

EXECUÇÃO E EVIDÊNCIAS DOS TESTES

1 TESTE DE INTEGRAÇÃO DE UNIDADES

Para o teste de integração tivemos como objetivo validar o funcionamento integrado das funcionalidades de cadastro de Aluno, Professor e Avaliação no sistema educacional. O que foi validado:

- Usuários do tipo Aluno e Professor sejam criados corretamente.
- A avaliação do aluno feita pelo professor armazene notas e faltas.
- O cálculo da média e o status do aluno estejam corretos.
- A associação entre usuário, aluno, professor e avaliação funcione conforme esperado.

1.1 Casos de Testes de Integração Unidades que foram testados

ID	Descrição	Resultado Esperado
CT01	Cadastro de Aluno	- Aluno cadastrado com sucesso
CT02	Cadastro de Professor	- Professor cadastrado com sucesso
CT03	Cadastro de Avaliação	- Avaliação com média e status calculados
CT04	Cálculo Automático de Média	- Média corretamente calculada
CT05	Cálculo Automático de Status	- Status corretamente calculado
CT06	Consulta de Avaliações por Aluno	- Visualização das próprias avaliações
CT07	Validação de Matrícula Única	- Mensagem de erro em caso de matrícula duplicada
CT08	Validação de Notas	- Mensagem de erro em caso de nota inválida

1.2 Resultados das execuções e evidências

Criamos o código de testes utilizando a ferramenta **Django TestCase** para validar o funcionamento do sistema. Para executar esses testes específicos, utilizamos o comando:

→ `python manage.py test alunos.test_integracao`

Ele roda todos os cenários de teste que verificam se o sistema relacionado ao módulo alunos está funcionando como esperado de forma integrada.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - escola + v [] [] ...
PS C:\Users\pc\CadastroEducatcional\escola> python manage.py test alunos.test_integracao
Found 8 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 8 tests in 0.019s

