



FGA 0238 - Testes de Software - Turma: T01

Semestre: 2024.1

Nome: João Eduardo Pereira Rabelo

Matrícula: 180053299

Equipe: NeverMind

Atividade 4 – TDD

1 Funcionalidade

1.1 Identificação da Issue

Issue #189 - Erro ao editar fatura com data que já existe.

<https://gitlab.com/lappis-unb/projetos-energia/mec-energia/mec-energia-api/-/issues/189>

Fazendo um adendo professora, quando estava redigindo o documento e commitando os ciclos, verifiquei que alguém fez um merge com as correções da issue, não me permitindo efetuar o merge para tratar o problema da issue, contudo, caso seja do interesse da professora, segue abaixo meu link para acompanhar a hora e ordem dos commits para demonstrar que fiz de fato a correção.

https://gitlab.com/JoaoEduardoP/mec-energia-api/-/commits/189-Bug_fatura_invalida?ref_type=heads

1.2 Especificação

Nessa issue, o sistema permite editar a data de uma fatura de uma conta de luz, para outra data que também já tem uma fatura gravada, isso acaba criando 2 faturas na mesma data e quebra todas as UCs da universidade, não permitindo acessar a área de unidades consumidoras e suas respectivas faturas.



1.3 Descrição da Funcionalidade

Ao acessar uma Unidade Consumidora, pode-se visualizar todas as faturas daquela unidade, de todos os anos que estava em uso, podendo acessar as faturas e editar, conforme a necessidade de casa usuário.

1.4 Ciclos

Os ciclos se basearam em 4 etapas:

- Um primeiro ciclo para validar o método “Update” quando há a inserção de datas inválidas e teste para este caso.
- O segundo ciclo se baseia de ajustar o método “Update” para validar se tem alguma fatura no mês selecionado (Com exceção do mês da fatura que está sendo editada, pois poderia ocorrer do usuário querer apenas mudar um dado aleatório da fatura), além de testes para este caso.
- O terceiro ciclo, foi utilizado refatorando os testes e adicionado um para caso de testes de funcionamento correto de update de dados.
- O quarto ciclo, foi refatorado novamente os testes para uma cobertura correta e completa dos testes.



2 Execução

Antes dos Ciclos

```
----- coverage: platform linux, python 3.10.5-final-0 -----
Name                                                    Stmts   Miss Branch BrPart  Cover
-----
contracts/apps.py                                       4        0      0      0    100%
contracts/models.py                                    108      18      38      8     77%
contracts/serializers.py                               66       3       0      0     95%
contracts/services.py                                  56      42      24      0     18%
contracts/urls.py                                       7        0       0      0    100%
contracts/utils.py                                     17       8       2      0     47%
contracts/validators.py                                46      32       8      0     30%
contracts/views.py                                     198     121      32      1     36%
```

```
-----
TOTAL                                                    2859   1061     483     47     59%
Coverage HTML written to dir reports/cov

-- Generated html report: file:///home/dev/mec-energia-api/reports/index.html --
===== 120 passed in 11.97s =====
```

2.1 Primeiro Ciclo

- O primeiro ciclo foi usado para validar se a data inserida no método “Update” foi inserida de forma correta nas suas respectivas chamadas.

```
166 def test_update_energy_bill_invalid_date(self):
167     update_data = {
168         'id': 1,
169         'consumer_unit': self.university.id,
170         'date': 'invalid-date-format',
171     }
172
173     response = self.client.put('/api/energybills/1/', update_data, format='json')
174
175     assert response.status_code == status.HTTP_400_BAD_REQUEST
176     assert response.data == 'Invalid date, try this format: "yyyy-mm-dd".'
```

- Resultado dos testes:

```
----- coverage: platform linux, python 3.10.5-final-0 -----
Name                                                    Stmts   Miss Branch BrPart  Cover
-----
contracts/apps.py                                         4        0      0      0    100%
contracts/models.py                                     108      18     38      8     77%
contracts/serializers.py                                 66        3      0      0     95%
contracts/services.py                                    56      42     24      0     18%
contracts/urls.py                                         7        0      0      0    100%
contracts/utils.py                                       17        8      2      0     47%
contracts/validators.py                                 46      32      8      0     30%
contracts/views.py                                       205     127     32      1     35%
```

