

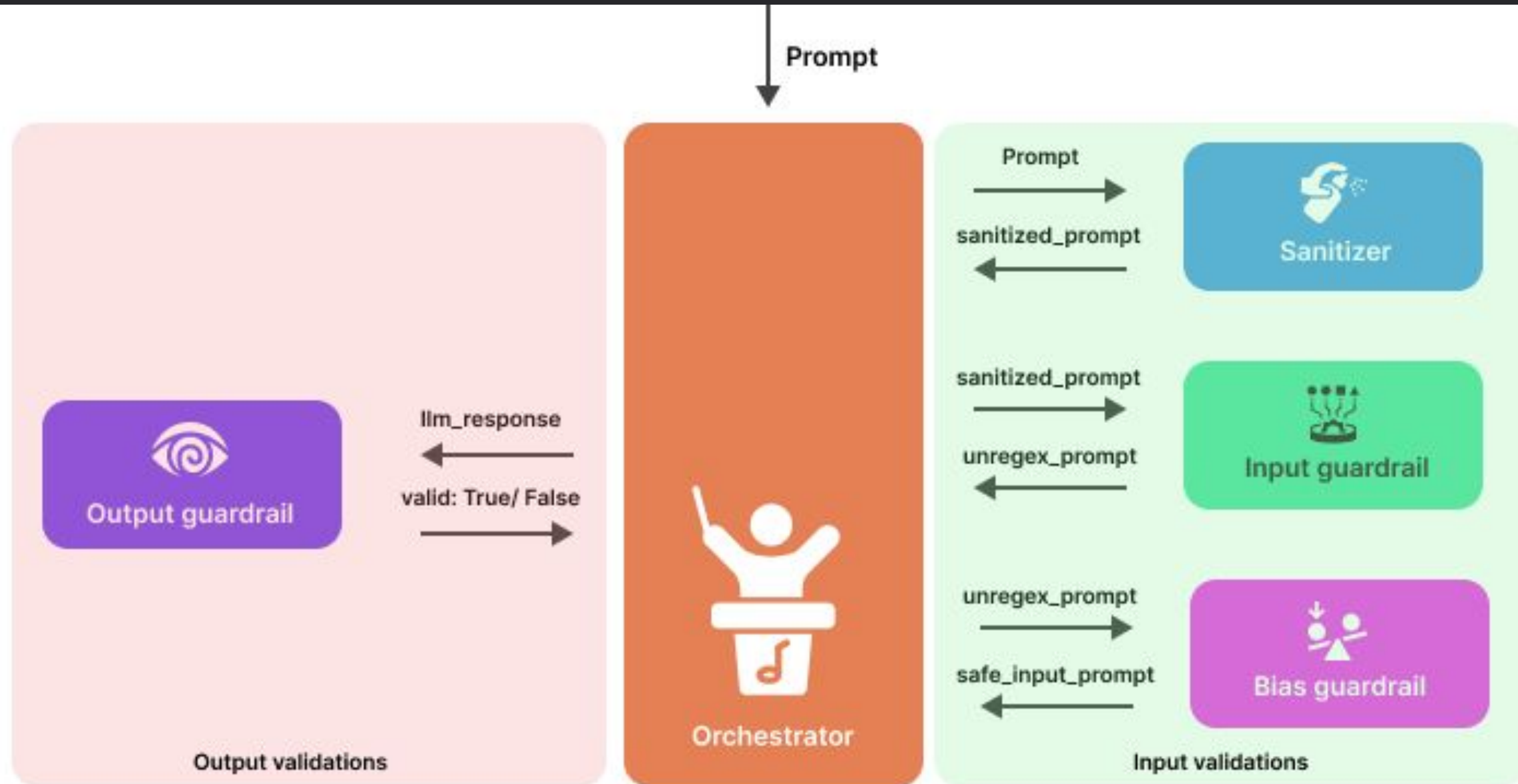
# Sprint 6: Execução da implementação pt. 2

