

API Rest - Sistema de gestão de recompensas para tarefas de crianças

Centro Universitário Adventista de São Paulo
Campus Hortolândia

Sistemas de Informação
Disciplina: Modelagem de Sistemas

Professor:

Eduardo Mendes Oliveira

Equipe do Projeto:

Adriano Martins
Matheus Julião
Henrique Albuquerque
Jackson Freitas

Hortolândia, 20 de Junho de 2023

Sumário

1. Introdução
2. Equipe do Projeto
3. Diagramas de Casos de Uso
 - 3.1. Atores
 - 3.2. Casos de Uso
4. Diagramas de Classe
 - 4.1. Diagrama de Classe 1: [Sponsor]
 - 4.2. Diagrama de Classe 2: [Child]
 - 4.3. Diagrama de Classe 3: [SponsorChild]
 - 4.4. Diagrama de Classe 4: [Task]
 - 4.5. Diagrama de Classe 5: [TotalMonthlyAmount]
 - 4.6. Diagrama de Classe 6: [Penalty]
 - 4.7. Diagrama de Classe 7: [Bonus]
5. Diagramas de Sequência
 - 5.1. Diagrama de sequência 1: [Cadastra-se]
 - 5.2. Diagrama de sequência 2: [Cadastrar criança]
 - 5.3. Diagrama de sequência 3: [Cadastrar bônus]
 - 5.4. Diagrama de sequência 4: [Cadastrar penalidade]
 - 5.5. Diagrama de sequência 5: [Cadastrar remuneração]
 - 5.6. Diagrama de sequência 6: [Gerenciar valor mensal]
 - 5.7. Diagrama de sequência 7: [Gerenciar tarefas]
 - 5.8. Diagrama de sequência 8: [Consultar tarefa]
 - 5.9. Diagrama de sequência 9: [Consultar recompensa]
6. API REST Desenvolvida
 - 6.1. Objetivos da API REST desenvolvida
 - 6.2. Tecnologias utilizadas
 - 6.3. Arquitetura da API Rest
 - 6.4. Endpoints da API REST
 - 6.5. Formatos de dados
 - 6.7. Testes e documentação da API
 - 6.8. Considerações de segurança
7. Conclusão

1. Introdução

Sobre o projeto:

O projeto tem por objetivo criar uma API REST que permita a gestão de recompensas para tarefas de crianças, proporcionando aos usuários uma ferramenta intuitiva para organização e acompanhamento de suas atividades diárias.

A gestão das tarefas foi realizada utilizando o JIRA Software, seguindo a metodologia ágil SCRUM. Todos os membros do grupo estão vinculados ao board do JIRA, que foi utilizado para planejamento das sprints e realização de dailys durante os encontros em sala de aula.

O versionamento do código foi feito utilizando o Git, garantindo um controle adequado das versões e permitindo a colaboração dos membros do grupo.

Além disso, foram elaborados os diagramas exigidos para o projeto, incluindo o Diagrama de Caso de Uso, o Diagrama de Classe e o Diagrama de Sequência para cada Caso de Uso. A implementação do código do diagrama de classe foi realizada, utilizando o framework do Java, Spring Boot, que atende ao paradigma da Programação Orientada a Objetos. Também foram implementados testes unitários para garantir a qualidade e cobertura completa do código.

O repositório do projeto foi criado no GitHub, onde todos os membros do grupo estão vinculados para colaboração e compartilhamento do código-fonte e documentações.

O projeto "API REST - Sistema de Gerenciamento de Tarefas" representa um esforço conjunto do grupo de alunos, visando aplicar os conhecimentos adquiridos na disciplina de Modelagem de Sistemas, bem como desenvolver habilidades em trabalho colaborativo, gestão de projetos e utilização de metodologias ágeis.

2. Equipe do Projeto

A equipe é composta por 4 alunos, sendo:

- **Matheus Julião:** parte integrante na documentação de diagramas de desenvolvimentos da API em Spring Boot;
- **Adriano Martins:** parte integrante na elaboração de diagramas, documentação e testes unitários da API;
- **Henrique Albuquerque:** parte integrante na elaboração de diagramas, documentação e testes unitários da API;
- **Jackson Freitas:** parte integrante na elaboração de diagramas, e documentação do projeto.

3. Diagramas de Casos de Uso

3.1. Atores

Os atores do sistema (usuários que interagem com o sistema) são:

a) Sponsor: aquele responsável por gerenciar tarefas e valores, bem como crianças;

b) Child: aquela responsável por cumprir as atividades propostas podendo receber recompensas.

3.2. Casos de Uso

Os possíveis casos de uso (função que um sistema desempenha para alcançar a meta do usuário) para o sistema de controle de tarefas são:

a) Cadastrar-se na plataforma:

Inicialmente, cabe ao Sponsor fazer o seu próprio cadastro na plataforma para que sejam iniciadas as atividades de gerenciamento do sistema.

b) Gerenciar criança:

Após cadastro e login na plataforma, o Sponsor pode fazer o gerenciamento de crianças na plataforma.

c) Gerenciar Valor Mensal Total:

Após cadastro e login, o Sponsor pode fazer o gerenciamento dos valores na plataforma.

d) Atribuir bônus:

O Sponsor pode fazer a atribuição de bônus a uma criança, somando ao valor total mensal.

e) Atribuir penalização:

O Sponsor pode fazer a atribuição de penas a uma criança, subtraindo ao valor total mensal.

f) Gerenciar tarefas:

O Sponsor pode fazer o gerenciamento de tarefas a serem realizadas por uma criança, bem como atribuir pesos a estas atividades.

