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.

```python
    def test_update_energy_bill_invalid_date(self):
        update_data = {
            'id': 1,
            'consumer_unit': self.university.id,
            'date': 'invalid-date-format',
        }

        response = self.client.put('/api/energybills/1/', update_data, format='json')

        assert response.status_code == status.HTTP_400_BAD_REQUEST
        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:

```python
    def update(self, request, *args, **kwargs):
        date_str = request.data.get('date')

        try:
            date = datetime.strptime(date_str, '%Y-%m-%d').date()
        except ValueError:
            return Response('Invalid date, try this format: "yyyy-mm-dd".', status=status.HTTP_400_BAD_REQUEST)

        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.

```python
    def test_update_energy_bill_duplicate(self):
        existing_date = '2024-08-01'
        update_data = {
            'id': 1,
            'consumer_unit': self.university.id,
            'date': existing_date,
        }

        EnergyBill.objects.create(
            id=2,
            consumer_unit_id=self.university.id,
            date=datetime.strptime(existing_date, '%Y-%m-%d').date(),
            value=100.0,
        )

        response = self.client.put('/api/energybills/1/', update_data, format='json')

        assert response.status_code == status.HTTP_400_BAD_REQUEST
        assert response.data == 'There is already an energy bill this month and year for this consumer unit.'
```

- Resutado 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”:

```python
    def update(self, request, *args, **kwargs):
        id = request.data.get('id')
        consumer_unit_id = request.data.get('consumer_unit')
        date_str = request.data.get('date')

        try:
            date = datetime.strptime(date_str, '%Y-%m-%d').date()
        except ValueError:
            return Response('Invalid date, try this format: "yyyy-mm-dd".', status=status.HTTP_400_BAD_REQUEST)

        if models.EnergyBill.check_energy_bill_id_month_year(id, consumer_unit_id, date):
            return Response('There is already an energy bill this month and year for this consumer unit.', status=status.HTTP_400_BAD_REQUEST)

        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:

```python
def test_update_energy_bill_success(self):
    # Crie uma unidade do consumidor
    consumer_unit_data = {
        'name': 'Faculdade do Gama',
        'code': '111111111',
        'created_on': '2022-10-02',
        'is_active': True,
        'university': self.university.id,
        'total_installed_power': None
    }
    consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
    created_consumer_unit = json.loads(consumer_unit_response.content)

    assert consumer_unit_response.status_code == status.HTTP_201_CREATED

    # Crie um contrato associado à unidade do consumidor
    contract_data = {
        'consumer_unit': created_consumer_unit['id'],
        'start_date': '2023-01-01',
        'end_date': '2023-12-31',
        'supply_voltage': 100.00,
        'distributor': self.distributor.id,
    }
    contract_response = self.client.post('/api/contracts/', contract_data, format='json')
    created_contract = json.loads(contract_response.content)

    assert contract_response.status_code == status.HTTP_201_CREATED

    # Crie uma fatura de energia
    data = {
        'consumer_unit': created_consumer_unit['id'],
        'contract': created_contract['id'],
        'date': '2023-01-01',
        'anotacoes': 'Some notes',
    }
    response = self.client.post('/api/energy-bills/', data, format='json')

    assert response.status_code == status.HTTP_201_CREATED

    # Atualize a fatura com uma nova data
    update_data = {
        'id': 1,
        'consumer_unit': created_consumer_unit['id'],
        'contract': created_contract['id'],
        'date': '2023-02-01',
        'anotacoes': 'Updated notes',
    }

    response = self.client.put('/api/energy-bills/1/', update_data, format='json')

    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, coforme 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”

```python
    def test_update_energy_bill_invalid_date(self):
        # Crie uma unidade do consumidor
        consumer_unit_data = {
            'name': 'Faculdade do Gama',
            'code': '111111111',
            'created_on': '2022-10-02',
            'is_active': True,
            'university': self.university.id,
            'total_installed_power': None
        }
        consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
        created_consumer_unit = json.loads(consumer_unit_response.content)

        assert consumer_unit_response.status_code == status.HTTP_201_CREATED

        # Crie um contrato associado à unidade do consumidor
        contract_data = {
            'consumer_unit': created_consumer_unit['id'],
            'start_date': '2023-01-01',
            'end_date': '2023-12-31',
            'supply_voltage': 100.00,
            'distributor': self.distributor.id,
        }
        contract_response = self.client.post('/api/contracts/', contract_data, format='json')
        created_contract = json.loads(contract_response.content)

        assert contract_response.status_code == status.HTTP_201_CREATED

        # Tente atualizar uma fatura de energia com uma data inválida
        update_data = {
            'id': 1,
            'consumer_unit': created_consumer_unit['id'],
            'contract': created_contract['id'],
            'date': 'invalid-date-format',
            'anotacoes': 'Updated notes',
        }

        response = self.client.put('/api/energy-bills/1/', update_data, format='json')

        assert response.status_code == status.HTTP_400_BAD_REQUEST
        assert response.data == 'Invalid date, try this format: "yyyy-mm-dd".'