- Código da funcionalidade responsável pelo update:

```
185 def update(self, request, *args, **kwargs):
186     date_str = request.data.get('date')
187
188     try:
189         date = datetime.strptime(date_str, '%Y-%m-%d').date()
190     except ValueError:
191         return Response('Invalid date, try this format: "yyyy-mm-dd".', status=status.HTTP_400_BAD_REQUEST)
192
193     return super().update(request, *args, **kwargs)
```

2.2 Segundo Ciclo

- O segundo ciclo foi feito usado para validar se poderia ser alterado a data de uma fatura de energia para um mês que já existe uma fatura existente.

```
178 def test_update_energy_bill_duplicate(self):
179     existing_date = '2024-08-01'
180     update_data = {
181         'id': 1,
182         'consumer_unit': self.university.id,
183         'date': existing_date,
184     }
185
186     EnergyBill.objects.create(
187         id=2,
188         consumer_unit_id=self.university.id,
189         date=datetime.strptime(existing_date, '%Y-%m-%d').date(),
190         value=100.0,
191     )
192
193     response = self.client.put('/api/energybills/1/', update_data, format='json')
194
195     assert response.status_code == status.HTTP_400_BAD_REQUEST
196     assert response.data == 'There is already an energy bill this month and year for this consumer unit.'
```

- Resultado dos testes:

----- coverage: platform linux, python 3.10.5-final-0 -----					
Name	Stmts	Miss	Branch	BrPart	Cover
contracts/apps.py	4	0	0	0	100%
contracts/models.py	115	24	42	8	72%
contracts/serializers.py	66	3	0	0	95%
contracts/services.py	56	42	24	0	18%
contracts/urls.py	7	0	0	0	100%
contracts/utils.py	17	8	2	0	47%
contracts/validators.py	46	32	8	0	30%
contracts/views.py	209	131	34	1	34%

- Código da funcionalidade responsável pelo “Update”:

```
185 def update(self, request, *args, **kwargs):
186     id = request.data.get('id')
187     consumer_unit_id = request.data.get('consumer_unit')
188     date_str = request.data.get('date')
189
190     try:
191         date = datetime.strptime(date_str, '%Y-%m-%d').date()
192     except ValueError:
193         return Response('Invalid date, try this format: "yyyy-mm-dd".', status=status.HTTP_400_BAD_REQUEST)
194
195     if models.EnergyBill.check_energy_bill_id_month_year(id, consumer_unit_id, date):
196         return Response('There is already an energy bill this month and year for this consumer unit.', status=status.HTTP_400_BAD_REQUEST)
197
198     return super().update(request, *args, **kwargs)
```

2.3 Terceiro Ciclo

- O terceiro ciclo foi utilizado para refatorar os testes e incluir mais um que valida o teste de verificar se o update foi feito corretamente, conforme o teste abaixo:

```
261 def test_update_energy_bill_success(self):
262     # Crie uma unidade do consumidor
263     consumer_unit_data = {
264         'name': 'Faculdade do Gama',
265         'code': '111111111',
266         'created_on': '2022-10-02',
267         'is_active': True,
268         'university': self.university.id,
269         'total_installed_power': None
270     }
271     consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
272     created_consumer_unit = json.loads(consumer_unit_response.content)
273
274     assert consumer_unit_response.status_code == status.HTTP_201_CREATED
275
276     # Crie um contrato associado à unidade do consumidor
277     contract_data = {
278         'consumer_unit': created_consumer_unit['id'],
279         'start_date': '2023-01-01',
280         'end_date': '2023-12-31',
281         'supply_voltage': 100.00,
282         'distributor': self.distributor.id,
283     }
284     contract_response = self.client.post('/api/contracts/', contract_data, format='json')
285     created_contract = json.loads(contract_response.content)
286
287     assert contract_response.status_code == status.HTTP_201_CREATED
288
289     # Crie uma fatura de energia
290     data = {
291         'consumer_unit': created_consumer_unit['id'],
292         'contract': created_contract['id'],
293         'date': '2023-01-01',
294         'anotacoes': 'Some notes',
295     }
296     response = self.client.post('/api/energy-bills/', data, format='json')
297
298     assert response.status_code == status.HTTP_201_CREATED
299
300     # Atualize a fatura com uma nova data
301     update_data = {
302         'id': 1,
303         'consumer_unit': created_consumer_unit['id'],
304         'contract': created_contract['id'],
305         'date': '2023-02-01',
306         'anotacoes': 'Updated notes',
307     }
308
309     response = self.client.put('/api/energy-bills/1/', update_data, format='json')
310
311     assert response.status_code == status.HTTP_200_OK
```