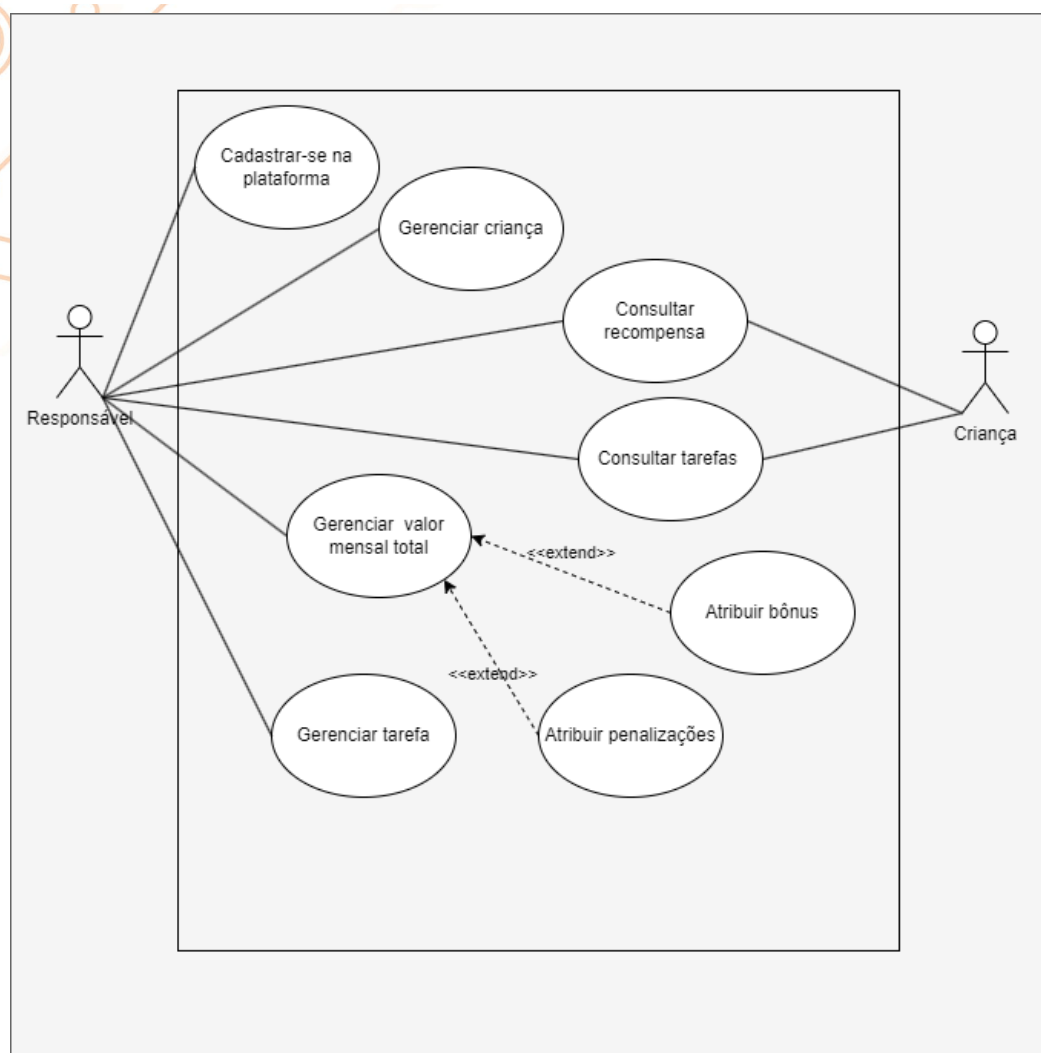
g) Consultar recompensa

Tanto a criança quanto o Sponsor poderão consultar o valor total mensal. A criança poderá fazer a consulta de recompensas já ganhas no sistema.

h) Consultar tarefas

Tanto a Criança quanto o Sponsor podem consultar tarefas criadas no sistema.

Figura 1 - Diagrama de Casos de Uso



4. Diagramas de Classe

Os diagramas de classe são os objetos que compõem o sistema. Eles também fornecem informações sobre o que o objeto faz e quais os serviços fornecidos e seus relacionamentos.

4.1. Diagrama de Classe 1: [Sponsor]

O diagrama de classe "Sponsor" representa uma classe chamada "Sponsor" que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem os atributos name, email, e password, e métodos createSponsor, updateSponsor, deleteSponsor, listSponsor, loginApp;

4.2. Diagrama de Classe 2: [Child]

O diagrama de classe "Child" representa uma classe chamada "Child" que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem os atributos name, age, nickname, password, userCreator e métodos createSponsor, updateChild, deleteChild, listChild e loginApp;

4.3. Diagrama de Classe 3: [SponsorChild]

O diagrama de classe "SponsorChild" representa o relacionamento "SponsorChild" que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem os atributos idSponsor e idChild.

4.4. Diagrama de Classe 4: [Task]

O diagrama de classe "Task" representa uma classe chamada "Task" e também uma tabela no banco de dados que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem os atributos name,

description, weight, value, isComplete e métodos createTask, updateTask, deleteTask, listAllTask, listTask, calculateTask e TaskCompleted;

4.5. Diagrama de Classe 5: [TotalMonthlyAmount]

O diagrama de classe "TotalMonthlyAmount" representa uma classe chamada "TotalMonthlyAmount" e também uma tabela no banco de dados que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem os atributos description, total, remainder e métodos createTotal, deleteTotal e showTotal;

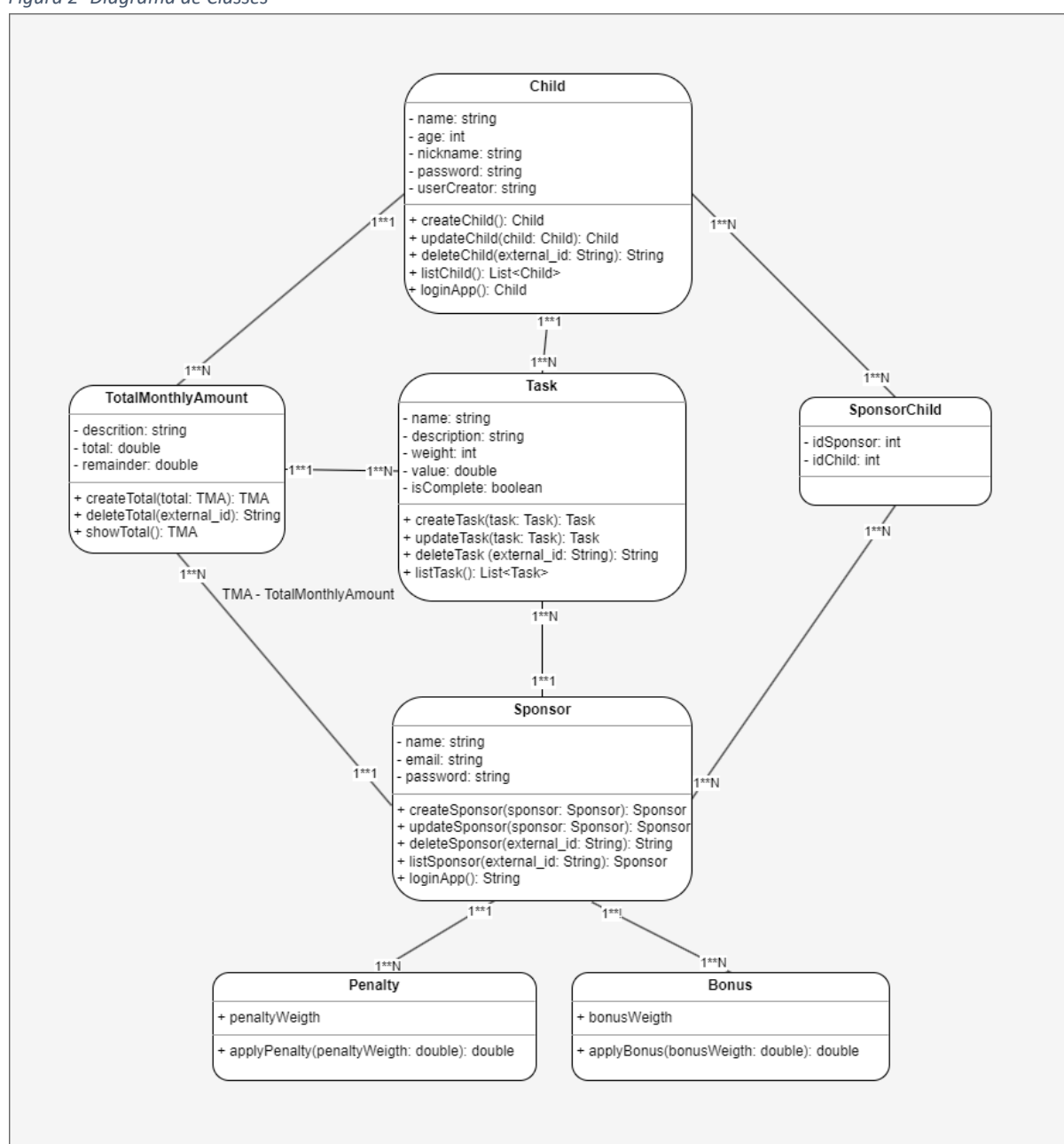
4.6. Diagrama de Classe 6: [Penalty]