OK
Destroying test database for alias 'default'...
PS C:\Users\pc\CadastroEducatcional\escola>

```

O teste foi executado rapidamente e obteve o resultado OK, indicando que passou sem apresentar falhas ou erros. Após a execução, o Django removeu o banco de dados temporário criado especialmente para o teste, garantindo que o ambiente de desenvolvimento permaneça limpo.

1.3 Identificação de bugs e falhas encontradas

Nenhum bug crítico ou falha foi identificado durante a execução do teste.

1.4 Conclusão e lições aprendidas

Com o teste de integração podemos verificar se os fluxos do sistemas estão implementados de forma adequada. Com o Django TestCase foi possível antecipar problemas futuros e reduzir retrabalho. Foi essencial para garantir que o sistema funcione corretamente no conjunto, não apenas isoladamente.

2 TESTE DE CARGA

Neste teste o objetivo foi avaliar o desempenho do sistema sob múltiplas requisições simultâneas, especialmente na funcionalidade de login e acesso à página de boletim do aluno. Foram analisados:

- Tempo de resposta do sistema durante picos de acesso.
- Estabilidade do processo de autenticação, garantindo que os logins sejam processados corretamente sem falhas.
- Resistência da aplicação, verificando se o sistema permanece funcional e responsivo mesmo com diversos usuários acessando simultaneamente.

2.1 Casos de Testes de Carga que foram testados

ID	Descrição	Resultado Esperado
CT06	Consulta de Avaliações por Aluno	- Visualização das próprias avaliações
CT09	Acesso autenticado ao boletim do aluno	- Página de boletim exibida com sucesso (status 200)

2.2 Resultados das execuções e evidências

O teste foi implementado utilizando a ferramenta **Locust**, simulando múltiplos usuários acessando a aplicação simultaneamente. Basicamente ele executa a página de login (/login/) para obter o token CSRF necessário, realiza o login utilizando as credenciais do usuário alice.beatriz. Por fim, com a autenticação bem sucedida, acessa a página de boletim do aluno (/aluno/boletim/).

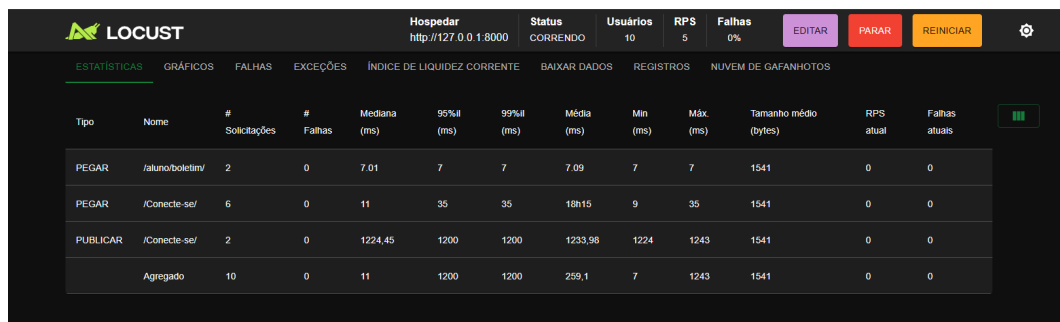
Primeiramente, nós rodamos o servidor Django localmente com o comando:

→ `python manage.py runserver`

Com a ferramenta Locust, esse fluxo foi executado por diversos usuários virtuais com intervalos aleatórios entre as requisições, permitindo avaliar o comportamento do sistema sob carga contínua. Iniciamos o Locust com o comando:

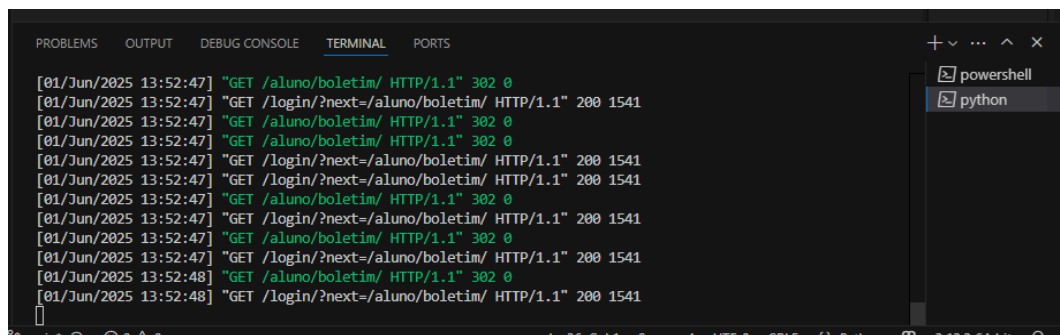
→ `locust -f test_carga.py --host=http://127.0.0.1:8000`

Para acessar abrimos a interface web do Locust em `http://localhost:8089` e configuramos a quantidade de usuários simultâneos e a taxa de geração.



Tipo	Nome	# Solicitações	# Falhas	Mediana (ms)	95%il (ms)	99%il (ms)	Média (ms)	Min (ms)	Máx (ms)	Tamanho médio (bytes)	RPS atual	Falhas atuais
PEGAR	/aluno/boletim/	2	0	7.01	7	7	7.09	7	7	1541	0	0
PEGAR	/Conecte-se/	6	0	11	35	35	18h15	9	35	1541	0	0
PUBLICAR	/Conecte-se/	2	0	1224.45	1200	1200	1233.98	1224	1243	1541	0	0
Agregado		10	0	11	1200	1200	259.1	7	1243	1541	0	0