```

```python
    def test_update_energy_bill_duplicate(self):
        consumer_unit_data = {
            'name': 'Faculdade do Gama',
            'code': '111111111',
            'created_on': '2022-10-02',
            'is_active': True,
            'university': self.university.id,
            'total_installed_power': None
        }
        consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
        created_consumer_unit = json.loads(consumer_unit_response.content)

        assert consumer_unit_response.status_code == status.HTTP_201_CREATED

        contract_data = {
            'consumer_unit': created_consumer_unit['id'],
            'start_date': '2023-01-01',
            'end_date': '2023-12-31',
            'supply_voltage': 100.00,
            'distributor': self.distributor.id,
        }
        contract_response = self.client.post('/api/contracts/', contract_data, format='json')
        created_contract = json.loads(contract_response.content)

        assert contract_response.status_code == status.HTTP_201_CREATED

        data = {
            'id': 1,
            'consumer_unit': created_consumer_unit['id'],
            'contract': created_contract['id'],
            'date': '2023-01-01',
            'anotacoes': 'Some notes',
        }
        response = self.client.post('/api/energy-bills/', data, format='json')

        assert response.status_code == status.HTTP_201_CREATED

        update_data = {
            'id': 2,
            'consumer_unit': created_consumer_unit['id'],
            'contract': created_contract['id'],
            'date': '2023-01-01',
            'anotacoes': 'Create notes',
        }

        response = self.client.put('/api/energy-bills/1/', update_data, format='json')

        assert response.status_code == status.HTTP_400_BAD_REQUEST
        assert response.data == 'There is already an energy bill this month and year for this consumer unit.'
```

```python
    def test_update_energy_bill_success(self):
        # Crie uma unidade do consumidor
        consumer_unit_data = {
            'name': 'Faculdade do Gama',
            'code': '111111111',
            'created_on': '2022-10-02',
            'is_active': True,
            'university': self.university.id,
            'total_installed_power': None
        }
        consumer_unit_response = self.client.post('/api/consumer-units/', consumer_unit_data, format='json')
        created_consumer_unit = json.loads(consumer_unit_response.content)

        assert consumer_unit_response.status_code == status.HTTP_201_CREATED

        # Crie um contrato associado à unidade do consumidor
        contract_data = {
            'consumer_unit': created_consumer_unit['id'],
            'start_date': '2023-01-01',
            'end_date': '2023-12-31',
            'supply_voltage': 100.00,
            'distributor': self.distributor.id,
        }
        contract_response = self.client.post('/api/contracts/', contract_data, format='json')
        created_contract = json.loads(contract_response.content)

        assert contract_response.status_code == status.HTTP_201_CREATED

        # Crie uma fatura de energia
        data = {
            'consumer_unit': created_consumer_unit['id'],
            'contract': created_contract['id'],
            'date': '2023-01-01',
            'anotacoes': 'Some notes',
        }
        response = self.client.post('/api/energy-bills/', data, format='json')

        assert response.status_code == status.HTTP_201_CREATED

        # Atualize a fatura com uma nova data
        update_data = {
            'id': 1,
            'consumer_unit': created_consumer_unit['id'],
            'contract': created_contract['id'],
            'date': '2023-02-01',
            'anotacoes': 'Updated notes',
        }

        response = self.client.put('/api/energy-bills/1/', update_data, format='json')

        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:

```
---------- coverage: platform linux, python 3.10.5-final-0 -----------
Name                                                                                                  Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------------------------------------------------------------------------------------
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:

```
---------- coverage: platform linux, python 3.10.5-final-0 -----------
Name                                                                                                  Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------------------------------------------------------------------------------------
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.