Gabriel Negri – 19.1.4976

Projeto API + Construção de Teste

Ouro Preto

2021

Gabriel Negri

Projeto API + Construção de Testes

Trabalho prático de cunho acadêmico sobre a construção de um sistema, utilizando o projeto de uma API e o desenvolvimento de testes, referente à disciplina de Engenharia de Software (BCC322), com o objetivo de explicar e apresentar o tema abordado, baseado nos estudos e conhecimento que a disciplina proporciona.

Sprint 1

Projeto API

REQUISITOS MÍNIMOS DE RESOLUÇÃO:

- Gerenciar um sistema amplo para desenvolvimento com base em análises e testes.
- Coesão de sistemas, interligações e fluxos contínuos.
- Sistema hierárquico com ligações que fomentam as classes filhas.

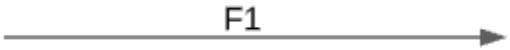
FUNCIONALIDADES EXIGIDAS:

- Modelagem: Modelo CRUD e sistemas singulares associados aos fluxos.
- Fluxos: CRUD e relatório dos fluxos feitos. Cálculo feito sobre cada um.
- Ligação de fluxos e sistemas: CRUD e a melhor conexão possível entre ambos.

CASOS DE USO:

Segue abaixo, os possíveis casos de uso da API em suas modelagens, feito através de um esboço de anagrama:

1. Fluxo sem origem e sem destino final:



Codificação ao caso de uso:

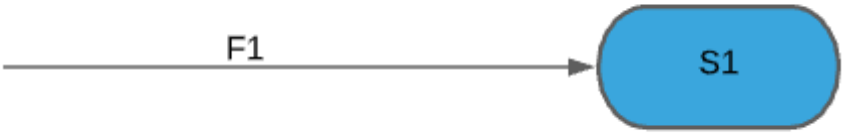
Flow f1;

2. Sistema isolado sem fluxo:

Codificação ao caso de uso:

System s1;

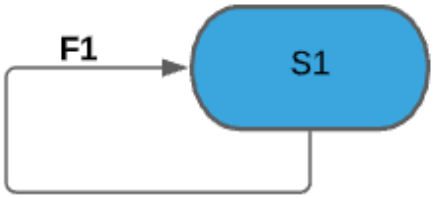
3. Um fluxo sem origem conectado a um sistema:



Codificação ao caso de uso:

Flow f1;
System s1;
s1.inflow(f1); //Conexao de f1 para s1

4. Um circuito de um fluxo conectado a um sistema:



Codificação ao caso de uso:

Flow f1;

System s1;

s1.inflow(f1); //Conexao de f1 para s1

s1.outflow(f1); //Saida de s1 para f1

5. Fluxo conectado a um sistema em sua origem, porém, sem destino final:

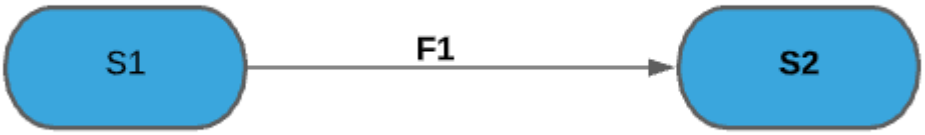
Codificação ao caso de uso:

Flow f1;

System s1;

s1.outflow(f1);

6. Fluxo conectado a dois sistemas, sendo um na origem e outro em seu destino final:



Codificação ao caso de uso:

Flow f1;

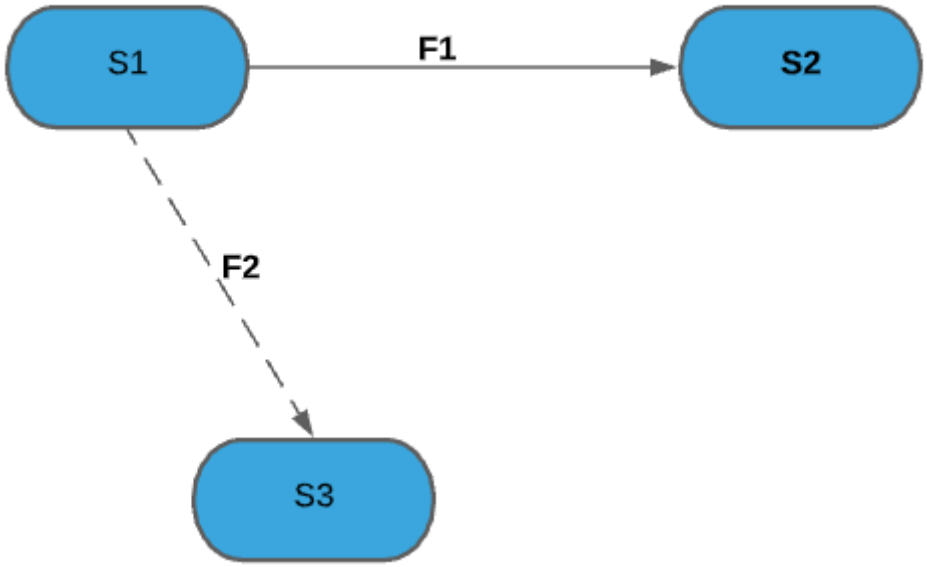
System s1;