O diagrama de classe "Penalty" representa uma classe chamada "Penalty" e também uma tabela no banco de dados que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem o atributo penaltyWeigth e método applyPenalty;

4.7. Diagrama de Classe 7: [Bonus]

O diagrama de classe "Bonus" representa uma classe chamada "Bonus" e também uma tabela no banco de dados que está relacionada ao sistema de gestão de recompensas para tarefas de crianças. Essa classe tem o atributo bonusWeigth e método applyBonus;

Figura 2- Diagrama de Classes



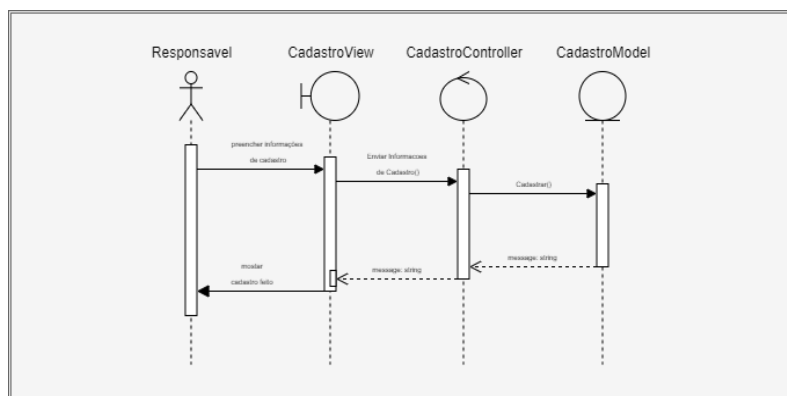
5. Diagramas de Sequência

O diagrama de sequência permite visualizar de forma clara e sucinta como os objetos se comunicam e interagem entre si em um sistema, facilitando a compreensão do fluxo de execução e a lógica por trás das interações.

5.1. Diagrama de sequência 1: [Cadastrar-se]

Esse diagrama de sequência descreve o processo de cadastro do usuário responsável no sistema. Ele mostra a interação entre o responsável e o sistema, onde este fornece as informações necessárias para criar a primeira conta do sistema, como nome, e-mail e senha. O sistema valida e registra as informações do usuário para permitir o acesso por login posteriormente.

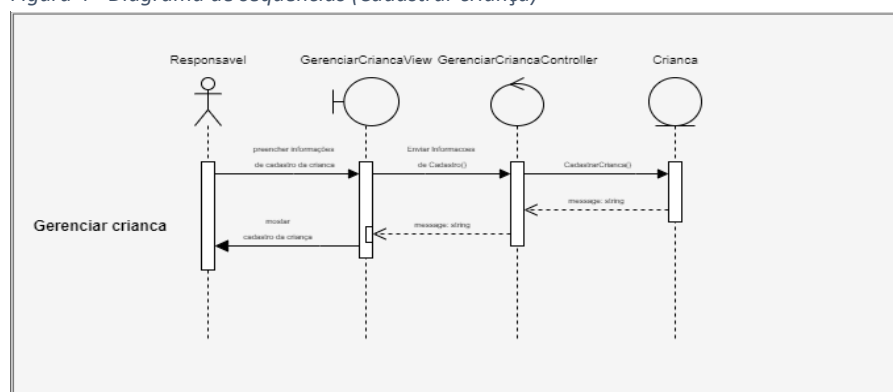
Figura 3 - Diagrama de sequências (cadastrar-se)



5.2. Diagrama de sequência 2: [Cadastrar criança]

Esse diagrama de sequência mostra o processo de cadastro de uma criança no sistema feita pelo responsável. Ele ilustra a interação entre o responsável e o sistema, onde este fornece os detalhes da criança, como nome, idade, nickname e senha. O sistema registra essas informações para permitir o gerenciamento das tarefas da criança posteriormente.

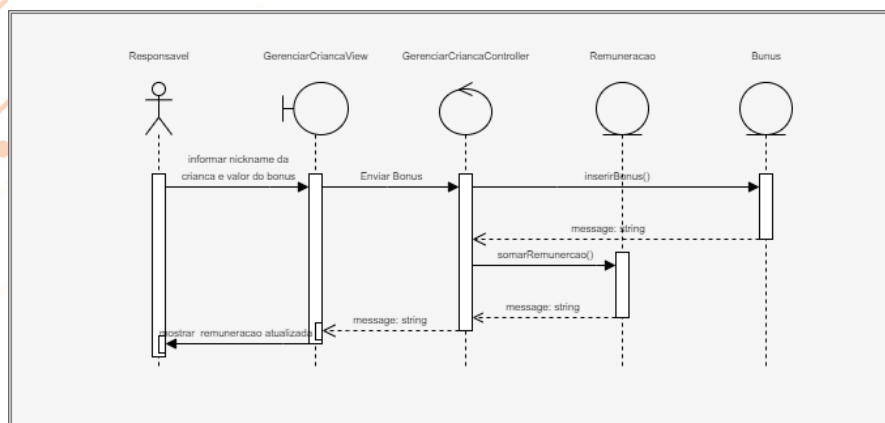
Figura 4 - Diagrama de sequências (Cadastrar criança)



5.3. Diagrama de sequência 3: [Cadastrar bônus]

Nesse diagrama de sequência, é apresentado o processo de cadastro de bônus no sistema. Ele representa a interação entre o responsável e o sistema, onde este fornece informações sobre o valor adicional além do valor total mensal já registrado. Para isso ele informa o valor do bônus. O sistema registra esses detalhes para permitir que os bônus sejam somados ao valor total da criança.