- Resultado dos testes:

```
----- coverage: platform linux, python 3.10.5-final-0 -----
```

Name	Stmts	Miss	Branch	BrPart	Cover
contracts/apps.py	4	0	0	0	100%
contracts/models.py	115	24	42	8	72%
contracts/serializers.py	66	3	0	0	95%
contracts/services.py	56	42	24	0	18%
contracts/urls.py	7	0	0	0	100%
contracts/utils.py	17	8	2	0	47%
contracts/validators.py	46	32	8	0	30%
contracts/views.py	209	131	34	1	34%

- Código da funcionalidade responsável pelo “Update” permanece o mesmo.

2.4 Quarto Ciclo

- O quarto ciclo foi utilizado para refatorar todos os testes, para cobertura correta dos dados, conforme demonstradas no tópico 3 logo abaixo.
- Resultado dos testes:

```
-----
```

TOTAL	2877	1060	489	46	59%
-------	------	------	-----	----	-----

```
Coverage HTML written to dir reports/cov

-- Generated html report: file:///home/dev/mec-energia-api/reports/index.html --
===== 124 passed in 12.58s =====
```

- Código da funcionalidade responsável pelo “Update” permanece o mesmo.

3 Código Fonte Testes

Testes Criados para ambos os métodos no arquivo “test_EnergyBillViewSet_create.py”

```
166 def test_update_energy_bill_invalid_date(self):
167     # Crie uma unidade do consumidor
168     consumer_unit_data = {
169         'name': 'Faculdade do Gama',
170         'code': '111111111',
171         'created_on': '2022-10-02',
172         'is_active': True,
173         'university': self.university.id,
174         'total_installed_power': None
175     }
176     consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
177     created_consumer_unit = json.loads(consumer_unit_response.content)
178
179     assert consumer_unit_response.status_code == status.HTTP_201_CREATED
180
181     # Crie um contrato associado à unidade do consumidor
182     contract_data = {
183         'consumer_unit': created_consumer_unit['id'],
184         'start_date': '2023-01-01',
185         'end_date': '2023-12-31',
186         'supply_voltage': 100.00,
187         'distributor': self.distributor.id,
188     }
189     contract_response = self.client.post('/api/contracts/', contract_data, format='json')
190     created_contract = json.loads(contract_response.content)
191
192     assert contract_response.status_code == status.HTTP_201_CREATED
193
194     # Tente atualizar uma fatura de energia com uma data inválida
195     update_data = {
196         'id': 1,
197         'consumer_unit': created_consumer_unit['id'],
198         'contract': created_contract['id'],
199         'date': 'invalid-date-format',
200         'anotacoes': 'Updated notes',
201     }
202
203     response = self.client.put('/api/energy-bills/1/', update_data, format='json')
204
205     assert response.status_code == status.HTTP_400_BAD_REQUEST
206     assert response.data == 'Invalid date, try this format: "yyyy-mm-dd".'
```



```
208 def test_update_energy_bill_duplicate(self):
209     consumer_unit_data = {
210         'name': 'Faculdade do Gama',
211         'code': '111111111',
212         'created_on': '2022-10-02',
213         'is_active': True,
214         'university': self.university.id,
215         'total_installed_power': None
216     }
217     consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
218     created_consumer_unit = json.loads(consumer_unit_response.content)
219
220     assert consumer_unit_response.status_code == status.HTTP_201_CREATED
221
222     contract_data = {
223         'consumer_unit': created_consumer_unit['id'],
224         'start_date': '2023-01-01',
225         'end_date': '2023-12-31',
226         'supply_voltage': 100.00,
227         'distributor': self.distributor.id,
228     }
229     contract_response = self.client.post('/api/contracts/', contract_data, format='json')
230     created_contract = json.loads(contract_response.content)
231
232     assert contract_response.status_code == status.HTTP_201_CREATED
233
234     data = {
235         'id': 1,
236         'consumer_unit': created_consumer_unit['id'],
237         'contract': created_contract['id'],
238         'date': '2023-01-01',
239         'anotacoes': 'Some notes',
