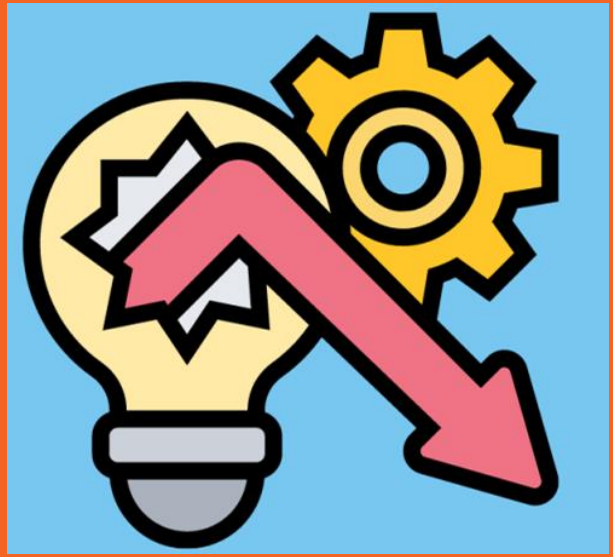
Trabajo Final - Construcción y Pruebas de Software

Hernan Eraso
William Martín Chávez González
Santiago Rojas

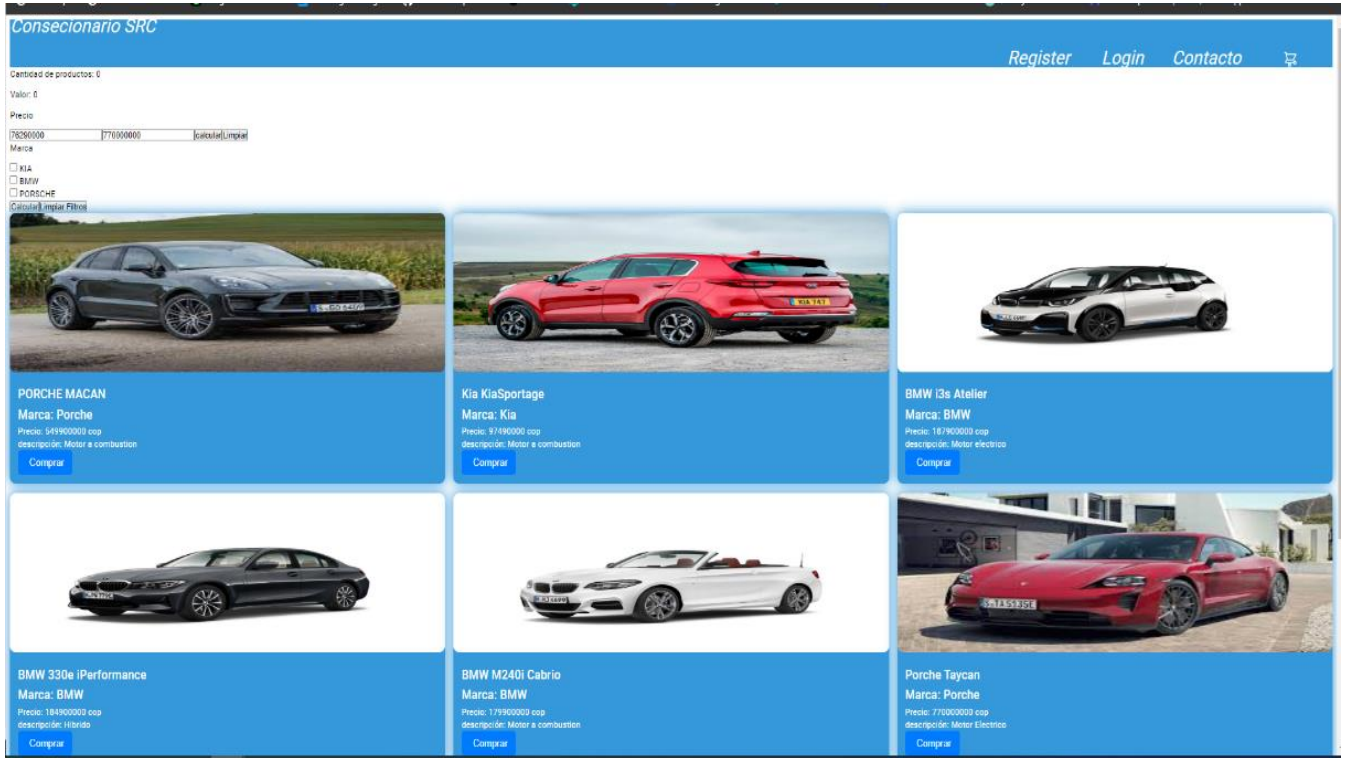
Pontificia Universidad JAVERIANA Cali

[VIGILADA MINEDUCACIÓN Res. 12220 de 2016]