System s2;

s1.outflow(f1);

s2.inflow(f1);

7. Sistema que liga aos demais por diversos fluxos.
   Obs: É possível perceber que um sistema pode se conectar a outro por um fluxo de pouca conexão, ou seja, a relação entre ambos é fraca, não há muita ligação entre os sistemas, (é possível perceber isso entre a conexão de S1 e S3).



Codificação ao caso de uso:

Flow f1;

Flow f2;

System s1;

System s2;

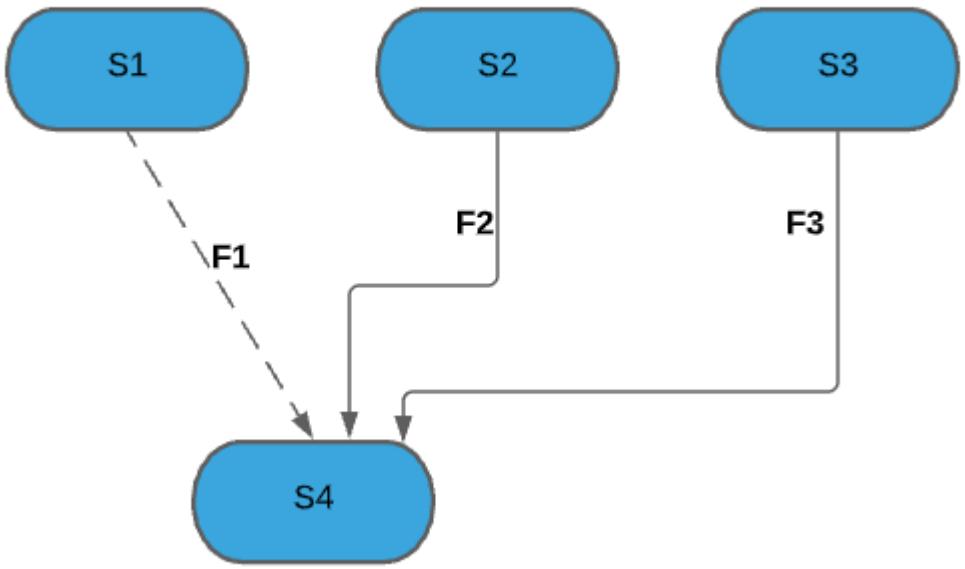
System s3;

s1.outflow(f1);

s2.inflow(f1);

s3.inflow(f2);

8. Ligação de sistemas para um único ou mais de um:



Codificação ao caso de uso:

System s1, s2, s3, s4;

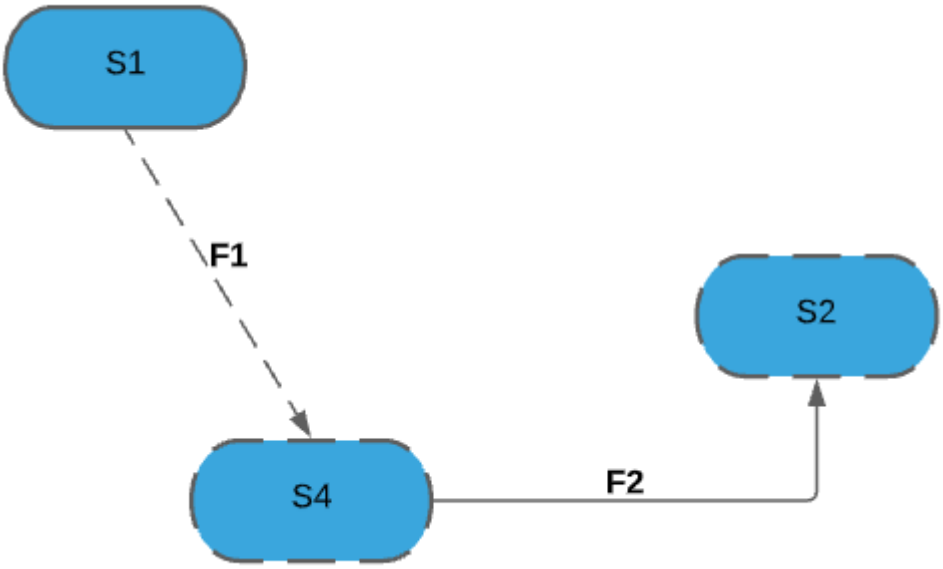
Flow f1, f2, f3;

f1.connect(s1, s4);

f2.connect(s2, s4);

f4.connect(s3, s4);