240     }
241     response = self.client.post('/api/energy-bills/', data, format='json')
242
243     assert response.status_code == status.HTTP_201_CREATED
244
245     update_data = {
246         'id': 2,
247         'consumer_unit': created_consumer_unit['id'],
248         'contract': created_contract['id'],
249         'date': '2023-01-01',
250         'anotacoes': 'Create notes',
251     }
252
253     response = self.client.put('/api/energy-bills/1/', update_data, format='json')
254
255     assert response.status_code == status.HTTP_400_BAD_REQUEST
256     assert response.data == 'There is already an energy bill this month and year for this consumer unit.'
```

```
258 def test_update_energy_bill_success(self):
259     # Crie uma unidade do consumidor
260     consumer_unit_data = {
261         'name': 'Faculdade do Gama',
262         'code': '111111111',
263         'created_on': '2022-10-02',
264         'is_active': True,
265         'university': self.university.id,
266         'total_installed_power': None
267     }
268     consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
269     created_consumer_unit = json.loads(consumer_unit_response.content)
270
271     assert consumer_unit_response.status_code == status.HTTP_201_CREATED
272
273     # Crie um contrato associado à unidade do consumidor
274     contract_data = {
275         'consumer_unit': created_consumer_unit['id'],
276         'start_date': '2023-01-01',
277         'end_date': '2023-12-31',
278         'supply_voltage': 100.00,
279         'distributor': self.distributor.id,
280     }
281     contract_response = self.client.post('/api/contracts/', contract_data, format='json')
282     created_contract = json.loads(contract_response.content)
283
284     assert contract_response.status_code == status.HTTP_201_CREATED
285
286     # Crie uma fatura de energia
287     data = {
288         'consumer_unit': created_consumer_unit['id'],
289         'contract': created_contract['id'],
290         'date': '2023-01-01',
291         'anotacoes': 'Some notes',
292     }
293     response = self.client.post('/api/energy-bills/', data, format='json')
294
295     assert response.status_code == status.HTTP_201_CREATED
296
297     # Atualize a fatura com uma nova data
298     update_data = {
299         'id': 1,
300         'consumer_unit': created_consumer_unit['id'],
301         'contract': created_contract['id'],
302         'date': '2023-02-01',
303         'anotacoes': 'Updated notes',
304     }
305
306     response = self.client.put('/api/energy-bills/1/', update_data, format='json')
307
308     assert response.status_code == status.HTTP_200_OK
```

Segue abaixo o repositório do Fork em que foi feito o trabalho:

https://gitlab.com/JoaoEduardoP/mec-energia-api/-/tree/189-Bug_fatura_invalida?ref_type=heads

4 Resultado Final Execução Testes

- Dados antes das implementações:

```
-----  
TOTAL                                                    2859   1061   483   47   59%  
Coverage HTML written to dir reports/cov  
  
-- Generated html report: file:///home/dev/mec-energia-api/reports/index.html --  
===== 120 passed in 11.97s =====
```

```
----- coverage: platform linux, python 3.10.5-final-0 -----  
Name                                                    Stmts    Miss  Branch BrPart  Cover  
-----  
contracts/apps.py                                         4         0      0      0   100%  
contracts/models.py                                     108        18      38      8    77%  
contracts/serializers.py                                 66         3       0      0    95%  
contracts/services.py                                    56        42      24      0    18%  
contracts/urls.py                                         7         0       0      0   100%  
contracts/utils.py                                       17         8       2      0    47%  
contracts/validators.py                                  46        32       8      0    30%  
contracts/views.py                                       198       121      32      1    36%
```