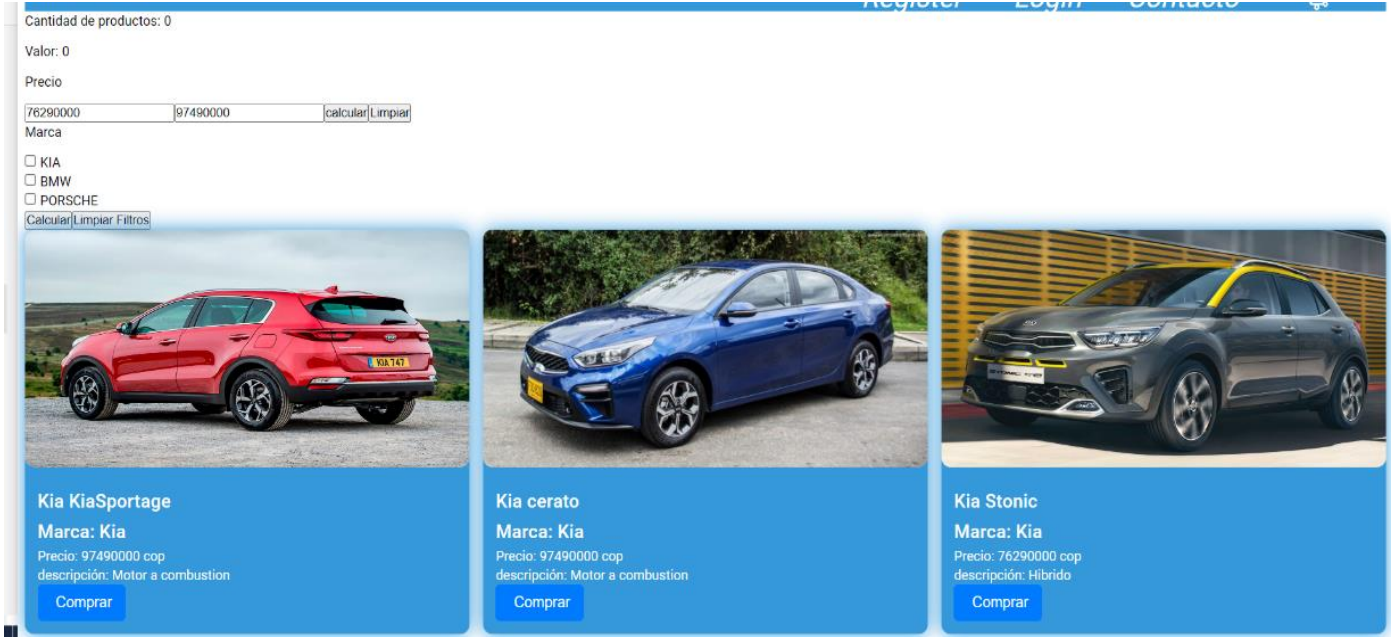
Pruebas Funcionales Front

1. Render de productos



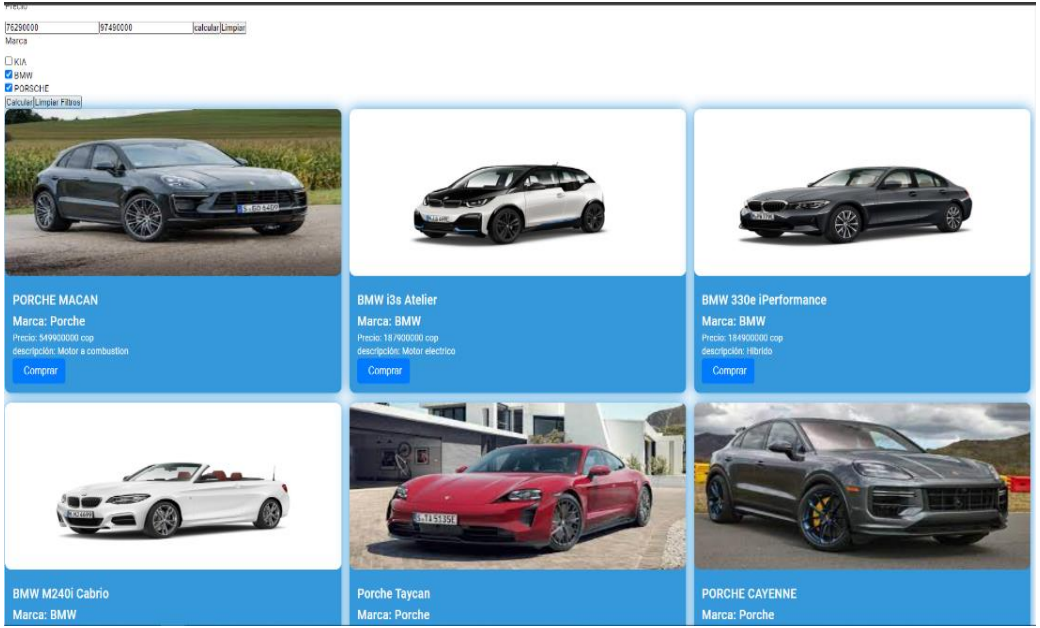
Pruebas Funcionales Front

1. Render de productos por precio



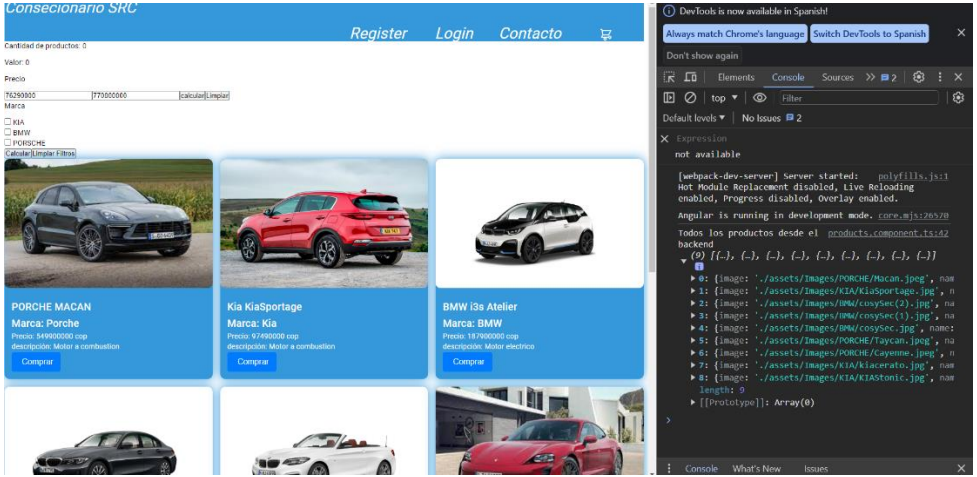
Pruebas Funcionales Front

1. Render de productos por marca



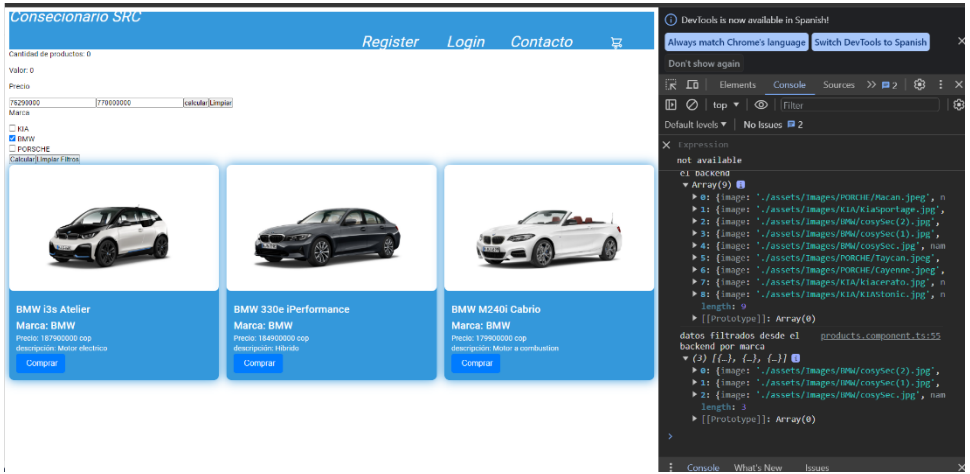
Pruebas Funcionales Front

1. Conexión Backend con Front render de todos los productos
   http://localhost:8080/api/cars



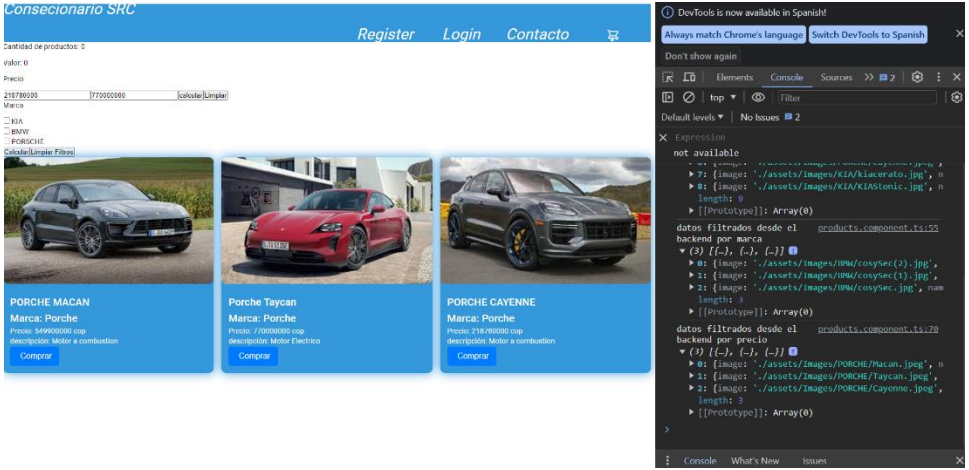
Pruebas Funcionales Front

1. Conexión Backend con Front render de los productos por marca
   http://localhost:8080/api/searchBrands?brands=BMW



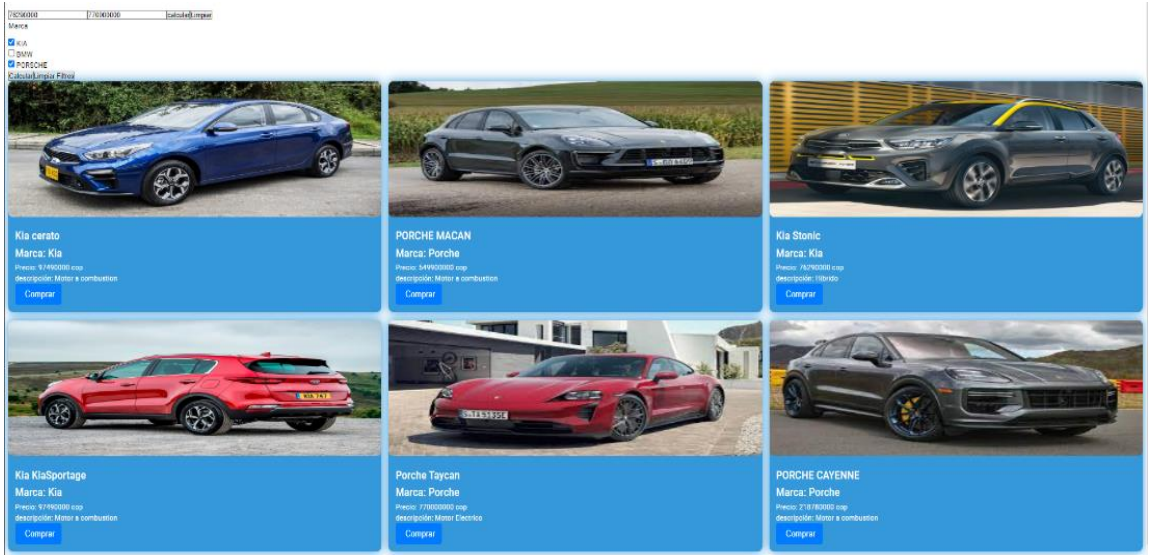
Pruebas Funcionales Front

1. Conexión Backend con Front render de los productos por precio
http://localhost:8080/api/search?initialPrice=218780000&finalPrice=770000000



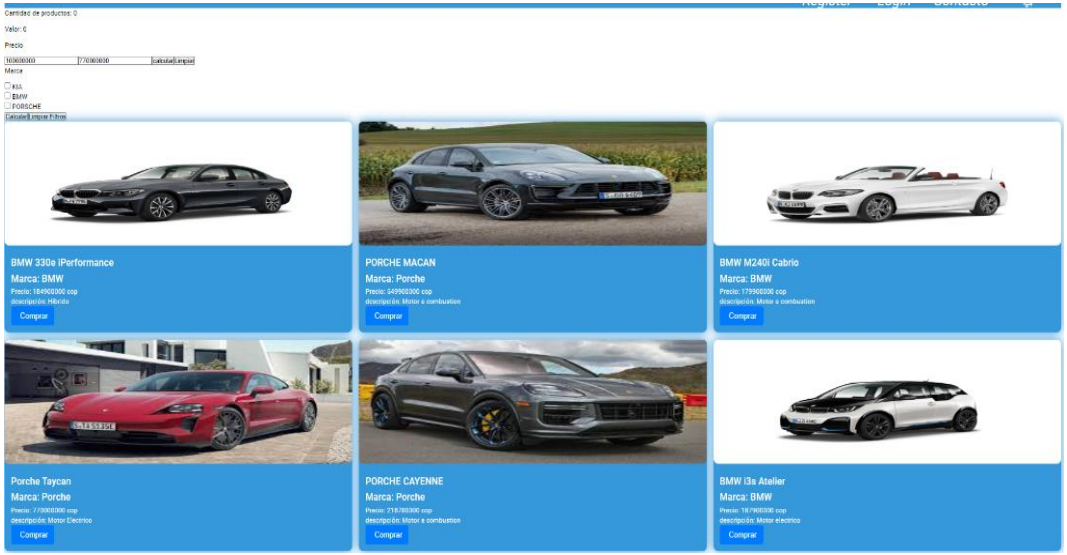
Pruebas E2E

1. Render de información por marcas