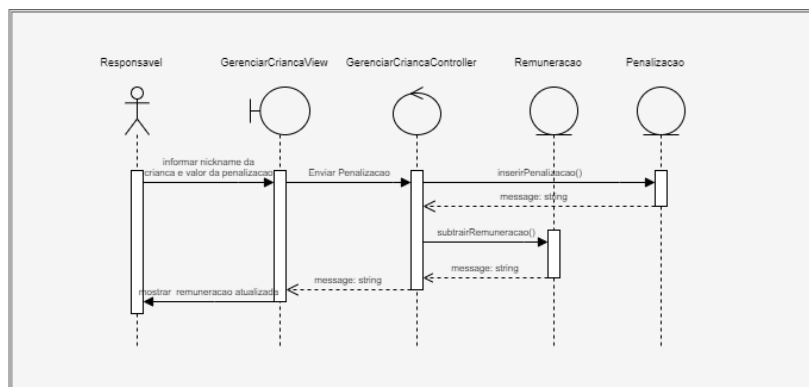
Figura 5 - Diagrama de seqüências (Cadastrar bônus)



5.4. Diagrama de seqüência 4: [Cadastrar penalidade]

Esse diagrama de seqüência descreve o processo de cadastro de penalidade no sistema. Ele mostra a interação entre o responsável e o sistema, onde este fornece informações sobre o valor que será subtraído do valor total. Para isso ele passa o valor da penalidade. O sistema registra esses detalhes para permitir que as penalidades sejam subtraídas do montante da criança.

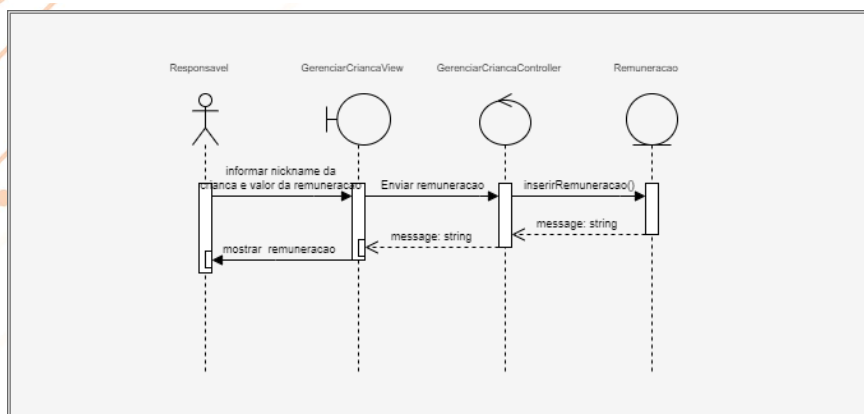
Figura 6 - Diagrama de seqüências (Registrar penalidade)



5.5. Diagrama de seqüência 5: [Cadastrar remuneração]

Nesse diagrama de seqüência, é apresentado o processo de cadastro de remuneração no sistema. Ele representa a interação entre o responsável e o sistema, onde este fornece informações sobre o valor total mensal atribuído as atividades a serem realizadas pela criança. Para isso ele informa o valor total mensal. O sistema registra esses detalhes para permitir que a remuneração seja concedida a criança conforme as tarefas são concluídas.

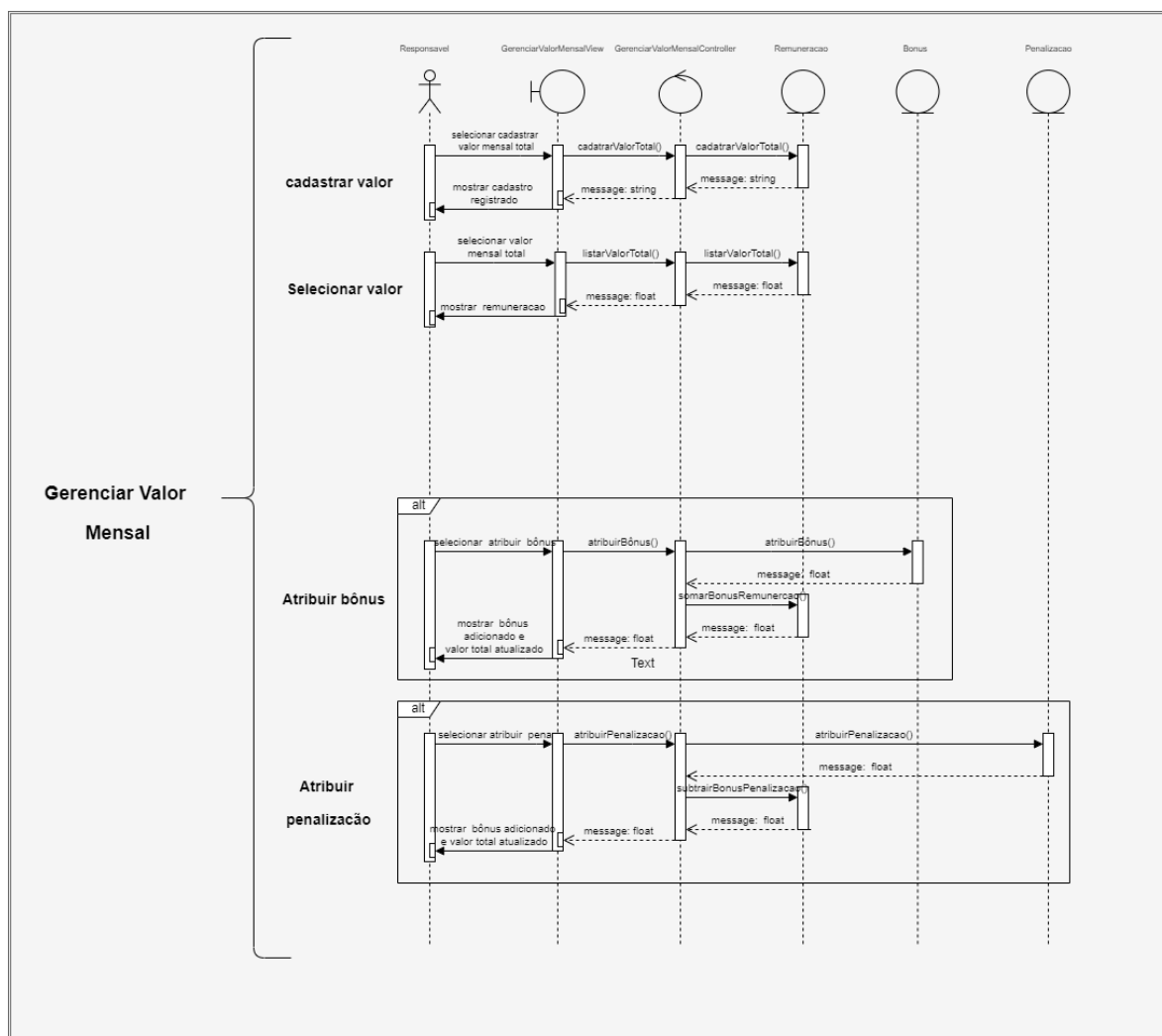
Figura 7 - Diagrama de seqüências (Cadastrar remuneração)



5.6. Diagrama de seqüência 6: [Gerenciar valor mensal]

Esse diagrama de seqüência mostra o processo de gerenciamento do valor mensal para uma criança feita pelo responsável. Ele ilustra a interação entre o responsável e o sistema, onde este define o valor mensal a ser atribuído à criança para a realização das tarefas (cadastro) bem como a adição de valores (bônus) ou retirada de valores (penalidade). Nesta seqüência não é possível editar o valor mensal, somente aplicar penas e bônus.