- Dados após as implementações:

```
-----  
TOTAL                                                    2877   1060   489   46   59%  
Coverage HTML written to dir reports/cov  
  
-- Generated html report: file:///home/dev/mec-energia-api/reports/index.html --  
===== 124 passed in 12.58s =====
```

```
----- coverage: platform linux, python 3.10.5-final-0 -----  
Name                                                    Stmts    Miss  Branch BrPart  Cover  
-----  
contracts/apps.py                                         4         0      0      0   100%  
contracts/models.py                                     115        24      42      8    72%  
contracts/serializers.py                                 66         3       0      0    95%  
contracts/services.py                                    56        42      24      0    18%  
contracts/urls.py                                         7         0       0      0   100%  
contracts/utils.py                                       17         8       2      0    47%  
contracts/validators.py                                  46        32       8      0    30%  
contracts/views.py                                       209       131      34      1    34%
```

5 Código Fonte da Funcionalidade Implementada

Método “Update” Criado no arquivo “views.py”:

```
185 def update(self, request, *args, **kwargs):
186     id = request.data.get('id')
187     consumer_unit_id = request.data.get('consumer_unit')
188     date_str = request.data.get('date')
189
190     try:
191         date = datetime.strptime(date_str, '%Y-%m-%d').date()
192     except ValueError:
193         return Response('Invalid date, try this format: "yyyy-mm-dd".', status=status.HTTP_400_BAD_REQUEST)
194
195     if models.EnergyBill.check_energy_bill_id_month_year(id, consumer_unit_id, date):
196         return Response('There is already an energy bill this month and year for this consumer unit.', status=status.HTTP_400_BAD_REQUEST)
197
198     return super().update(request, *args, **kwargs)
```

Método “check_energy_bill_id_month_year” Criado no arquivo “models.py”:

```
258 def check_energy_bill_id_month_year(id_energy_bill, consumer_unit_id, date):
259     has_already_energy_bill = EnergyBill.objects.filter(
260         consumer_unit=consumer_unit_id,
261         date__year=date.year,
262         date__month=date.month)
263
264     if has_already_energy_bill.exists():
265         if has_already_energy_bill.first().id == id_energy_bill:
266             return False
267         else:
268             return True
269
270     return False
```

Segue abaixo o repositório do Fork em que foi feito o trabalho:

https://gitlab.com/JoaoEduardoP/mec-energia-api/-/tree/189-Bug_fatura_invalida?ref_type=heads

6 Pull Request

Conforme detalhei no tópico 1.1, não fiz o Pull Request pois não conclui antes de uma correção ter sido feita no merge, ficando sem necessidade de 2 merge requests para um Bug já resolvido, contudo, conforme afirmei logo acima, segue abaixo os commits feitos em ciclos para correção do mesmo:

https://gitlab.com/JoaoEduardoP/mec-energia-api/-/commits/189-Bug_fatura_invalida?ref_type=heads

7 Conclusão

O desenvolvimento e a implementação da solução para a issue #189 no projeto "MEC Energia API" permitiram uma análise aprofundada dos ciclos de Test-Driven Development (TDD), focando na correção de um bug crítico relacionado à edição de faturas. Apesar de não ter sido possível realizar o merge final devido à intervenção de outro desenvolvedor, todos os ciclos de testes foram documentados e os commits registrados demonstram a eficácia das correções propostas. A experiência prática adquirida ao longo deste processo reforçou a importância de uma abordagem estruturada e iterativa na validação de software, garantindo que cada alteração contribua para a robustez e a integridade do sistema. Este trabalho evidencia a capacidade de identificar, corrigir e prevenir erros no código, além de destacar a relevância da comunicação e colaboração em ambientes de desenvolvimento compartilhado.