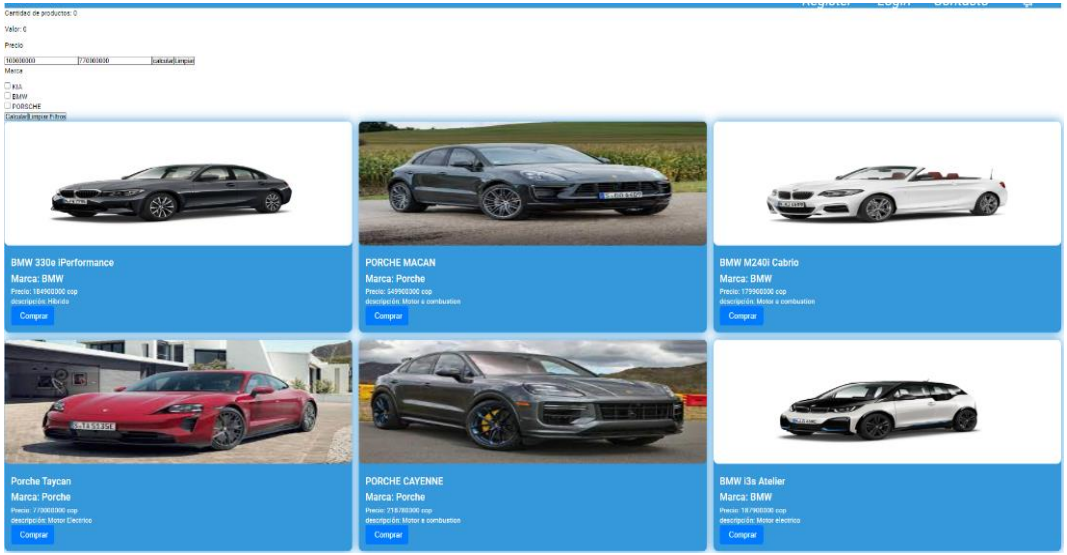
Pruebas E2E

1. Render de información por precio



Pruebas E2E

1. Render de información por precio



Casos de pruebas

	VARIABLE	DE EQUIVALENCIA o CON	REPRESENTANTE	TIPO	P1	P2	P3	P4	P5	P6	P7	P8	P9
CONDICIONES	minPrice	minPrice <0	-1	Inválido	x								x
		minPrice>=0	0	Válido		x	x	x	x		x	x	
		minPrice <maxprice	1999999999	Inválido						x			
		minprice=maxprice	2000000000	Inválido									
		minprice>maxprice	2000000001	Inválido									
	MaxPrice	MaxPrice <= 2000000000	80000000	Válido	x	x	x			x	x		
		MaxPrice > 2000000000	2000000001	Inválido				x					x
		maxPrice < 0	-1	Inválido								x	
		maxPrice = 0	0	Inválido					x				
ACCIONES	Llama a limpiarFiltro() y "entre" en log				x			x				x	x
	Llama a getFilterByPrice() y sin productos								x	x	x		
	Llama a getFilterByPrice()					x	x						

Front Pruebas Unitarias KARMA y JASMINE

1. https://github.com/Tiago9617/-consecionario-mat-pruebas/tree/main/consecionario-mat-pruebas