Observamos que o login foi realizado com usuário e senha válidos, recebendo status HTTP 200 assim como o acesso ao boletim do aluno também.



Timestamp	Request	Status	Size
[01/Jun/2025 13:52:47]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:47]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541
[01/Jun/2025 13:52:47]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:47]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:47]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541
[01/Jun/2025 13:52:47]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541
[01/Jun/2025 13:52:47]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:47]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541
[01/Jun/2025 13:52:47]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:47]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541
[01/Jun/2025 13:52:48]	"GET /aluno/boletim/ HTTP/1.1"	302	0
[01/Jun/2025 13:52:48]	"GET /login/?next=/aluno/boletim/ HTTP/1.1"	200	1541

Foi analisado também que algumas requisições POST ao login demoraram mais tempo (~1200 ms), possível ponto para análise futura.

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s		
GET	/aluno/boletim/	442	0(0.00%)	9	4	20	9	4.84	0.00		
GET	/login/	10	0(0.00%)	15	8	35	11	0.11	0.00		
POST	/login/	10	0(0.00%)	1219	1196	1244	1200	0.11	0.00		
Aggregated		462	0(0.00%)	36	4	1244	10	5.06	0.00		
Response time percentiles (approximated)											
Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%
%	100% # reqs										

2.3 Identificação de bugs e falhas encontradas

Durante toda a simulação, nenhum erro crítico ou falha HTTP foi registrado.

2.4 Conclusão e lições aprendidas

Podemos concluir que os testes de carga demonstraram que o sistema é capaz de lidar com múltiplos acessos simultâneos de forma estável e com baixa latência, mantendo a responsividade e sem apresentar falhas.

3 TESTE DE STRESS

Avaliamos a resistência e o comportamento do sistema sob carga extrema, com o intuito de identificar:

- Monitoramento de áreas que apresentam lentidão sob carga intensa.
- Determinação do número máximo de usuários simultâneos que o sistema consegue suportar mantendo um desempenho aceitável.
- Identificação de momentos em que o sistema deixa de responder.

3.1 Casos de Testes de Stress que foram testados

ID	Descrição	Resultado Esperado
CT09	Acesso autenticado ao boletim do aluno	- Página de boletim exibida com sucesso (status 200)
CT10	Acesso do Aluno ao Sistema	- Login realizado com sucesso - Página de boletim exibida com sucesso - Página de dashboard exibida com sucesso

3.2 Resultados das execuções e evidências

No teste de stress nós também usamos a ferramenta **Locust**, simulando dois tipos de carga. Mas primeiramente nós rodamos o servidor Django localmente com o comando:

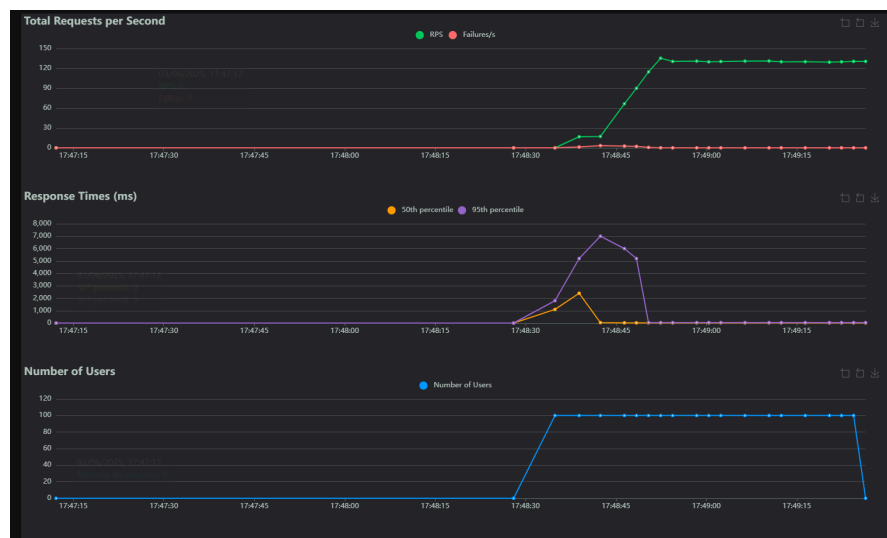
→ `python manage.py runserver`

Nós fizemos primeiramente com 100 usuários no total, rodando 50 deles simultaneamente no Locust. Através do comando:

```
→ locust -f test_stress.py --host=http://127.0.0.1:8000
```

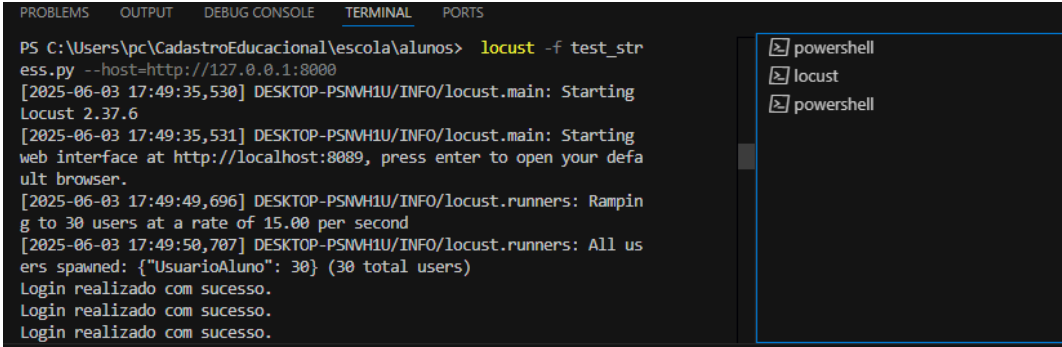
As rotas que foram trafegadas pelos usuários virtuais simulando acesso simultâneo em /login/, /dashboard/aluno/ e /aluno/boletim/.

<div>  <div> <div>Hospedar</div> <div>http://127.0.0.1:8000</div> </div> <div> <div>Status</div> <div>PAROU</div> </div> <div> <div>RPS</div> <div>130.6</div> </div> <div> <div>Falhas</div> <div>0%</div> </div> <div>NOVO</div> <div>REINICIAR</div> </div>												
<div> <div>ESTATÍSTICAS</div> <div>GRÁFICOS</div> <div>FALHAS</div> <div>EXCEÇÕES</div> <div>ÍNDICE DE LIQUIDEZ CORRENTE</div> <div>BAIXAR DADOS</div> <div>REGISTROS</div> <div>NUVEM DE GAFANHOTOS</div> </div>												
Tipo	Nome	# Solicitações	# Falhas	Mediana (ms)	95%il (ms)	99%il (ms)	Média (ms)	Min (ms)	Máx (ms)	Tamanho médio (bytes)	RPS atual	Falhas atuais
PEGAR	/aluno/boletim/	236	0	17	2000	3700	264,3	4	4152	1541	0,75	0
PEGAR	/painel/aluno/	90	0	18	3100	4300	422,25	5	4288	1541	0,5	0
PEGAR	/Conecte-se/	100	27	1400	5200	5300	2022,68	169	5296	1124,93	12,5	3,38
PUBLICAR	/Conecte-se/	73	0	6500	7500	7500	5946,87	2303	7545	1541	3,5	0
Agregado		499	27	110	7000	7500	1476,49	4	7545	1457,62	17h25	3,38



Observamos que o teste indicou 499 requisições, com 27 falhas concentradas apenas no endpoint GET /Conecte-se/, representando cerca de 5,4% de falhas totais. A latência média do sistema foi de aproximadamente 1,47 segundos, porém com variações extremas que vão de milissegundos até picos superiores a 7 segundos. Indicando para nós que o sistema consegue respostas rápidas, mas possui momentos de maior carga ou em determinadas operações.

Para verificar se o sistema apresentava problemas com carga, realizamos um teste com 30 usuários simulados, sendo 15 ativos simultaneamente no Locust. O resultado foi 0% de falhas, o que indica que o sistema conseguiu lidar bem com essa quantidade de usuários simultâneos, sem apresentar erros ou lentidão significativa.





LOCUST

Hospedar

http://127.0.0.1:8000

Status

PAROU

RPS

39.1

Falhas

0%

NOVO

REINICIAR

ESTATÍSTICAS

GRÁFICOS

FALHAS

EXCEÇÕES

ÍNDICE DE LIQUIDEZ CORRENTE

BAIXAR DADOS

REGISTROS

NUVEM DE GAFANHOTOS

Tipo	Nome	# Solicitações	# Falhas	Mediana (ms)	95%il (ms)	99%il (ms)	Média (ms)	Min (ms)	Máx. (ms)	Tamanho médio (bytes)	RPS atual	Falhas atuais
PEGAR	/aluno/boletim/	445	0	12	24	31	13,7	4	183	1541	29	0
PEGAR	/painel/aluno/	147	0	10	28	39	13,91	5	105	1541	10.1	0
PEGAR	/Conecte-se/	30	0	45	59	70	45,55	27	70	1541	0	0
PUBLICAR	/Conecte-se/	30	0	2400	2800	2800	2396,29	1844	2835	1541	0	0
Agregado		652	0	12	100	2600	124,84	4	2835	1541	39.1	0



3.3 Identificação de bugs e falhas encontradas

Inicialmente, observamos o endpoint `/aluno/boletim/`, que obteve 236 requisições sem falhas, indicando uma boa estabilidade do serviço. Mas, é importante destacar que os percentis 95 e 99 mostram que alguns pedidos chegaram a demorar até 2 e 3,7 segundos, ou seja, embora a maioria das requisições seja atendida rapidamente, existem casos em que o sistema apresenta lentidão.

Da mesma forma que o endpoint `/painel/aluno/`, apesar de ter um somente 90 requisições, também não apresentou falhas e mostrou latências medianas baixas. Já o endpoint `/Conecte-se/` no método GET apresentou problemas mais graves, com 27 falhas em 100 requisições, correspondendo a uma taxa de falhas de 27%.

Porém quando fizemos o teste com 30 usuários, tivemos um total de 0% de falha, concluindo que o sistema sofre com pico de estresse com muitos usuários simultâneos.

3.4 Conclusões e lições aprendidas

Com base no teste de stress nós conseguimos avaliar o comportamento do sistema sob condições extremas de carga. No teste de 100 usuários, precisaremos melhorar o sistema s, especialmente no endpoint `/Conecte-se/`, que apresentou uma taxa de falhas de 27%. Porém não tivemos erros com 30 usuários, mas embora o sistema suporta bem uma carga leve a moderada, ele apresenta falhas e lentidão em cenários de estresse com muitos usuários simultâneos.

4 TESTE DE ACEITAÇÃO

Avaliamos, ao final do processo, se o sistema atende as necessidades e expectativas solicitadas:

- Avaliação das funcionalidades principais do sistema, com foco na lógico de negócio
- Avaliação se está ocorrendo o redirecionamento de forma correta.
- Identificação dos dados exibidos estão coerentes com o cálculo desejado

4.1 Casos de Testes de Aceitação que foram testados

ID	Descrição	Resultado Esperado
CT01	Testar o cadastro de um novo aluno no sistema.	<ul style="list-style-type: none"> - Aluno cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT02	Testar o cadastro de um novo professor no sistema.	<ul style="list-style-type: none"> - Professor cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT03	Testar o cadastro de uma nova avaliação para um aluno.	<ul style="list-style-type: none"> - Avaliação cadastrada com sucesso - Média calculada de forma automática (resultado esperado da média é 7.75) - Status calculado de forma automática (resultado esperado do status “Aprovado”)
CT04	Verificar se o sistema calcula corretamente a média das notas.	<ul style="list-style-type: none"> - Cálculo da média esperada sendo 6.0

CT05	Verifica se o sistema calcula corretamente o status do aluno.	- Resultado do status esperado seja "Reprovado"
-------------	---	---

Tabela 27 - Tabela Casos de testes de aceitação que foram testados

4.2 Resultados das execuções e evidências

No teste de aceitação nós utilizamos o framework Pytest, um framework próprio Django, em um ambiente de desenvolvimento voltado para a realização de testes automatizados. A execução dos testes resultou em 4 testes com sucesso e um teste com falha.

Nós testes que obtiveram sucesso foi validado os seguintes pontos:

- Cadastro de alunos e professor: se foi criado corretamente os usuários no banco de dados, tendo a vinculação às respectivas tabelas, sendo elas Aluno e Professor.
- Login de aluno: se o resultado do redirecionamento foi a rota /dashboard/aluno/, demonstrando se houve a autenticação e autorização correta.
- Lançamento de avaliação: se houve o processamento correto das notas e se a média aritmética atribuiu de forma correta o status de "Aprovado" ou "Reprovado".
- Boletim do aluno: se todos os dados esperados foram exibidos de forma correta. Sendo eles: nome do aluno, notas lançadas, média e status.

O comando executado para a realização do teste em específico foi:

→ `python manage.py test alunos.test_aceitacao`

Com isso, o caso de teste 5 obteve falha e a mensagem fornecida foi:

AssertionError: False is not true : Couldn't find '5.50' in the following response

...