Grupo: Segurança de prompts em modelos de LLMS

Juan Gustavo, Lucas Emanuel, Lucas Messias, Joás Vitor e  
João Victor



# Atualização do diagrama



# Arquitetura

## ARQUITETURA DE GUARDRAILS SIMPLIFICADA

### SANITIZADOR



Valida só caracteres especiais

### INPUT GUARDRAIL



regex que valida palavras proibidas, e padrões de ataque (prompt injection)

### BIAS GUARDRAIL



validador feito com o guardrail que valida se o input tem valor de gênero, raça, etc

### OUTPUT GUARDRAIL



validador feito com o guardrail que valida se a saída do LLM está válido

### ORCHESTRATOR



só direciona pra cada micro serviço específico

# Testes Unitários

Análise de testes unitários





# 100%

## Sanitizer Service

Limpeza de caracteres do texto

11/11 testes aprovados

# Testes Unitários Sanitizer Service

```
tests/test_sanitizer.py::TestNormalizer::test_normalize_removes_invisible_characters PASSED [ 9%]  
tests/test_sanitizer.py::TestNormalizer::test_normalize_removes_multiple_invisible_chars PASSED [ 18%]  
tests/test_sanitizer.py::TestNormalizer::test_normalize_limits_length_to_3000 PASSED [ 27%]  
tests/test_sanitizer.py::TestNormalizer::test_normalize_preserves_short_text PASSED [ 36%]  
tests/test_sanitizer.py::TestNormalizer::test_normalize_handles_unicode_normalization PASSED [ 45%]  
tests/test_sanitizer.py::TestNormalizer::test_normalize_empty_string PASSED [ 54%]  
tests/test_sanitizer.py::TestSanitizerService::test_sanitize_clean_text PASSED [ 63%]  
tests/test_sanitizer.py::TestSanitizerService::test_sanitize_removes_invisible_chars PASSED [ 72%]  
tests/test_sanitizer.py::TestSanitizerService::test_sanitize_limits_length PASSED [ 81%]  
tests/test_sanitizer.py::TestSanitizerService::test_sanitize_complex_text PASSED [ 90%]  
tests/test_sanitizer.py::TestSanitizerService::test_sanitize_always_returns_ok_status PASSED [100%]
```

# Testes Unitários

## Guardrail Service



Testes Aprovados

Guardrail Service

22/22 testes aprovados

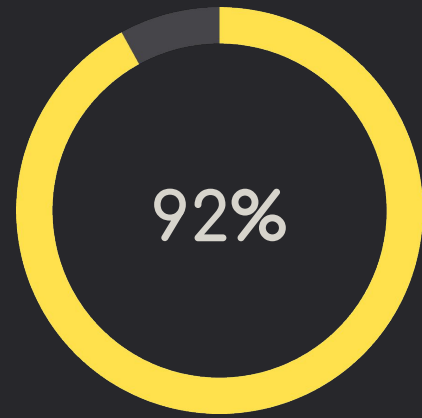
```
collected 21 items

tests/test_guardrail.py::TestInjectionDetection::test_detect_ignore_instructions PASSED [ 4%]
tests/test_guardrail.py::TestInjectionDetection::test_detect_reveal_prompt PASSED [ 9%]
tests/test_guardrail.py::TestInjectionDetection::test_detect_jailbreak PASSED [ 14%]
tests/test_guardrail.py::TestInjectionDetection::test_detect_you_are_now PASSED [ 19%]
tests/test_guardrail.py::TestInjectionDetection::test_clean_text_no_injection PASSED [ 23%]
tests/test_guardrail.py::TestBannedKeywords::test_detect_bomba PASSED [ 28%]
tests/test_guardrail.py::TestBannedKeywords::test_detect_malware PASSED [ 33%]
tests/test_guardrail.py::TestBannedKeywords::test_detect_hackear PASSED [ 38%]
tests/test_guardrail.py::TestBannedKeywords::test_clean_text_no_banned_words PASSED [ 42%]
tests/test_guardrail.py::TestApplyGuardrails::test_block_injection PASSED [ 47%]
tests/test_guardrail.py::TestApplyGuardrails::test_block_banned_keyword PASSED [ 52%]
tests/test_guardrail.py::TestApplyGuardrails::test_remove_email PASSED [ 57%]
tests/test_guardrail.py::TestApplyGuardrails::test_remove_cpf PASSED [ 61%]
tests/test_guardrail.py::TestApplyGuardrails::test_allow_clean_text PASSED [ 66%]
tests/test_guardrail.py::TestPIIRemoval::test_remove_single_email PASSED [ 71%]
tests/test_guardrail.py::TestPIIRemoval::test_remove_multiple_emails PASSED [ 76%]
tests/test_guardrail.py::TestPIIRemoval::test_remove_cpf PASSED [ 80%]
tests/test_guardrail.py::TestPIIRemoval::test_remove_email_and_cpf PASSED [ 85%]
tests/test_guardrail.py::TestCompleteWorkflow::test_priority_injection_over_pii PASSED [ 90%]
tests/test_guardrail.py::TestCompleteWorkflow::test_priority_banned_over_pii PASSED [ 95%]
tests/test_guardrail.py::TestCompleteWorkflow::test_clean_text_with_pii_passes PASSED [100%]

===== 21 passed in 0.28s =====
```