TS products.component.spec.ts M | TS products.service.spec.ts M × | TS products.service.ts

consecionario-mat-pruebas > src > app > services > TS products.service.spec.ts > fdescribe('ProductsService') callback

```
17        providers: [ProductsService]
18      });
19
20      service = TestBed.inject(ProductsService);
21      httpMock = TestBed.inject(HttpTestingController);
22    });
23
24    afterEach(() => {
25      httpMock.verify(); // Verificar que no hay peticiones pendientes
26    });
27    //Todos los productos
28    it('deberia traer todos los productos ', () => {
29
30 >    const products: Product[] = [···
94      ];
95      service.getAllProducts().subscribe(data => {
96        expect(data).toEqual(products);
97      });
98
99      const req = httpMock.expectOne('/api/cars');
100     expect(req.request.method).toBe('GET');
101     req.flush(products);
102   });
```

```
START:
  ProductsService
    √ no deberia traerme ningun producto 1001 y1000
    √ no deberia traerme ningun producto 0 y-1
    √ deberia traer todos los productos 0y 2000000000
    √ deberia traer todos los productos
    √ no deberia traerme ningun producto  -1 y2000000001
    √ no deberia traer datos para el rango de  -1y 2000000000
    √ no deberia traerme ningun producto 2000 y2000000001
    √ deberia traer todos los productos KIA 76290000 y97490000
    √ no deberia traerme ningun producto 2000000000 y2000000000
    √ no deberia traerme ningun producto 0 y0
  ProductsComponent
    √ should create
LOG: 'Todos los productos desde el backend', []
    √ should load user products on init
LOG: 'datos filtrados desde el backend por marca', [Object{image: './assets/Images/KIA/K
KiaSportage.jpg', name: 'Kia KiaSportage', price: 97490000, describe: 'Motor a combustio
 a combustion', marca: 'Kia'}]
    √ should filter by brand
LOG: 'Todos los productos desde el backend', []
    √ should load all products from service on init
  ProductComponent
    √ should create (skipped)
  DataService
    √ should be created (skipped)
  LoginComponent
    √ should create (skipped)
  NavComponent
    √ should create (skipped)
  ShoppingCartComponent
    √ should create (skipped)
  RegisterComponent
    √ should create (skipped)
  ContactComponent
    √ should create (skipped)
  ImgComponent
    √ should create (skipped)
  AppComponent
    √ should have as title 'consecionario-mat-pruebas' (skipped)
    √ should create the app (skipped)
    √ should render title (skipped)

Finished in 0.17 secs / 0.125 secs @ 08:45:47 GMT-0500 (hora estándar de Colombia)

SUMMARY:
√ 14 tests completed
i 11 tests skipped
```

Backend - Code Coverage

1. el coverage que arroja propriamente JaCoCo:
   se compila, se encuentra en la carpeta target/site/jacoco/index.html (cargarlo en Chrome o Firefox)

localhost:63342/test-automation-java-service/target/site/jacoco/index.html?_ijt=eroo2o1ghk8q4t72h2gp0tjlo7&_ij_r 90%

test-automation-java-service-backend

Sessions

test-automation-java-service-backend

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.test.automation.project.testautomationjavaservicebackend.api.database		55 %		0 %	11	13	1	3	4	6	0	1
com.test.automation.project.testautomationjavaservicebackend.api.model		77 %		100 %	5	15	1	11	5	14	0	1
com.test.automation.project.testautomationjavaservicebackend		37 %		n/a	1	2	2	3	1	2	0	1
com.test.automation.project.testautomationjavaservicebackend.api.exceptions		92 %		50 %	1	6	3	14	0	5	0	2
com.test.automation.project.testautomationjavaservicebackend.api.controllers		92 %		50 %	1	5	0	6	0	4	0	1
com.test.automation.project.testautomationjavaservicebackend.api.model.datastructure		100 %		100 %	0	34	0	56	0	20	0	2
com.test.automation.project.testautomationjavaservicebackend.api.services		100 %		100 %	0	15	0	26	0	7	0	1
com.test.automation.project.testautomationjavaservicebackend.api.config		100 %		n/a	0	4	0	7	0	4	0	2
Total	96 of 694	86 %	16 of 64	75 %	19	94	7	126	10	62	0	11

Created with JaCoCo 0.8.11.202310140853

Backend - Code Coverage

1. Cobertura de la estructura de datos de Árbol Binario.

localhost:63342/test-automation-java-service/target/site/jacoco/com.test.automation.project.testautomationjavaserv

test-automation-java-service-backend > com.test.automation.project.testautomationjavaservicebackend.api.model.datastructure

com.test.automation.project.testautomationjavaservicebackend.api.model.datastructure

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
NodoArbol		100 %		100 %	0	26	0	43	0	14	0	1
ArbolBinario		100 %		100 %	0	8	0	13	0	6	0	1
Total	0 of 197	100 %	0 of 28	100 %	0	34	0	56	0	20	0	2

Backend - Code Coverage

1. Cobertura de la estructura de datos de Árbol Binario.

localhost:63342/test-automation-java-service/target/site/jacoco/com.test.automation.project.testautoma

test-automation-java-service-backend > com.test.automation.project.testautomationjavaservicebackend.api.model.datastructure > ArbolBinario

ArbolBinario

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
agregarElemento(Object)		100 %		100 %	0	2	0	4	0	1
recorrerInorden()		100 %		n/a	0	1	0	2	0	1
estaVacio()		100 %		100 %	0	2	0	1	0	1
ArbolBinario()		100 %		n/a	0	1	0	3	0	1
setRaiz(NodoArbol)		100 %		n/a	0	1	0	2	0	1
getRaiz()		100 %		n/a	0	1	0	1	0	1
Total	0 of 44	100 %	0 of 4	100 %	0	8	0	13	0	6

Backend - Code Coverage

1. Cobertura de la estructura de datos de Árbol Binario.

localhost:63342/test-automation-java-service/target/site/jacoco/com.test.automation.project.testautomationjavase

test-automation-java-service-backend > com.test.automation.project.testautomationjavaservicebackend.api.model.datastructure > NodoArbol

NodoArbol

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
agregarElemento(Object)		100 %		100 %	0	6	0	13	0	1
recorrerInorden(ArrayList)		100 %		100 %	0	3	0	6	0	1
compareTo(NodoArbol)		100 %		100 %	0	3	0	6	0	1
NodoArbol(Object)		100 %		n/a	0	1	0	5	0	1
esHoja()		100 %		100 %	0	3	0	1	0	1
esMenor(NodoArbol)		100 %		100 %	0	2	0	1	0	1
equals(Object)		100 %		n/a	0	1	0	1	0	1
setElemento(Object)		100 %		n/a	0	1	0	2	0	1
setIzq(NodoArbol)		100 %		n/a	0	1	0	2	0	1
setDer(NodoArbol)		100 %		n/a	0	1	0	2	0	1
toString()		100 %		n/a	0	1	0	1	0	1
getElemento()		100 %		n/a	0	1	0	1	0	1
getIzq()		100 %		n/a	0	1	0	1	0	1
getDer()		100 %		n/a	0	1	0	1	0	1
Total	0 of 153	100 %	0 of 24	100 %	0	26	0	43	0	14

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6643289475/job/18050041346

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	49.52%
✔ Number of Lines Covered	52
✘ Number of Lines Missed	53
Total Number of Lines	105

Details:

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/pull/2/checks?check_run_id=18062224277

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	54.29%
✔ Number of Lines Covered	57
✘ Number of Lines Missed	48
Total Number of Lines	105

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6647302127/job/18062460823

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	58.1%
✔ Number of Lines Covered	61
✘ Number of Lines Missed	44
Total Number of Lines	105

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6647340868/job/18062561003

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	60%
✔ Number of Lines Covered	63
✘ Number of Lines Missed	42
Total Number of Lines	105

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6647561943/job/18063186567

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	70.75%
✔ Number of Lines Covered	75
✘ Number of Lines Missed	31
Total Number of Lines	106