```
<div class="alert alert-info">
```

```
  <p>Nenhuma avaliação registrada ainda.</p>
```

```
</div>
```

Desta forma, identificamos que esse erro indica que a média 5.0, que era esperada, não foi exibida no HTML. Assim, compreende-se que nenhuma avaliação foi cadastrada no momento da execução do teste ou que o cálculo da média não foi realizado conforme o esperado.

4.3 Identificação de bugs e falhas encontradas

Sendo assim, identificamos um Bug no teste CT05 em que o sistema não exibe a média calculada no boletim do aluno. Com isso, foi retornada a mensagem “Nenhuma avaliação registrada ainda”, que sugere a ausência de dados simulados durante a execução do teste.

Para correção deste teste, pode-se realizar duas abordagens:

1. Verificando se o fixture ou método setUp() está inserido de forma correta com as avaliações necessárias no banco de dados;
2. Garantir se a lógica de cálculo e a exibição da média está chamando os dados mockados.

4.4 Conclusões e lições aprendidas

Com base na execução do teste de aceitação pudemos certificar se nosso usuário final possui todas as necessidades e expectativas alcançadas.

Esse tipo de teste foca na experiência do usuário e no comportamento funcional desse sistema, sendo caracterizada ainda como um teste de caixa preta, pois é validado apenas o funcionamento do sistema.

Esse tipo de teste ajuda a identificar quais são as lacunas nos dados de teste e, ainda, reforça a importância da configuração correta no ambiente de testes automatizados.

5 TESTE DE COMPONENTE

Avaliamos se durante o teste houve eficiência e comunicação entre sistemas.

- Verificação se a integração está correta entre os componentes do sistema.
- Verificação se está havendo o redirecionamento de forma correta.
- Garantia do bom funcionamento das funcionalidades do sistema.

5.1 Casos de Testes de Componente que foram testado

ID	Descrição	Resultado Esperado
CT01	Testar o cadastro de um novo aluno no sistema.	<ul style="list-style-type: none"> - Aluno cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT02	Testar o cadastro de um novo professor no sistema.	<ul style="list-style-type: none"> - Professor cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados

CT03	Testar o cadastro de uma nova avaliação para um aluno.	<ul style="list-style-type: none"> - Avaliação cadastrada com sucesso - Média calculada de forma automática (resultado esperado da média é 7.75) - Status calculado de forma automática (resultado esperado do status “Aprovado”)
CT04	Verificar se o sistema calcula corretamente a média das notas.	<ul style="list-style-type: none"> - Cálculo da média esperada sendo 6.0
CT05	Verifica se o sistema calcula corretamente o status do aluno.	<ul style="list-style-type: none"> - Resultado do status esperado seja “Reprovado”
CT06	Testa a consulta de avaliação por aluno	<ul style="list-style-type: none"> - Lista de todas as avaliações do aluno. - Médias e status de cada avaliação

5.2 Resultados das execuções e evidências

No teste de componente também foi utilizado o framework Pytest em um ambiente próprio de desenvolvimento de testes automatizados. A execução dos 8 testes resultou no seguinte resultado: 7 teste com sucesso e um teste com falha.

O erro apresentado no caso de teste 05 foi:

```
AssertionError: False is not true : Couldn't find '7.5' in the following response [...]
<p>Nenhuma avaliação registrada ainda.</p>
```

Isso significa que nenhuma das avaliações está aparecendo na página, mesmo eu criando diretamente no teste.

Os outros 7 testes foram executados de forma completa e obtiverem sucesso em sua execução.

5.3 Identificação de bugs e falhas encontradas

A falha observada no teste que verifica se o sistema calcula corretamente o status do aluno indica que nenhuma avaliação foi exibida na página do boletim do aluno, mesmo após a criação de uma instância de Avaliação. Sendo assim, a página HTML retornou a mensagem de que não há avaliações registradas apesar de ter sido criado uma avaliação.

A causa desse bug pode se dar devido às seguintes causas:

1. A avaliação pode estar sendo filtrada por usuário logado.
2. A criação da avaliação pode não estar associada corretamente ao professor.
3. A view pode usar request.user.aluno, mas o User usado não está vinculado a nenhum aluno.

4. A view pode estar esperando avaliações salvas via formulário com lógica adicional

5.4 Conclusões e lições aprendidas

Com base no teste de componente podemos compreender melhor o funcionamento de cada parte do sistema de forma individual e identificar uma falha. Com isso, o teste obteve falha por não ter sido exibida para o aluno em sua interface, embora a avaliação esteja sendo criada no banco de dados.

Essa falha apresenta uma inconsistência entre a camada de persistência e a camada de apresentação. Logo, essa falha indica que a lógica criada para a view pode estar ignorando dados válidos ou que há dependência de campos calculados que não estão sendo criados de forma automática como deveria.

Como aprendizados obtidos durante esse teste devemos alinhar a criação de dados no teste com a lógica real do sistema, necessário ter atenção com alguns métodos que podem ignorar outros métodos importantes que validam informações. Além disso, é importante verificar se todos os dados obrigatórios e quais os efeitos colaterais presentes, pois esse teste que houve a inserção dos dados diretamente no banco de dados deve simular o estado final que é esperado pela nossa aplicação.

6 TESTE DE INTEGRAÇÃO DE SISTEMA

Avaliamos se durante o teste houve eficiência e comunicação entre sistemas.

- Verificação se a integração está correta entre os componentes do sistema.
- Verificação se está havendo o redirecionamento de forma correta.
- Garantia do bom funcionamento das funcionalidades do sistema.

6.1 Casos de Testes de Integração de Sistema que foram testados

ID	Descrição	Resultado Esperado
CT01	Testar o cadastro de um novo aluno no sistema.	<ul style="list-style-type: none"> - Aluno cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT02	Testar o cadastro de um novo professor no sistema.	<ul style="list-style-type: none"> - Professor cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT03	Testar o cadastro de uma nova avaliação para um aluno.	<ul style="list-style-type: none"> - Avaliação cadastrada com sucesso - Média calculada de forma automática (resultado esperado da média é 7.75) - Status calculado de forma automática (resultado esperado do status "Aprovado")

CT06	Testa a consulta de avaliação por aluno	<ul style="list-style-type: none"> - Lista de todas as avaliações do aluno. - Médias e status de cada avaliação
CT09	Acesso autenticado ao boletim do aluno	<ul style="list-style-type: none"> - Página de boletim exibida com sucesso (status 200)

6.2 Resultados das execuções e evidências

No teste de integração do sistema ainda foi utilizado a ferramenta Django, Pytets, por ser uma ferramenta automatizada e correspondente a esse tipo de teste. Desta forma, os 5 testes foram executados com sucesso com o status retornando 200 e 302, conforme o esperado.

Os objetos criados - User, Aluno, Professor, Avaliacao - foram validados e obtiveram sucesso a partir da consulta ao banco de dados. Além disso, todos os redirecionamentos e conteúdos das páginas foram verificados conforme o esperado.

A evidência do sucesso dos teste está no próprio asserts dos teste, que passam sem nenhuma exceção. Os asserts fornecidos foram:

- ➔ `assertEqual(response.status_code, 302)`: que confirma o redirecionamento correto após o POST
- ➔ `assertRedirects(...)`: que confirma o caminho para onde os usuários foram encaminhados
- ➔ `assertTrue9User.objects.filter(...).exists())`: em que válida a persistência de todos os dados
- ➔ `assertContains(response, 'Boletim')`: onde há a confirmação da renderização correta do conteúdo

6.3 Identificação de bugs e falhas encontradas

Nesse teste, não obtivemos nenhuma falha ou bug durante toda a execução. Todos os fluxos principais de cadastro, avaliação e consulta foram validados de maneira completa e efetiva.

6.4 Conclusões e lições aprendidas

Com base no teste de integração do sistema objetivos como resultado que o sistema funciona corretamente seguindo os fluxos pré-estabelecidos. Entre os fluxos, pode-se considerar sendo eles o de cadastro de alunos e professores, o lançamento de avaliações por professores, a consulta de avaliações e acesso ao boletim por alunos. Desta maneira, comprova que os componentes da nossa aplicação estão conectados e que todos os dados fluem de forma correta entre todas as camadas do sistema.

Entre os principais aprendizados obtidos por meio da realização desse teste destacamos o entendimento que a realização desse teste garante que o sistema esteja funcionando de forma esperada no mundo real. Também, consideramos que separar responsabilidades por classe de teste ajuda na boa manutenção e legibilidade dos testes, além que testes automatizados ajudam a prevenir regressões.

7 TESTE DE INTERFACE DO USUÁRIO

Avaliamos por meio desse teste como diferentes componentes do sistema se comunicam entre si para garantir uma troca precisa de todos os dados.

- Verificação se as entradas e saídas entre os sistemas são tratadas corretamente
- Garantia do bom funcionamento do sistema como um todo.

7.1 Casos de Testes de Interface de Usuário que foram testados

ID	Descrição	Resultado Esperado
CT01	Testar o cadastro de um novo aluno no sistema.	<ul style="list-style-type: none"> - Aluno cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT02	Testar o cadastro de um novo professor no sistema.	<ul style="list-style-type: none"> - Professor cadastrado com sucesso - Mensagem de confirmação enviada - Dados salvos no banco de dados
CT03	Testar o cadastro de uma nova avaliação para um aluno.	<ul style="list-style-type: none"> - Avaliação cadastrada com sucesso - Média calculada de forma automática (resultado esperado da média é 7.75) - Status calculado de forma automática (resultado esperado do status “Aprovado”)
CT04	Verificar se o sistema calcula corretamente a média das notas.	<ul style="list-style-type: none"> - Cálculo da média esperada sendo 6.0
CT05	Verifica se o sistema calcula corretamente o status do aluno.	<ul style="list-style-type: none"> - Resultado do status esperado seja “Reprovado”
CT06	Testa a consulta de avaliação por aluno	<ul style="list-style-type: none"> - Lista de todas as avaliações do aluno. - Médias e status de cada avaliação
CT09	Acesso autenticado ao boletim do aluno	<ul style="list-style-type: none"> - Página de boletim exibida com sucesso (status 200)

7.2 Resultados das execuções e evidências

No teste de interface do sistema foi obtido sucesso em todos os 7 casos de testes realizados durante o código. Evidências obtidas durante a execução do teste:

- Usuário User e Aluno forma criado de forma correta redirecionando-os
- No teste do cadastro do professor obteve sucesso na criação da função __str__ do professor
- Cadastro da avaliação de forma correta e com o status esperado
- Conteúdo do Boletim em string em conformidade e o status 200

7.3 Identificação de bugs e falhas encontradas

Nesse teste, não obtivemos nenhuma falha ou bug durante toda a execução. Todos os fluxos principais de cadastro, avaliação e consulta foram validados de maneira completa e efetiva.

7.4 Conclusões e lições aprendidas

Com base no teste de interface do usuário, os testes automatizados cobriram os principais fluxos do sistema e obtiveram sucesso em cada um deles. Desta forma, obtivemos o correto funcionamento dos modelos Django, avaliamos a lógica de negócio relacionada às principais funcionalidades do sistema, garantimos o acesso controlado e seguro via autenticação de usuários e a representação coerente das informações em string, conforme esperado.