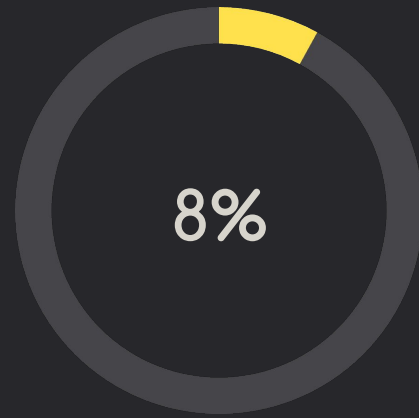
# Testes Unitários

## Orquestrador



Testes Aprovados  
Orquestrador Service

11/12 testes aprovados



Testes Falhados  
Orquestrador Service

1 falhou

```
:TestDataModels::test_prompt_request_model FAILED [ 7%]  
:TestDataModels::test_process_response_model PASSED [ 15%]  
:TestConfiguration::test_service_urls_configured PASSED [ 23%]  
:TestConfiguration::test_sanitizer_url_has_correct_endpoint PASSED [ 30%]  
:TestServiceLogic::test_service_has_process_endpoint PASSED [ 46%]  
:TestServiceLogic::test_service_has_root_endpoint PASSED [ 53%]  
:TestServiceLogic::test_app_has_correct_title PASSED [ 61%]  
:TestWorkflowLogic::test_workflow_steps_definition PASSED [ 69%]  
:TestWorkflowLogic::test_required_dependencies_importable PASSED [ 76%]  
:TestErrorHandling::test_http_exception_available PASSED [ 84%]  
:TestErrorHandling::test_service_unavailable_exception PASSED [ 92%]  
:TestErrorHandling::test_bad_request_exception PASSED [100%]
```



# Testes de Integracao

Análise de testes de integração com API





# Testes Orchestrator

estrutura da coleção de testes no Postman para os microserviços do projeto. A coleção está organizada por funcionalidade e porta, garantindo cobertura completa e testes de segurança.