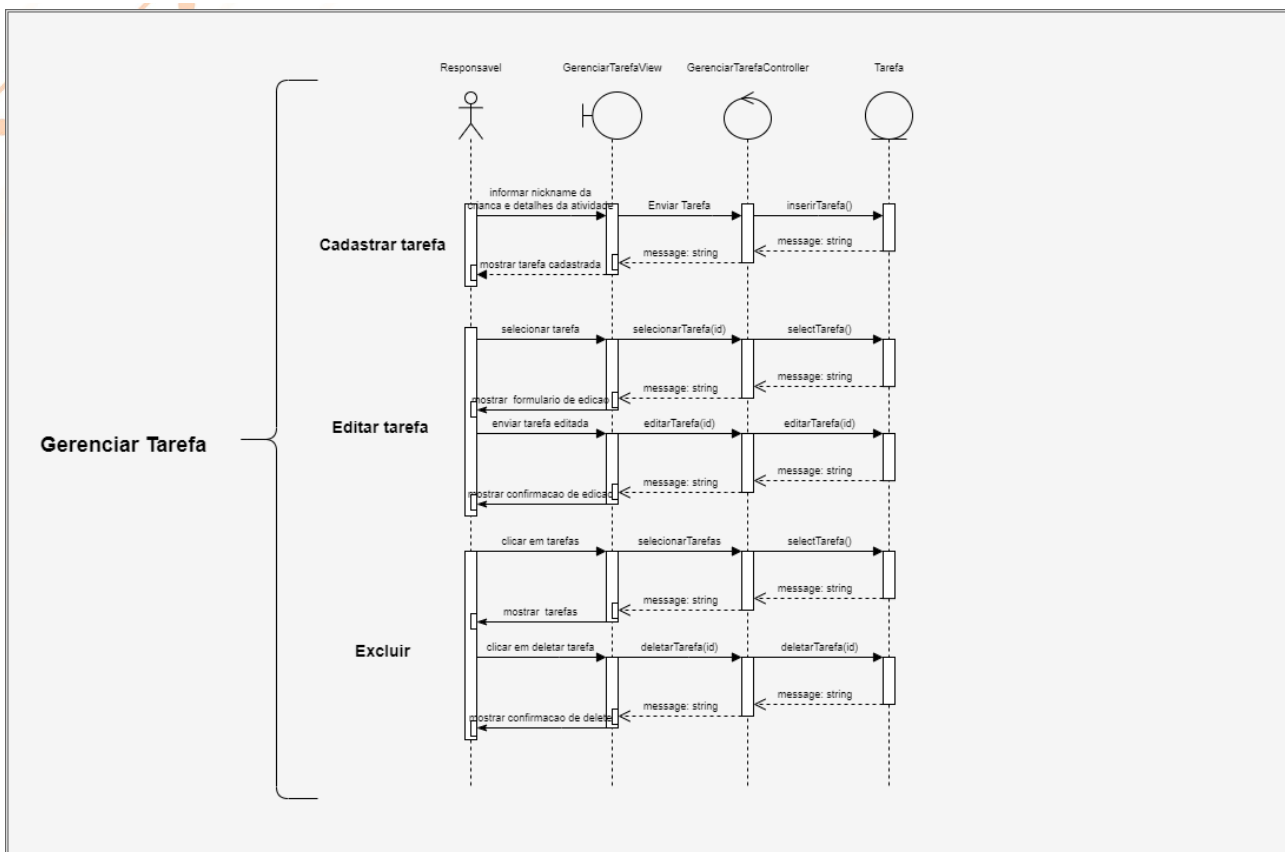
Figura 8- Diagrama de seqüências (Gerenciar Valor Mensal)



5.7. Diagrama de sequência 7: [Gerenciar tarefas]

Nesse diagrama de sequência, é apresentado o processo de gerenciamento das tarefas feito pelo responsável. Ele representa a interação entre o responsável e o sistema, onde este pode adicionar, remover, editar e atribuir pesos a tarefa para uma determinada criança.

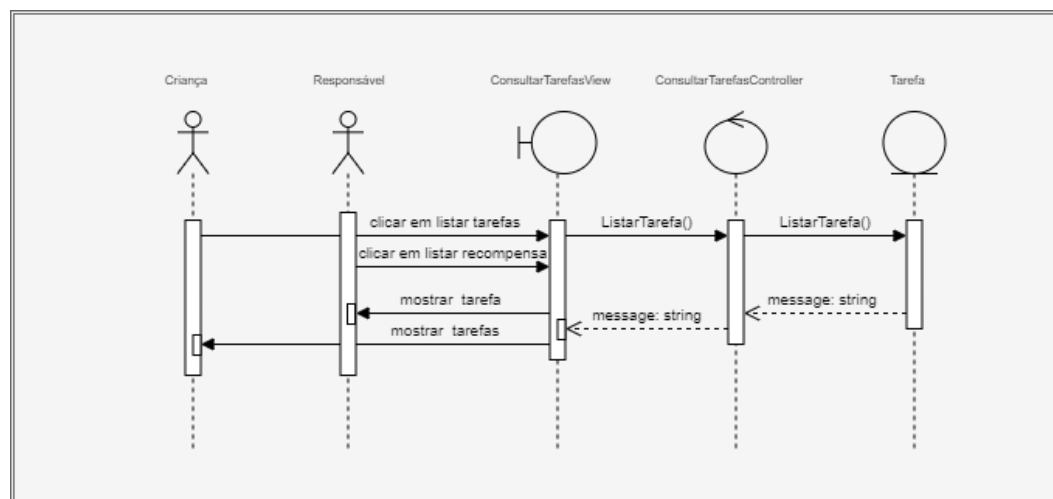
Figura 9 - Diagrama de sequências (Gerenciar tarefa)



5.8. Diagrama de sequência 9: [Consultar tarefa]

Nesse diagrama de sequência, é apresentado o processo de consulta de informações sobre uma tarefa no sistema. Ele representa a interação entre o responsável ou criança e o sistema, onde os dois podem solicitar a consulta de uma tarefa específica. O sistema recupera os dados da tarefa, como, nome, descrição, peso, valor e se está concluída e exibe para o usuário.