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6654376805/job/18082312500

Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	77.57%
✔ Number of Lines Covered	83
✘ Number of Lines Missed	24
Total Number of Lines	107

Backend - Code Coverage

1. https://github.com/willy23martin/test-automation-java-service/actions/runs/6654989575/job/18084247519

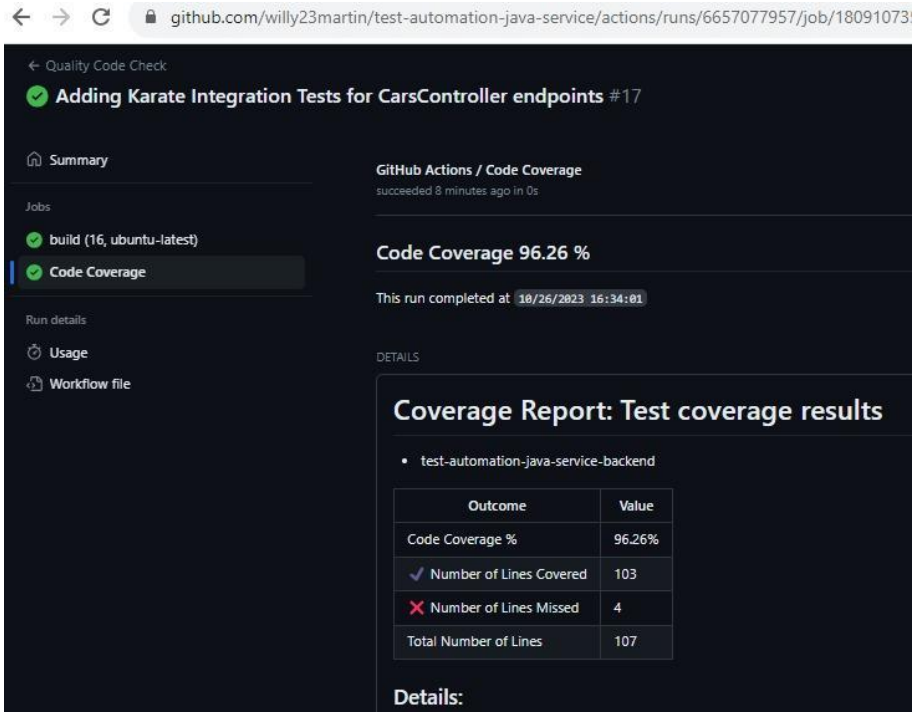
Coverage Report: Test coverage results

- test-automation-java-service-backend

Outcome	Value
Code Coverage %	92.52%
✔ Number of Lines Covered	99
✘ Number of Lines Missed	8
Total Number of Lines	107

Backend - Code Coverage

1. Después de las pruebas de Integración, Sólo restan 4 lineas de código por cubrir que no son realmente relevantes para nuestros casos de prueba.
2. Nuestro Quality Gate sea del 85% como mínimo para nuevas funcionalidades con base en el reporte de JaCoCo.
3. https://github.com/willy23martin/test-automation-java-service/actions/runs/6657077957/job/18091073564



Backend - Pruebas de Integración - Karate

localhost:63342/test-automation-java-service/target/karate-reports/karate-summary.html

Karate Labs

3

0

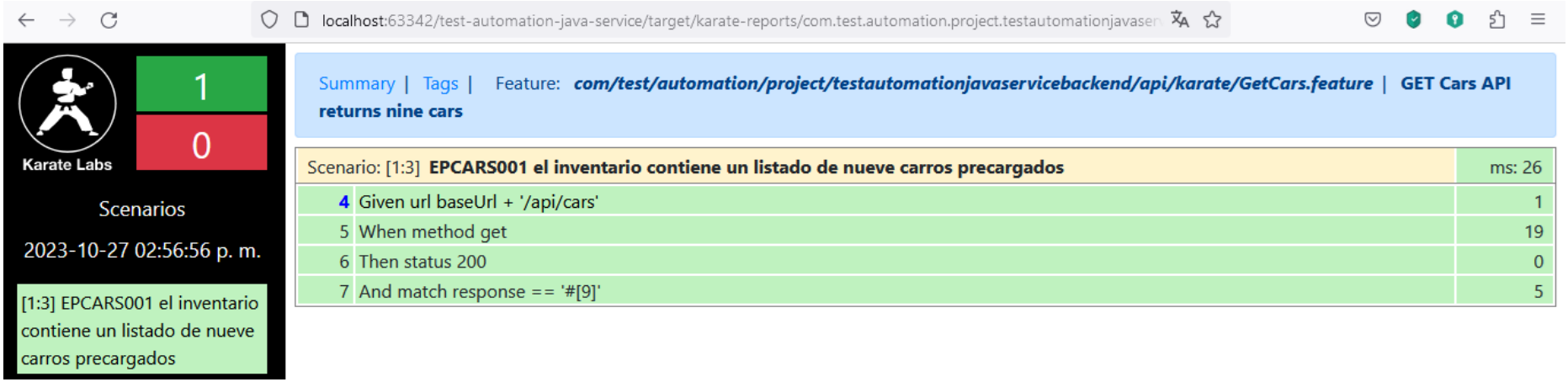
Features

2023-10-27 02:56:58 p. m.

Tags | Timeline

Feature	Title	Passed	Failed	Scenarios	Time (ms)
com/test/automation/project/testautomationjavaservicebackend/api/karate/GetCars.feature	GET Cars API returns nine cars	1	0	1	26
com/test/automation/project/testautomationjavaservicebackend/api/karate/SearchByPriceRange.feature	GET Search Cars By Price Range	7	0	7	234
com/test/automation/project/testautomationjavaservicebackend/api/karate/SearchByBrand.feature	GET Search Cars By Brand	9	0	9	269

Backend - Pruebas de Integración - Karate
Endpoint: GET /api/cars



Backend - Pruebas de Integración - Karate
Endpoint: GET api/search?initialPrice=<>&finalPrice=<>

localhost:63342/test-automation-java-service/target/karate-reports/com.test.automation.project.testautomatio 67%

Karate Labs

7

0

Scenarios

2023-10-27 02:56:57 p. m.

[1:3] EPSEARCHP001_searchCarsB

[2:11] EPSEARCHP002_searchCarsB

[3:19] EPSEARCHP003_searchCarsB

[4:27] EPSEARCHP004_searchCarsB

[5:35] EPSEARCHP005_searchCarsB

[6:43] EPSEARCHP006_searchCarsB

[7:51] EPSEARCHP007_searchCarsB

Summary | Tags | Feature: *com/test/automation/project/testautomationjavaservicebackend/api/karate/SearchByPriceRange.feature* | GET Search Cars By Price Range

Scenario: [1:3] EPSEARCHP001_searchCarsByPriceRange_DebeRetornarUnaListaConTresCarros_CuandoElPrecioDelCarroNoSuperaLos100Millones	ms: 21
4 Given url baseUrl + '/api/search'	1
5 And param initialPrice = 0.0	1
6 And param finalPrice = 100000000.0	1
7 When method get	14
8 Then status 200	0
9 And match response == '#[3]'	4

Scenario: [2:11] EPSEARCHP002_searchCarsByPriceRange_DebeRetornarUnaListaConTresCarros_CuandoElPrecioDelCarroEstribaEntre100MillonesY200Millones	ms: 43
12 Given url baseUrl + '/api/search'	1
13 And param initialPrice = 100000000.0	2
14 And param finalPrice = 200000000.0	1
15 When method get	35
16 Then status 200	0
17 And match response == '#[3]'	2