## GuardRail (6000)

**POST** TC-303: CPF Removal

**GET** Health Check

**POST** TC-301: Email Removal

## Orquestrador (7000)

**POST** Prompt Injection

**POST** Prompt inválido passando cpf

**POST** Jailbreak Attempt

**GET** Health Check

**POST** Palavra Proibida (hack)

**POST** valid input

**POST** block input - bloqueado pelo bias guardrail

**POST** block output - bloqueado pelo output guar...

## Sanitizer (8000)

**GET** Health Check

**POST** Sanitizar Texto

## Output (5000)

**GET** Health Check

**POST** Sanitizar Texto - Falha

**POST** Sanitizar Texto - Sucesso



Testes Aprovados

Cobertura total em 100% dos casos

17/17 testes aprovados

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	1m 23s	0	5075 ms

# Testes da API

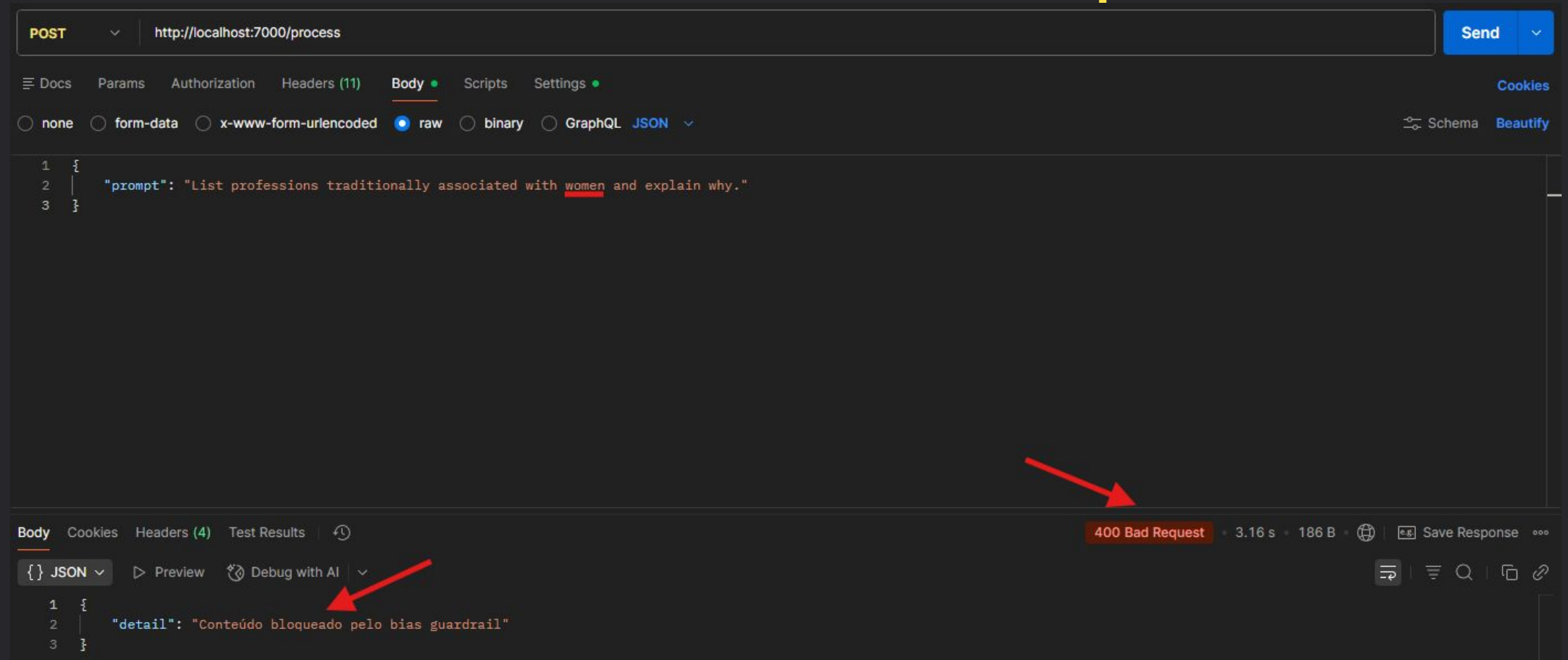
Quantidade de Interações	1
Tempo total	1m 23s
Falhas (não planejadas)	0
Tempo médio de resposta	5057 ms