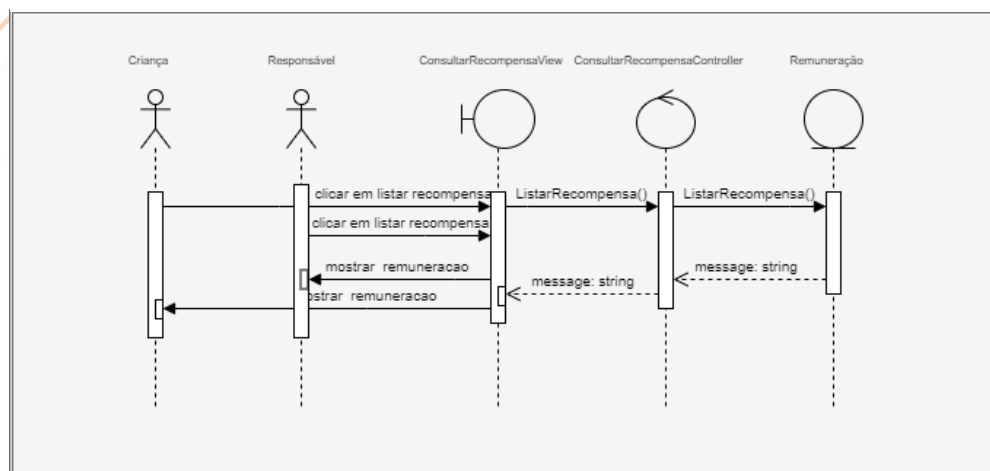
Figura 10 - Diagrama de sequências (Consultar tarefas)



5.9. Diagrama de sequência 10: [Consultar recompensa]

Esse diagrama de sequência descreve o processo de consulta de informações sobre a recompensa ganha pela criança. Ele ilustra a interação entre o responsável ou criança e o sistema, onde os dois podem solicitar a consulta de uma do valor de recompensa ganha. O sistema recupera os dados da recompensa, como o valor ganho pela criança em relação ao valor total e exibe ao usuário.

Figura 11 - Diagrama de sequências (Consultar remuneração)



6. API REST Desenvolvida

6.1. Objetivos da API REST desenvolvida

a. Descrição dos principais objetivos e funcionalidades da API

O objetivo principal da API é mostrar a prática do funcionamento da modelagem de dados realizada através dos conhecimentos adquiridos ao decorrer do semestre.

b. Casos de uso em que a API é aplicável

Utilizada no dia a dia para o gerenciamento de atividades de crianças.

6.2 Tecnologias utilizadas

a. Linguagem de programação e framework escolhidos para o desenvolvimento da API

Java com o framework Spring boot

b. Ferramentas e bibliotecas utilizadas na implementação da API

IntelliJ – IDE;

Postman - Plataforma de API para desenvolvedores projetarem, construírem, testarem e iterarem suas APIs.

6.3 Arquitetura da API REST

a. Descrição da arquitetura geral da API

Utilizamos o padrão MVC

b. Explicação dos componentes e camadas envolvidas

Criamos a camada de controller que recebe os dados da view. A camada de controller se comunica com a camada de service. Na camada de service fica as validações e a regra de negócio. A camada de service se comunica com o a camada de model e repository que por sua vez se comunica com o banco de dados.

c. Decisões de design e padrões arquiteturais adotados

Utilizamos o conhecimento extraclasse para construir a API. Os padrões utilizados, são os padrões básicos para construção de uma API.

6.4 Endpoints da API Rest

a. Listagem dos principais endpoints disponibilizados pela API

/api-rest/sponsor/new-register
/api-rest/update-sponsor/{externalId}
/api-rest/list-sponsor/{externalId}
/api-rest/delete-sponsor/{externalId}
/api-rest/child/new-register
/api-rest/update-child/{externalId}
/api-rest/list-child/{externalId}
/api-rest/delete-child/{externalId}
/api-rest/login
/api-rest/new-total
/api-rest/show-total/{externalId}
/api-rest/delete-total/{externalId}
/api-rest/apply-bonus
/api-rest/apply-penalty
/api-rest/new-task
/api-rest/list-task/{externalId}
/api-rest/update-task/{externalId}
/api-restdelete-task/{externalId}

PUT

PUT	/api-rest/update-task/{externalId}	Update task	▼
PUT	/api-rest/update-sponsor/{externalId}	Update sponsor	▼
PUT	/api-rest/update-child/{externalId}	Update child	▼


PUT	/api-rest/update-task/{externalId}	Update task	
PUT	/api-rest/update-sponsor/{externalId}	Update sponsor	
PUT	/api-rest/update-child/{externalId}	Update child	

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
5   "externalIdSponsor": "88f9c581-8372-4785-b414-618f9c972297",
6   "name": "Llevar dar banho no cachorro",
7   "description": "Deverá ser dado banho no Bili todas as sextas feiras",
8   "weight": 10,
9   "isComplete": false
10 }
11 //Saida
12 {
13   "externalId": "52ffc5f6-1f84-4713-b848-8c1d7b1e18a2",
14   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
15   "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
16   "externalIdSponsor": "88f9c581-8372-4785-b414-618f9c972297",
17   "name": "Llevar dar banho no cachorro",
18   "description": "Deverá ser dado banho no Bili todas as sextas feiras",
19   "weight": 10,
20   "value": 200,
21   "remainder": 100,
22   "complete": false
23 }
```

```
1 //Entrada
2 {
3   "name": "Jackson Marcelino de Freitas",
4   "email": "jackson@gmail.com",
5   "password": "troqueidesenha"
6 }
7 //Saida
8 {
9   "externalId": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
10  "email": "jackson@gmail.com",
11  "name": "Jackson Marcelino de Freitas"
12 }
```

```
1 //Entrada
2 {
3   "externalId": "88f9c581-8372-4785-b414-618f9c972297",
4   "name": "Juliana Vilela Santana",
5   "nickname": "Juju",
6   "age": 13
7 }
8 //Saida
9 {
10  "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
11  "name": "Juliana Vilela Santana",
12  "nickname": "Juju",
13  "age": 13,
14  "password": "ju123"
15 }
16
17 }
```

POST

POST	/api-rest/sponsor/new-register	Register sponsor	▼
POST	/api-rest/new-total	Register total monthly amount	▼
POST	/api-rest/new-task	Register new task	▼
POST	/api-rest/login	Logs the user into the application	▼
POST	/api-rest/child/new-register	Register child	▼
POST	/api-rest/apply-penalty	Apply penalty	▼ 
POST	/api-rest/apply-bonus	Apply bonus	▼

POST	/api-rest/sponsor/new-register	Register sponsor	
POST	/api-rest/new-total	Register total monthly amount	
POST	/api-rest/new-task	Register new task	