Scenario: [3:19] EPSEARCHP003_searchCarsByPriceRange_DebeRetornarUnaListaConTresCarros_CuandoElPrecioDelCarroSupereLos200Millones	ms: 34
20 Given url baseUrl + '/api/search'	3
21 And param initialPrice = 200000000.0	1
22 And param finalPrice = 1000000000.0	0
23 When method get	27
24 Then status 200	0
25 And match response == '#[3]'	3

Scenario: [4:27] EPSEARCHP004_searchCarsByPriceRange_DebeRetornarUnaListaCon6Carros_CuandoElPrecioDelCarroSupereLos100Millones	ms: 32
28 Given url baseUrl + '/api/search'	1
29 And param initialPrice = 100000000.0	0
30 And param finalPrice = 1000000000.0	0
31 When method get	20
32 Then status 200	0
33 And match response == '#[6]'	10

Backend - Pruebas de Integración - Karate
Endpoint: GET api/search?initialPrice=<>&finalPrice=<>

localhost:63342/test-automation-java-service/target/karate-reports/com.test.automation.project.testautomatio 67%

Karate Labs

7

0

Scenarios

2023-10-27 02:56:57 p. m.

[1:3] EPSEARCHP001_searchCarsB

[2:11] EPSEARCHP002_searchCarsB

[3:19] EPSEARCHP003_searchCarsB

[4:27] EPSEARCHP004_searchCarsB

[5:35] EPSEARCHP005_searchCarsB

[6:43] EPSEARCHP006_searchCarsB

[7:51] EPSEARCHP007_searchCarsB

23	When method get	27
24	Then status 200	0
25	And match response == '#[3]'	3

Scenario: [4:27] EPSEARCHP004_searchCarsByPriceRange_DebeRetornarUnaListaCon6Carros_CuandoElPrecioDelCarroSupereLos100Millones — ms: 32

28	Given url baseUrl + '/api/search'	1
29	And param initialPrice = 100000000.0	0
30	And param finalPrice = 1000000000.0	0
31	When method get	20
32	Then status 200	0
33	And match response == '#[6]'	10

Scenario: [5:35] EPSEARCHP005_searchCarsByPriceRange_DebeRetornarUnaListaVacia_CuandoElPrecioDelCarroSeaInferiorA50Millones — ms: 32

36	Given url baseUrl + '/api/search'	1
37	And param initialPrice = 0.0	1
38	And param finalPrice = 50000000.0	1
39	When method get	25
40	Then status 200	0
41	And match response == '#[0]'	5

Scenario: [6:43] EPSEARCHP006_searchCarsByPriceRange_DebeRetornarUnaListaCon9Carros_CuandoElPrecioDelCarroSupereLos50Millones — ms: 41

44	Given url baseUrl + '/api/search'	0
45	And param initialPrice = 50000000.0	1
46	And param finalPrice = 1000000000.0	10
47	When method get	25
48	Then status 200	0
49	And match response == '#[9]'	5

Scenario: [7:51] EPSEARCHP007_searchCarsByPriceRange_DebeLanzarUnIllegalArgumentException_CuandoElPrecioInicialDelCarroSupereAlPrecioFinalParaFiltrar — ms: 32

52	Given url baseUrl + '/api/search'	1
53	And param initialPrice = 1000000000.0	0
54	And param finalPrice = 50000000.0	1
55	When method get	25
56	Then status 400	0
57	And match karate.toString(response) contains "Bad Request"	5

Backend - Pruebas de Integración - Karate
Endpoint: GET /api/searchBrands?brands=<>,<>,<>,…

localhost:63342/test-automation-java-service/target/karate-reports/com.test.automation.project.testautomatio 80%

Karate Labs

9

0

Scenarios

2023-10-27 02:56:57 p. m.

[1:3] EPSEARCHB001_searchCarsBy

[2:10] EPSEARCHB002_searchCarsBy

[3:17] EPSEARCHB003_searchCarsBy

[4:24] EPSEARCHB004_searchCarsBy

[5:31] EPSEARCHB005_searchCarsBy

[6:38] EPSEARCHB006_searchCarsBy

[7:45] EPSEARCHB007_searchCarsBy

[8:52] EPSEARCHB008_searchCarsBy

[9:59]

Summary | Tags | Feature: *com/test/automation/project/testautomationjavaservicebackend/api/karate/SearchByBrand.feature* | **GET Search Cars By Brand**

Scenario: [1:3] **EPSEARCHB001_searchCarsByBrands_DebeDevolver3Carros_CuandoSeFiltrePorLaMarcaBMW**	ms: 25
4 Given url baseUrl + '/api/searchBrands'	2
5 And param brands = 'BMW'	1
6 When method get	16
7 Then status 200	0
8 And match response == '#[3]'	7

Scenario: [2:10] **EPSEARCHB002_searchCarsByBrands_DebeDevolver3Carros_CuandoSeFiltrePorLaMarcaKia**	ms: 27
11 Given url baseUrl + '/api/searchBrands'	2
12 And param brands = 'Kia'	1
13 When method get	20
14 Then status 200	0
15 And match response == '#[3]'	5

Scenario: [3:17] **EPSEARCHB003_searchCarsByBrands_DebeDevolver3Carros_CuandoSeFiltrePorLaMarcaPorche**	ms: 23
18 Given url baseUrl + '/api/searchBrands'	2
19 And param brands = 'Porche'	1
20 When method get	16
21 Then status 200	0
22 And match response == '#[3]'	5

Scenario: [4:24] **EPSEARCHB004_searchCarsByBrands_DebeDevolver6Carros_CuandoSeFiltrePorLaMarcaBMW_Y_Kia**	ms: 22
25 Given url baseUrl + '/api/searchBrands'	2
26 And param brands = 'BMW,Kia'	1
27 When method get	14
28 Then status 200	0
29 And match response == '#[6]'	6

Backend - Pruebas de Integración - Karate
Endpoint: GET /api/searchBrands?brands=<>,<>,<>,…

localhost:63342/test-automation-java-service/target/karate-reports/com.test.automation.project.testautomatio 67%

Karate Labs

9

0

Scenarios

2023-10-27 02:56:57 p. m.

[1:3] EPSEARCHB001_searchCarsB

[2:10] EPSEARCHB002_searchCarsB

[3:17] EPSEARCHB003_searchCarsB

[4:24] EPSEARCHB004_searchCarsB

[5:31] EPSEARCHB005_searchCarsB

[6:38] EPSEARCHB006_searchCarsB

[7:45] EPSEARCHB007_searchCarsB

[8:52] EPSEARCHB008_searchCarsB

[9:59] EPSEARCHB009_searchCarsB

29	And match response == '#[6]'	6

Scenario: [5:31] EPSEARCHB005_searchCarsByBrands_DebeDevolver6Carros_CuandoSeFiltrePorLaMarcaBMW_Y_Porche		ms: 30
32	Given url baseUrl + '/api/searchBrands'	2
33	And param brands = 'BMW,Porche'	1
34	When method get	23
35	Then status 200	0
36	And match response == '#[6]'	5

Scenario: [6:38] EPSEARCHB006_searchCarsByBrands_DebeDevolver6Carros_CuandoSeFiltrePorLaMarcaKia_Y_Porche		ms: 42
39	Given url baseUrl + '/api/searchBrands'	1
40	And param brands = 'Kia,Porche'	1
41	When method get	33
42	Then status 200	0
43	And match response == '#[6]'	8