Entre os principais aprendizados obtidos por meio da realização desse teste destacamos como os testes automatizados garantem a confiabilidade, por cobrir todos os principais caminhos. Além disso, pudemos validar se o acesso ao boletim após login mostrou que a autenticação está funcionando de forma correta.

8 TESTE END-TO-END (E2E)

Avaliamos por meio desse teste se o funcionamento do sistema desde a forma inicial ao seu ponto final está acontecendo da forma esperada, simulando a experiência real do usuário.

- Verificação se as entradas e saídas entre os sistemas são tratadas corretamente
- Garantia do bom funcionamento do sistema como um todo.

8.1 Casos de Testes de End-to-end que foram testados

ID	Descrição	Resultado Esperado
CT01	Cadastro de Aluno	- Aluno cadastrado com sucesso

CT02	Cadastro de Professor	- Professor cadastrado com sucesso
CT03	Cadastro de Avaliação	- Avaliação com média e status calculados
CT04	Cálculo Automático de Média	- Média corretamente calculada
CT05	Cálculo Automático de Status	- Status corretamente calculado
CT06	Consulta de Avaliações por Aluno	- Visualização das próprias avaliações
CT07	Validação de Matrícula Única	- Mensagem de erro em caso de matrícula duplicada
CT08	Validação de Notas	- Mensagem de erro em caso de nota inválida

8.2 Resultados das execuções e evidências

No teste end-to-end o resultado da execução dos testes foi falhas em todos os textos, tendo sido evidenciado mensagens de erros repetidos em todos os oito casos. Os testes feito foram utilizando o Selenium WebDriver com o navegador Chrome, sendo o WebDriver a parte responsável pelo erro nos testes, com o objetivo de verificar funcionalidades do sistema.

Como dito anteriormente, a falha aconteceu no momento da criação do navegador Chrome via Selenium que significa que o Selenium está tentando executar um arquivo não executável válido do Windows. Portanto, o erro identificado foi com relação ao GoogleDriver e não com os testes.

8.3 Identificação de bugs e falhas encontradas

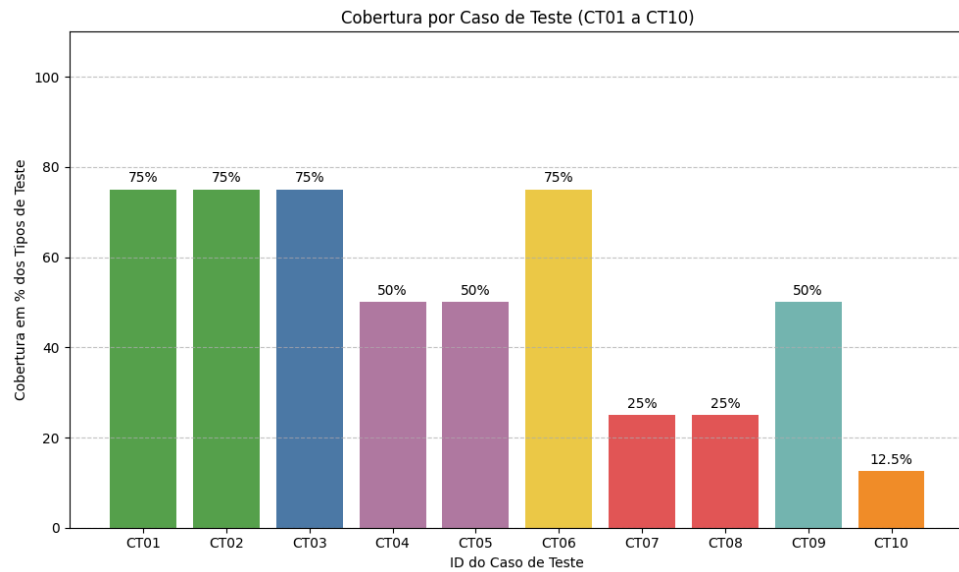
Nesse teste, como descrito anteriormente, encontramos falhas no teste por conta do WebDriver não ser executável por ser inválido no sistema operacional. Foi feito a realização do teste em diferente máquinas e utilizando o arquivo chromedriver.exe em todas, em todos essas tentativas tivemos o mesmo resultado.

8.4 Conclusões e lições aprendidas

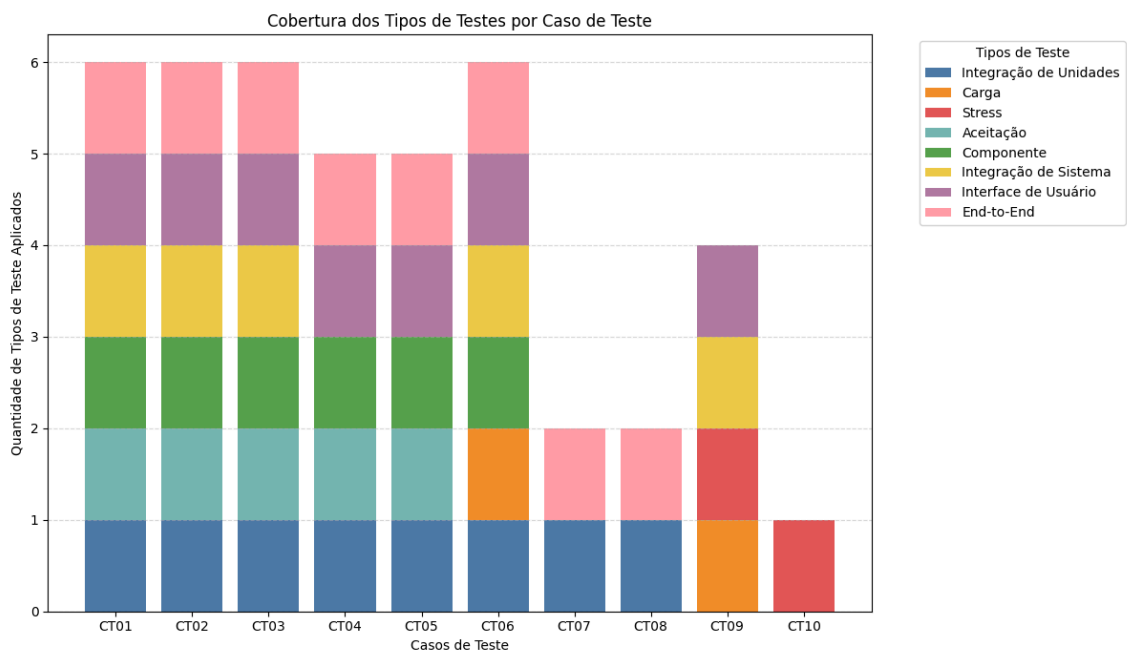
Os testes E2E não puderam ser executados com sucesso devido a uma falha na configuração do próprio ambiente de testes, com foco no momento da execução do ChromeDriver por alguma questão não identificada.

9 GRÁFICO RESULTADO DOS TESTES

Nós construímos dois gráficos com a ferramenta matplotlib, e para fins de resultado decidimos elaborar um gráfico que representa, em percentual, a cobertura dos casos de teste em relação a cada tipo de teste aplicado.



Abaixo teremos um gráfico que representa os diferentes tipos de testes que serão aplicados em cada funcionalidade. Cada barra corresponde a um caso de teste específico e é segmentada de acordo com os tipos de testes realizados.



10 CONCLUSÕES GERAIS

Com o nosso trabalho, podemos concluir que a automação de processos nas instituições de ensino é uma necessidade, para que possam aumentar a eficiência e reduzir os erros. Com a implementação do sistema, teremos um impacto significativo na produtividade dos controles de dados, beneficiando assim todo o corpo docente e discente.

Foi observado também, que para garantir a qualidade e a eficácia do sistema, é de extrema importância fazer testes para desenvolver o protótipo do software, tanto back-end quanto no front-end. Não só ajudou a criar um sistema mais completo e confiável, mas também criou um direcionamento para validar nosso projeto.

Portanto, o projeto não só contribuiu para a modernização do sistema educacional, mas também impactou positivamente o aprendizado e o desenvolvimento dos indivíduos envolvidos.