```
1 //Entrada
2 {
3   "name": "Jackson Freitas",
4   "email": "jackson@gmail.com",
5   "password": "jackson123"
6 }
7 //Saida
8 {
9   "message": "successfully registered user",
10  "code": 201
11 }
12 }
```

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
5   "total": 500,
6   "description": "Varrer a calçada"
7 }
8 //Saida
9 {
10  "externalId": "ebc51e4a-6ea3-4c8b-b854-7cd8ae4553e",
11  "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
12  "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
13  "total": 500,
14  "totalValue": 0,
15  "remainder": 500,
16  "description": "Varrer a calçada"
17 }
```

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
5   "externalIdSponsor": "88f9c581-8372-4785-b414-618f9c972297",
6   "name": "Llevar casa",
7   "description": "Esta tarefa inclui a prática de varrer a calçada e lavar as louças",
8   "weight": 50,
9   "value": 200,
10  "remainder": 200,
11  "complete": false
12 }
```

POST	/api-rest/login	Logs the user into the application	
POST	/api-rest/child/new-register	Register child	
POST	/api-rest/apply-penalty	Apply penalty	

```
1 //Entrada
2 {
3   "user": "Vilela",
4   "password": "vilela123",
5   "isChild": true
6 }
7 //Saida
8 {
9   "externalId": "88f9c581-8372-4785-b414-618f9c972297",
10  "name": "Juliana Vilela Santana",
11  "nickname": "Vilela",
12  "age": 13
13 }
14 }
```

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "name": "Juliana Vilela Santana",
5   "nickname": "Vilela",
6   "age": 13,
7   "password": "vilela123"
8 }
9 //Saida
10 {
11  "message": "successfully registered user",
12  "code": 201
13 }
14 }
```

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "externalIdTotal": "ebc51e4a-6ea3-4c8b-b854-7cd8ae4553e",
5   "penaltyWeight": 25
6 }
7 //Saida
8 {
9   "externalId": "ebc51e4a-6ea3-4c8b-b854-7cd8ae4553e",
10  "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
11  "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
12  "total": 125,
13  "totalValue": 125,
14  "remainder": 125,
15  "description": "Varrer a calçada"
16 }
```

POST	/api-rest/apply-bonus	Apply bonus	
------	-----------------------	-------------	--

```
1 //Entrada
2 {
3   "name": "Jackson Freitas",
4   "email": "jackson@gmail.com",
5   "password": "jackson123"
6 }
7 //Saida
8 {
9   "externalId": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
10  "email": "jackson@gmail.com",
11  "name": "Jackson Freitas"
12 }
```

```
1 //Entrada
2 {
3   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
4   "externalIdTotal": "ebc51e4a-6ea3-4c8b-b854-7cd8ae4553e",
5   "bonusWeight": 10
6 }
7 //Saida
8 {
9   "externalId": "ebc51e4a-6ea3-4c8b-b854-7cd8ae4553e",
10  "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
11  "externalIdChild": "88f9c581-8372-4785-b414-618f9c972297",
12  "total": 410,
13  "totalValue": 37,5,
14  "remainder": 102,5,
15  "description": "Varrer a calçada"
16 }
```

GET

GET	/api-rest/show-total/{externalId}	Returns a total monthly amount	▼
GET	/api-rest/list-task/{externalId}	Returns all tasks of Sponsor	▼
GET	/api-rest/list-sponsor/{externalId}	Returns sponsor	▼
GET	/api-rest/list-child/{externalId}	Returns all child of Sponsor	▼

GET /api-rest/show-total/{externalId} Returns a total monthly amount

```
1 //Entrada
2 // /api-rest/show-total/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5
6 {
7   "externalId": "ebc51e4a-6ea3-4c88-b854-7ce81ae4553e",
8   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
9   "externalIdChild": "88f9c501-8372-4785-b414-610f9c972297",
10  "total": 412.5,
11  "totalValue": 0.0,
12  "remainder": 162.5,
13  "description": "Varrer a calçada"
14 }
15
16
```

GET /api-rest/list-task/{externalId} Returns all tasks of Sponsor

```
1 //Entrada
2 // /api-rest/list-task/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5 {
6   "externalId": "52ffc5f4-1f04-4713-b0d0-8c1d7b1ef8e2",
7   "externalIdSponsor": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
8   "externalIdChild": "88f9c501-8372-4785-b414-610f9c972297",
9   "externalIdTotal": "ebc51e4a-6ea3-4c88-b854-7ce81ae4553e",
10  "name": "Dar banho no cachorro",
11  "description": "Deverá ser dado banho no Biliu todas as sextas feiras",
12  "weight": 50,
13  "value": 250.0,
14  "remainder": 162.5,
15  "complete": false
16 }
17
```

GET /api-rest/list-child/{externalId} Returns all child of Sponsor

```
1 //Entrada
2 // /api-rest/list-child/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5 {
6   "externalId": "88f9c501-8372-4785-b414-610f9c972297",
7   "name": "Juliana Vilela Santana",
8   "nickname": "Jujuz",
9   "age": 13
10 }
```

GET /api-rest/list-sponsor/{externalId} Returns sponsor

```
1 //Entrada
2 // /api-rest/list-sponsor/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5 {
6   "externalId": "1c57bd38-8a93-456b-ae1b-bbfcfe4647b6",
7   "email": "jackson@gmail.com",
8   "name": "Jackson Marcelino de Freitas"
9 }
```

DELETE

DELETE	/api-rest/delete-total/{externalId}	Delete a total monthly amount	▼
DELETE	/api-rest/delete-task/{externalId}	Delete a task	▼
DELETE	/api-rest/delete-sponsor/{externalId}	Delete a sponsor	▼
DELETE	/api-rest/delete-child/{externalId}	Delete a child	▼

DELETE /api-rest/delete-total/{externalId} Delete a total monthly amount

```
1 //Entrada
2 // /api-rest/delete-sponsor/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5 {
6   "message": "successfully delete user",
7   "code": 200
8 }
9
```

DELETE /api-rest/delete-task/{externalId} Delete a task

```
1 //Entrada
2 // /api-rest/delete-task/52ffc5f4-1f04-4713-b0d0-8c1d7b1ef8e2
3
4 //Saída
5 {
6   "message": "successfully delete total monthly amount",
7   "code": 200
8 }
9
```

DELETE /api-rest/delete-child/{externalId} Delete a child

```
1 //Entrada
2 // /api-rest/delete-child/88f9c501-8372-4785-b414-610f9c972297
3
4 //Saída
5 {
6   "message": "successfully delete user",
7   "code": 200
8 }
9
```

DELETE /api-rest/delete-sponsor/{externalId} Delete a sponsor