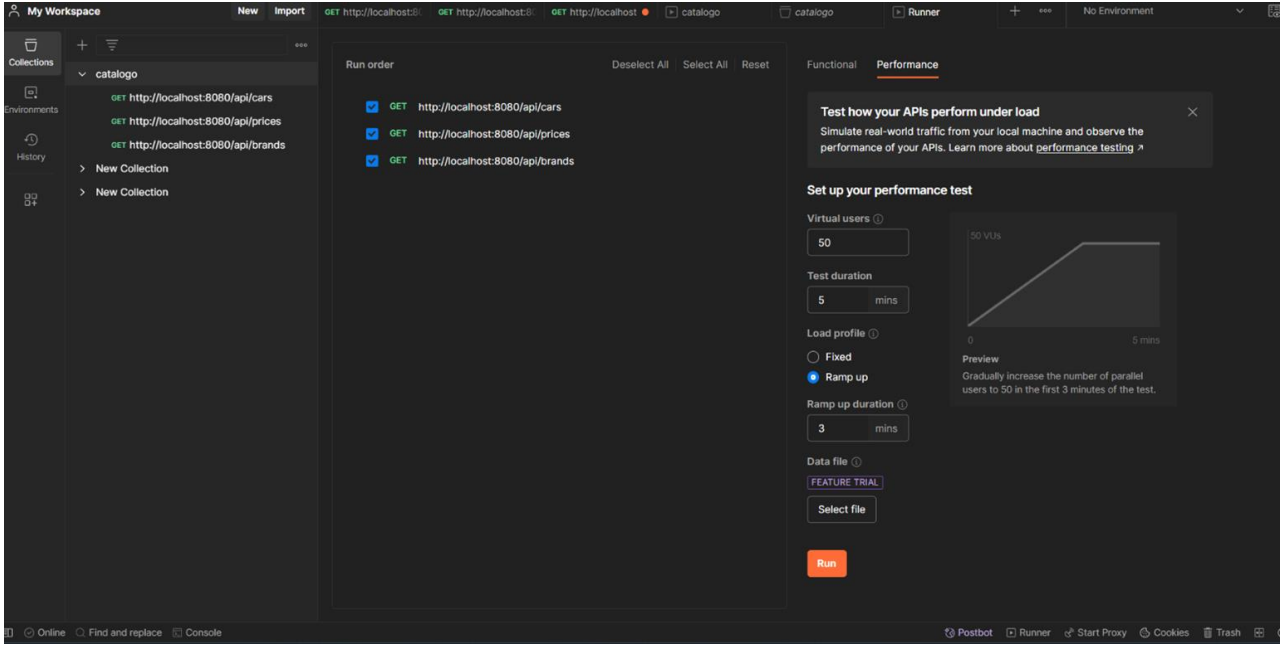
Scenario: [7:45] EPSEARCHB007_searchCarsByBrands_DebeDevolver9Carros_CuandoSeFiltrePorLaMarcaBMW_Y_Kia_Y_Porche		ms: 48
46	Given url baseUrl + '/api/searchBrands'	2
47	And param brands = 'BMW,Kia,Porche'	1
48	When method get	39
49	Then status 200	0
50	And match response == '#[9]'	6

Scenario: [8:52] EPSEARCHB008_searchCarsByBrands_DebeDevolver9Carros_CuandoNoSeEspecifiqueNingunaMarca_DadoQuePorDefectoBuscaraLasTresDisponibles		ms: 31
53	Given url baseUrl + '/api/searchBrands'	2
54	And param brands = ''	1
55	When method get	26
56	Then status 200	0
57	And match response == '#[9]'	2

Scenario: [9:59] EPSEARCHB009_searchCarsByBrands_DebeDevolver9Carros_CuandoNoSeEspecifiqueElParametroMarcas_DadoQuePorDefectoBuscaraLasTresDisponibles		ms: 19
60	Given url baseUrl + '/api/searchBrands'	1
61	When method get	15
62	Then status 200	0
63	And match response == '#[9]'	4

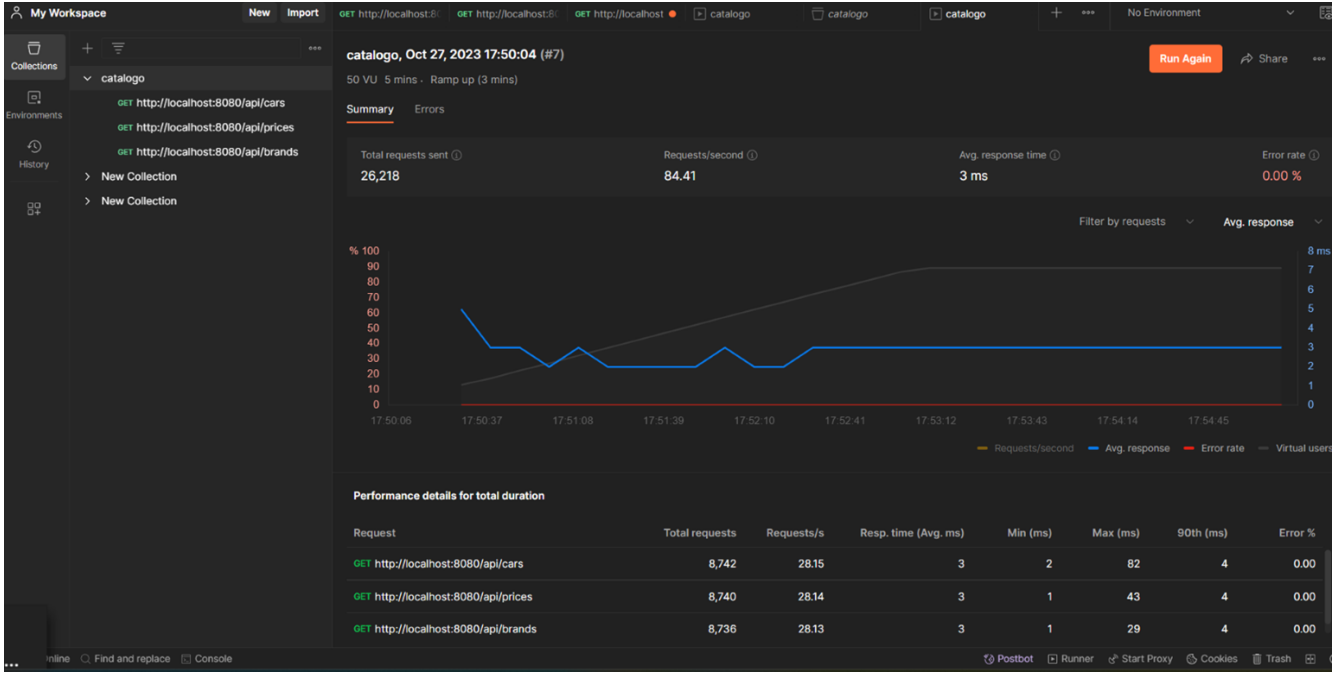
Diseño de las Pruebas de Desempeño

simular la carga y medir el rendimiento del backend Java en diferentes situaciones