9. Sistema com fluxos de entrada e saída entre sistemas:



Codificação ao caso de uso:

System s1,s2,s4;

Flow f1,f2;

f1.connect(s1, s2);

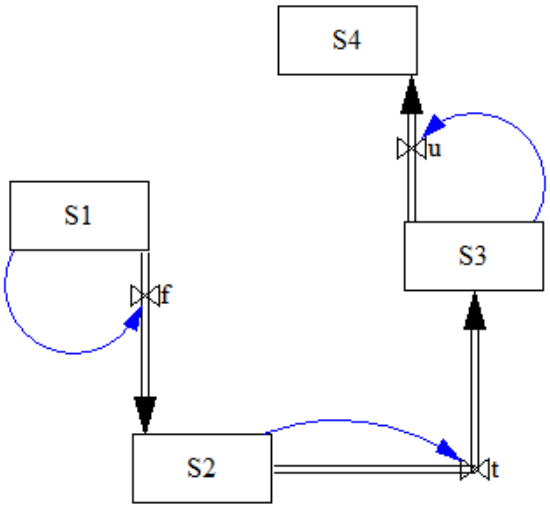
f2.connect(s4, s2);

Cenário de Testes:

É importante afirmar que o cliente gostaria de que os fluxos passassem de uma forma eficiente e com o menor custo possível. Com base nas análises feitas pelo mesmo, foram feitos 3 testes:

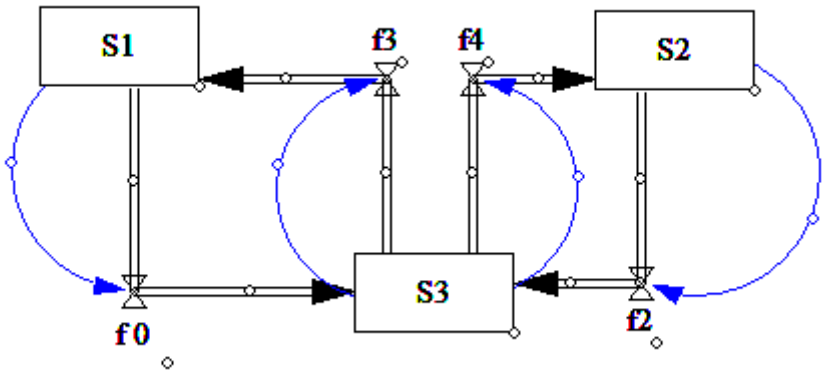
Teste 1:

```
//Representado como o modelo que efetua conexão
// em série dos sistemas conectados desde System 1 até System 4.

Model m;
Flow f1, f2, f3;
System s1, s2, s3, s4;

System s[ ] = {s1, s2, s3, s4};
Flow f[ ] = {f1, f2, f3};

m.addSystem(s);
m.addFlow(f);

f1.connect(s1, s2);
f2.connect(s2, s3);
f3.connect(s3, s4);

f4.setTarget(s4);

m.run();
m.report();
```



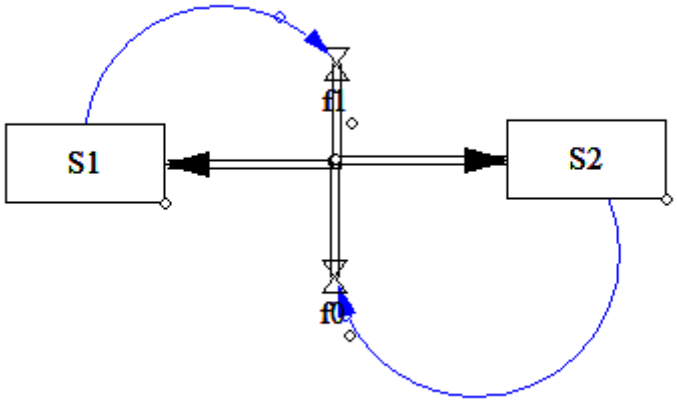
Teste 2:

```
// Modelo que provém de uma conexão aonde 2 sistemas se conectam a um mesmo
// com entradas e saídas, cada sistema origem tem um par de conexão e
// o sistema que recebe as conexões há 2 fluxos de entrada e 2 de saídas

Model m;
Flow f1, f2, f3, f4;
System s1, s2, s3;

System s[] = {s1, s2, s3};
Flow f[] = {f1, f2, f3, f4};

m.addSystem(s);
m.addFlow(f);

f1.connect(s1, s3);
s3.outflow(f2);
f2.connect(s3, s1);
s3.outflow(f3)
f3.connect(s3, s2);

f1.setTarget(s3);
f2.setTarget(s3);
f3.setTarget(s1);
f4.setTarget(s1);

m.run();
m.report();
```