```
1 //Entrada
2 // /api-rest/delete-sponsor/1c57bd38-8a93-456b-ae1b-bbfcfe4647b6
3
4 //Saída
5 {
6   "message": "successfully delete user",
7   "code": 200
8 }
9
```


6.5 Formatos de dados

a. Formato de dados utilizado nas requisições e respostas

JSON

6.6 Tratamento de erros e exceções

a. Estratégias para lidar com erros e exceções na API

Na API temos um serviço personalizado para tratamento de erro no package exceptions

b. Códigos de status HTTP utilizados para indicar diferentes tipos de erros

200 - SUCCESS

201 - CREATED

400 - BAD REQUEST

401 - UNAUTHORIZED

404 - NOT FOUND

6.7 Testes e documentação da API

a. Estratégias de teste adotadas

Testes unitários da camada de service

b. Testes realizados

Os testes realizados fazem a avaliação dos métodos da aplicação, testando caminhos onde a possibilidade é o erro ou o acerto. Para isso é utilizado o Mockito que é um framework de mocks que permite escrever uma API de testes simples e limpa. Portanto é feito o preenchimento mockado e estático dos métodos.

ApiRestApplicationTests: 29 total, 29 passed

2.54 s

Collapse | Expand

testCreateChild_WhenChildExists_ExpectBadRequest()	passed	346 ms
testUpdateChild_WhenChildDoesNotExist_ExpectNotFoundResponse()	passed	28 ms
testCreateSponsor_WhenSponsorExists_ExpectBadRequest()	passed	19 ms
testCreateTask_WhenValidData_ExpectCreatedResponse()	passed	28 ms
testLogin_WhenSponsorLoginWithRegisteredEmailAndPassword_ExpectOkResponse()	passed	369 ms
testCreateChild_WhenChildCreated_ExpectCreated()	passed	17 ms
testUpdateTask_WhenTaskExists_ExpectOkResponse()	passed	183 ms
testLogin_WhenChildLoginWithInvalidNickname_ExpectBadRequestException()	passed	15 ms
testListTask_WhenSponsorExists_ExpectOkResponse()	passed	14 ms
testDeleteChild_WhenChildExists_ExpectOkResponse()	passed	17 ms
testUpdateSponsor_WhenSponsorExists_ExpectOkResponse()	passed	249 ms
testCreateChild_WhenSponsorNotFound_ExpectNotFound()	passed	13 ms
testListChild_WhenChildExists_ExpectOkResponse()	passed	18 ms
testCreateTask_WhenChildNotFound_ExpectBadRequestResponse()	passed	12 ms
testLogin_WhenChildLoginWithRegisteredNicknameAndPassword_ExpectOkResponse()	passed	267 ms
testDeleteTask_WhenTaskDoesNotExist_ExpectNotFoundResponse()	passed	15 ms
testLogin_WhenChildLoginWithValidCredentials_ExpectOkResponse()	passed	311 ms
testListTask_WhenNoMatchFound_ExpectNotFoundResponse()	passed	48 ms
testListChild_WhenChildDoesNotExist_ExpectNotFoundResponse()	passed	15 ms
testCreateSponsor_WhenSponsorDoesNotExist_ExpectCreated()	passed	19 ms
testDeleteTask_WhenTaskExists_ExpectOkResponse()	passed	12 ms
testLogin_WhenSponsorLoginWithInvalidUser_ExpectBadRequestException()	passed	17 ms
testListSponsor_WhenSponsorExists_ExpectOkResponse()	passed	14 ms
testDeleteSponsor_WhenSponsorDoesNotExist_ExpectNotFoundResponse()	passed	44 ms
testLogin_WhenChildLoginWithInvalidPassword_ExpectUnauthorizedException()	passed	250 ms
testDeleteSponsor_WhenSponsorExists_ExpectOkResponse()	passed	16 ms
testUpdateChild_WhenChildExists_ExpectOkResponse()	passed	160 ms
testUpdateTask_WhenTaskDoesNotExist_ExpectNotFoundResponse()	passed	15 ms
testCreateTask_WhenSponsorNotFound_ExpectBadRequestResponse()	passed	9 ms

Generated by IntelliJ IDEA on 6/20/23, 10:42 PM

b. Documentação da API, incluindo documentação de endpoints e exemplos de uso

Foi utilizado o Swagger que é uma ferramenta popular para documentar endpoints em APIs. Com ele, é possível descrever detalhes como os pontos de extremidade, parâmetros, formatos de dados e respostas da API. A documentação gerada pelo Swagger é legível tanto para humanos quanto para máquinas, facilitando a compreensão e o consumo da API. Além disso, o Swagger também permite testar os endpoints diretamente na interface, tornando o processo de desenvolvimento mais eficiente.

6.8 Considerações de segurança

a. Medidas de segurança implementadas para proteger a API e os dados transmitidos

Criptografia de senha

Uso do UUID para gerar o id que será transitado fora da API

7. Conclusão

A modelagem de sistemas desempenha um papel fundamental no processo de desenvolvimento de software, pois permite a representação visual e estruturada do sistema em diferentes perspectivas. Os diagramas de sequência, diagramas de classes e casos de uso são ferramentas essenciais nesse contexto, pois proporcionam uma visão abrangente do sistema, desde a interação entre os objetos em diferentes cenários até a estrutura estática do sistema e as funcionalidades disponíveis para os usuários. Essas representações visuais facilitam a comunicação e a compreensão entre os membros da equipe de desenvolvimento, permitindo uma visão clara dos requisitos, comportamento e estrutura do sistema. Além disso, esses diagramas servem como base para a tomada de decisões e para o planejamento de implementação, ajudando a identificar possíveis problemas ou lacunas no design antes da fase de implementação, economizando tempo e recursos no longo prazo. Em resumo, a modelagem de sistemas com o uso de diagramas adequados é crucial para garantir a correta concepção e desenvolvimento de um sistema de software, fornecendo uma representação visual completa e compreensível de suas funcionalidades, interações e estrutura.

De outra forma, a metodologia Scrum ágil desempenhou um papel fundamental na promoção de uma boa modelagem de software no projeto. Por meio de sua abordagem iterativa e incremental, o Scrum permitiu uma colaboração efetiva entre os membros da equipe de desenvolvimento, fomentando a comunicação constante e a troca de feedback. Isso foi especialmente importante na modelagem do software, pois permitiu que as necessidades do software e as alterações nos requisitos fossem prontamente incorporadas ao processo. Além disso, ao utilizar de artefatos leves e iterativos, como histórias de usuário e tarefas, pudemos causar engajamento e entrega entre toda a equipe. Dessa forma, a metodologia Scrum ágil promoveu uma modelagem mais adaptativa, colaborativa e flexível, garantindo que o software seja desenvolvido de acordo com as necessidades e expectativas dos modelos de diagramas apresentados, com a possibilidade de ajustes contínuos ao longo do projeto.

Repositório:

<https://github.com/Matheus-Juliao/Projeto-Modelagem-De-Sistemas>