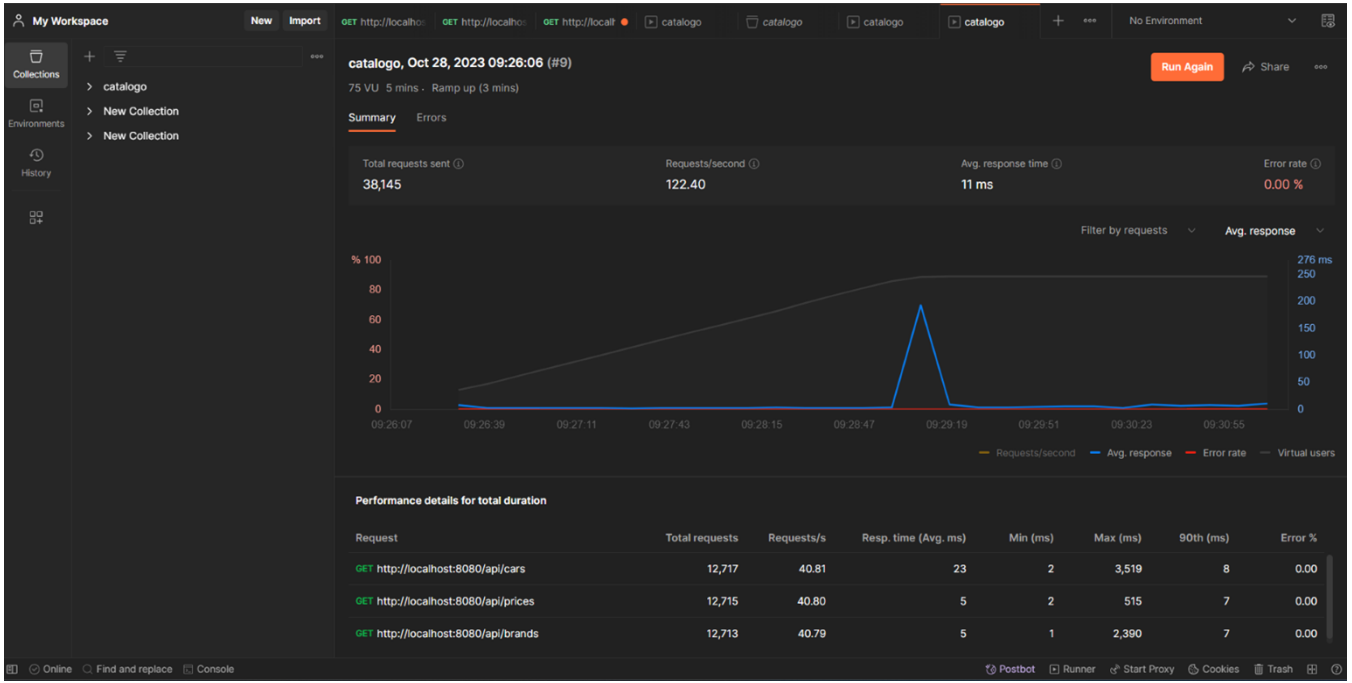
Diseño de las Pruebas de Desempeño

50 usuarios en la opción de rampa con periodo de carga en rampa de 3 min y de carga estable de 2 min,



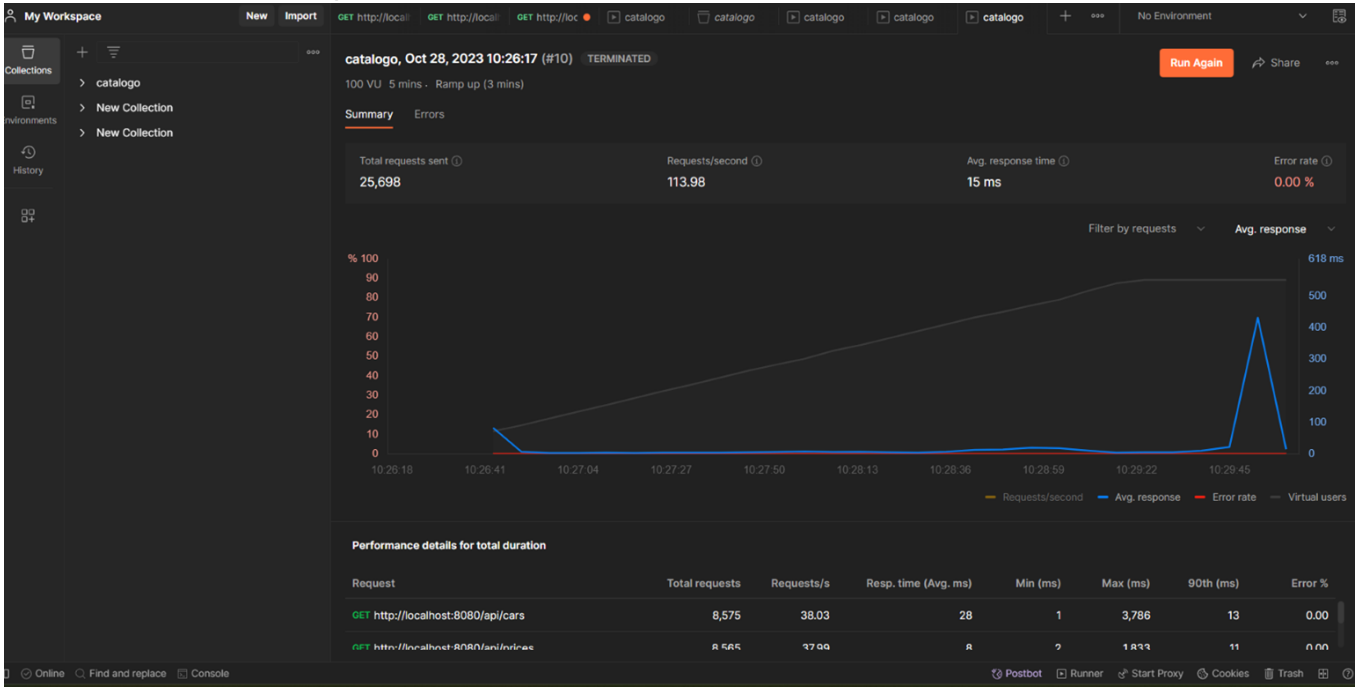
Diseño de las Pruebas de Desempeño

75 usuarios en la opción de rampa con periodo de carga en rampa de 3 min y de carga estable de 2 min,



Diseño de las Pruebas de Desempeño

100 usuarios en la opción de rampa con periodo de carga en rampa de 3 min y de carga estable de 2 min,



Peticiones con tiempos de respuesta más lentos

1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	3	4	5	11	2	82
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	3	4	4	7	1	43
GET http://localhost:8080/api/brands http://localhost:8080/api/searchBrands?brands=Porche,BMW,	3	4	4	6	1	29

1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	23	8	12	73	2	3,519
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	5	7	10	22	2	515
GET http://localhost:8080/api/brands http://localhost:8080/api/searchBrands?brands=Porche,BMW,	5	7	9	20	1	2,390

1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	28	13	28	216	1	3,786
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	8	11	19	72	2	1,833
GET http://localhost:8080/api/brands http://localhost:8080/api/searchBrands?brands=Porche,BMW,	8	11	17	65	1	2,292

Métricas de cada solicitud

2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	8,742	28.16	2	3	4	82	0
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	8,740	28.15	1	3	4	43	0
GET http://localhost:8080/api/brands	8,736	28.14	1	3	4	29	0

2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	12,717	40.95	2	23	8	3,519	0
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	12,715	40.94	2	5	7	515	0
GET http://localhost:8080/api/brands http://localhost:8080/api/searchBrands?brands=Porche,BMW,	12,713	40.94	1	5	7	2,390	0

1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
GET http://localhost:8080/api/cars http://localhost:8080/api/cars	28	13	28	216	1	3,786
GET http://localhost:8080/api/prices http://localhost:8080/api/search?initialPrice=76290000&finalPrice=97490000	8	11	19	72	2	1,833
GET http://localhost:8080/api/brands http://localhost:8080/api/searchBrands?brands=Porche,BMW,	8	11	17	65	1	2,292

THANK
YOU!