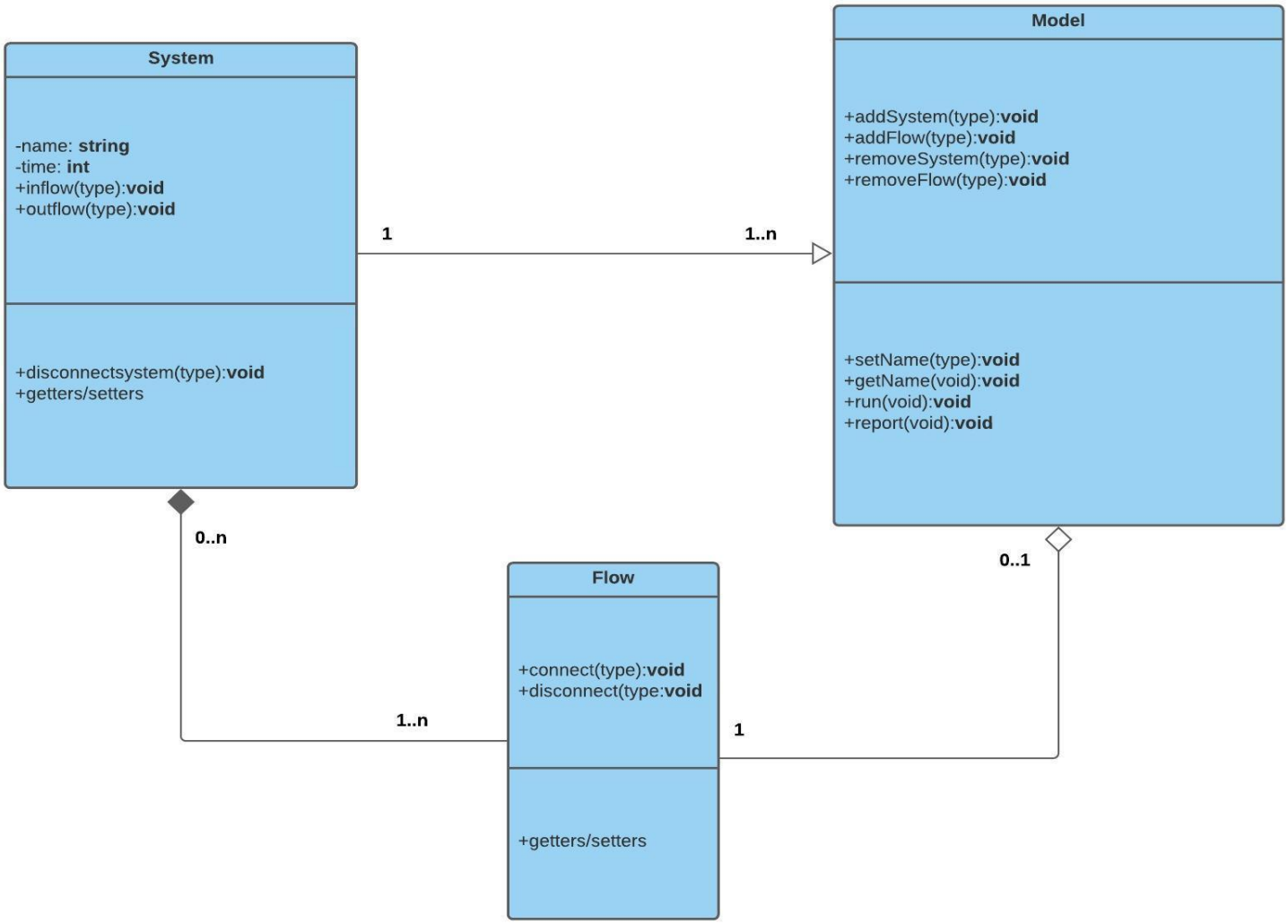


Teste 3:

```
//Modelo que terá uma conexão de 2 sistemas, 2 fluxos de entrada e 1 de saídas

Model m;
Flow f1, f2, f3;
System s1, s2;

System s[] = {s1, s2};
Flow f[] = {f1, f2, f3};

m.addSystem(s);
m.addFlow(f);

f1.connect(s1, s2);
f2.connect(s1, s2);
s1.outflow(f1, f2);

f3.connect(s2, s1);
s2.inflow(f3);

f3.setTarget(s1);

m.run();
m.report();
```


S1
S2
f1
f0

Modelo UML:


System
-name: string
-time: int
+inflow(type):void
+outflow(type):void
+disconnectsystem(type):void
+getters/setters
Model
+addSystem(type):void
+addFlow(type):void
+removeSystem(type):void
+removeFlow(type):void
+setName(type):void
+getName(void):void
+run(void):void
+report(void):void
Flow
+connect(type):void
+disconnect(type:void
+getters/setters
1
1..n
0..n
1..n
1
0..1