBuddyPaymentResponse**.**getPaymentResponseStatus**()** **==** PayBuddyPaymentResponse**.**PaymentResponseStatus**.**SUCCESS**)** **{**

**return** **new** BookingResponse**(**bookingPayment**.**getBookingId**(),** BookingResponse**.**BookingResponseStatus**.**SUCCESS**);**

**}** **else** **{**

**return** **new** BookingResponse**(**bookingPayment**.**getBookingId**(),** BookingResponse**.**BookingResponseStatus**.**REJECTED**);**

**}**

**}**

public Set**<**BookingResponse**>** payForMultipleBookingsInBatch**(**final BookingPayment **...** bookingPayment**)** **{**

**return** Arrays**.**stream**(**bookingPayment**).**map**(this::**payForBooking**).**collect**(**toSet**());**

**}**

**}**

**===**

package com**.**booking**.**service**;**

public class BookingServiceTest **{**

@Rule

public WireMockRule wireMockRule **=** **new** WireMockRule**();**

private BookingService bookingService**;**

@Before

public void setUp**()** **{**

bookingService **=** **new** BookingService**(new** PayBuddyGateway**(**"localhost"**,** wireMockRule**.**port**()));**

**}**

@Test

public void shouldPayForMultipleBookingsOneOfWhichWillFailOnTheFraudCheck**()** **{**

// Given

stubFor**(**post**(**urlPathEqualTo**(**"/payments"**))**

**.**withRequestBody**(**matchingJsonPath**(**"$.creditCardNumber"**))**

**.**withRequestBody**(**matchingJsonPath**(**"$.creditCardExpiry"**))**

**.**withRequestBody**(**matchingJsonPath**(**"$.amount"**))**

**.**withRequestBody**(**matchingJsonPath**(**"$.paymentId"**))**

**.**willReturn**(**

okJson**(**"{" **+**

" \"paymentResponseStatus\": \"SUCCESS\"" **+**

"}"**)));**

stubFor**(**get**(**urlPathEqualTo**(**"/blacklisted-cards/0000-0000-0000-0000"**))**

**.**atPriority**(**1**)**

**.**willReturn**(**okJson**(**"{" **+**

" \"blacklisted\": \"true\"" **+**

"}"**)));**

stubFor**(**get**(**urlPathMatching**(**"/blacklisted-cards/.\*"**))**

**.**atPriority**(**100**)**

**.**willReturn**(**okJson**(**"{" **+**

" \"blacklisted\": \"false\"" **+**

"}"**)));**

final BookingPayment fraud **=** **new** BookingPayment**(**

"0"**,**

**new** BigDecimal**(**"1.11"**),**

**new** CreditCard**(**"0000-0000-0000-0000"**,**

LocalDate**.**of**(**2011**,** 1**,** 1**))**

**);**

final BookingPayment toSucceed1 **=** **new** BookingPayment**(**

"1"**,**

**new** BigDecimal**(**"1.11"**),**

**new** CreditCard**(**"1111-1111-1111-1111"**,**

LocalDate**.**of**(**2011**,** 1**,** 1**))**

**);**

final BookingPayment toSucceed2 **=** **new** BookingPayment**(**

"2"**,**

**new** BigDecimal**(**"2.22"**),**

**new** CreditCard**(**"2222-2222-2222-2222"**,**

LocalDate**.**of**(**2022**,** 2**,** 2**))**

**);**

// When

final Set**<**BookingResponse**>** bookingResponses **=** bookingService

**.**payForMultipleBookingsInBatch**(**toSucceed1**,** toSucceed2**,** fraud**);**

// Then

assertThat**(**bookingResponses**).**containsExactlyInAnyOrder**(**

**new** BookingResponse**(**"1"**,** SUCCESS**),**

**new** BookingResponse**(**"2"**,** SUCCESS**),**

**new** BookingResponse**(**"0"**,** SUSPECTED\_FRAUD**)**

**);**

**}**

**}**