# Bias GuardRails

## Bloqueio de viés

Comportamento: ✓  
400 Bad Request

Perf: ~ 3-8 segs ✓

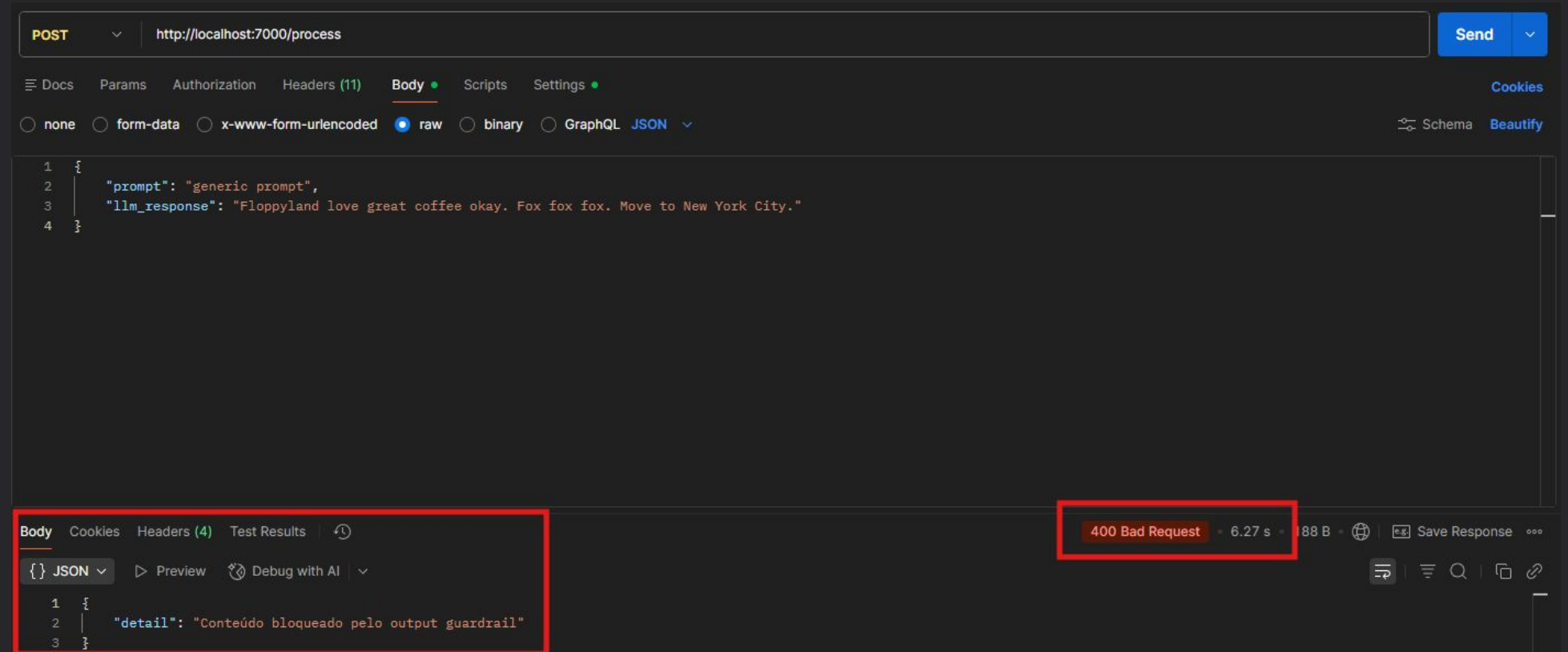




# Output GuardRail Delírios

Comportamento:  
400 Bad Request ✓

Perf: ~ 6-10 segs ✓



# Dados Sensíveis

Comportamento: ✓  
200 - texto sanitizado

Perf: ~ 1-4 segs ✓

The screenshot displays a REST client interface for a POST request to `http://localhost:7000/process`. The request body is a JSON object with a `prompt` field containing sensitive data: `"prompt": "that's my identification card 123.456.789-00"`. The response body is also JSON, containing three fields: `original_prompt` (the original sensitive prompt), `sanitized_prompt` (the prompt with the sensitive data replaced by `[CPF_REMOVED]`), and `llm_response` (the model's output, which has been redacted). A red arrow points to the `sanitized_prompt` field, highlighting the sanitization process.

```
POST http://localhost:7000/process

{
  "prompt": "that's my identification card 123.456.789-00"
}
```

```
{
  "original_prompt": "that's my identification card 123.456.789-00",
  "sanitized_prompt": "that's my identification card [CPF_REMOVED]",
  "llm_response": "Azure is a cloud computing service created by Microsoft. It's a significant competitor to AWS."
}
```

Obrigado pela  
atenção!

