Split&Conquer

Aplicação de Gerenciamento de Despesas em Grupo

Autores: Gabriel Perini, Paulo Ceccato, Pedro Bavaresco e Rafael Petry

Sumário

- Descrição;
- Implementação;
- Testes unitários;
- Demonstração;

Descrição: Problema

- Fazer o gerenciamento de despesas de um grupo;
- Criação de grupos;
- Registro de despesas;
- Divisão automática de valores;
- Similar a outras aplicações, como SplitWise;

Descrição: Tecnologias Utilizadas

- Front-end: ReactJS;
- Back-end: API em Java utilizando a *framework* Spring;
- Testes: JUnit;

Implementação: API

- Pacotes do Spring: *boot* e *web.bind.annotation*;
- Pacote do Projeto: *com.splitandconquer.api*:
  - *ApiApplication*: Inicializa a API;
  - *controllers*: Roteamento de consultas;
  - *models*: Entidades envolvidas nas consultas;
  - *payloads*: "Moldes" para receber valores de POST;
  - *responses*: "Moldes" de resposta às consultas;
  - *views*: Interfaces que selecionam atributos do retorno em JSON;

```java
@RestController
@RequestMapping("users")
public class UserController {
    private static ArrayList<User> allUsers = new ArrayList<User>();

    @GetMapping("")
    public static AllUsersResponse getUsers() {
        return new AllUsersResponse(true, UserController.allUsers);
    }
```

Exemplo de classe do tipo *controller*

```java
public class User {
    @JsonView({GroupViews.SingleGroupView.class, GroupViews.ExpensesView.class})
    private int id;

    @JsonView({GroupViews.SingleGroupView.class, GroupViews.ExpensesView.class})
    private String name;

    @JsonIgnore
    private ArrayList<Balance> balances = new ArrayList<Balance>();
```

Exemplo de classe do tipo *model*

```
public record PaymentPayload(int payerId, int receiverId, float amount, int groupId) { }
```

Exemplo de *Payload*

```
@JsonView(GroupViews.AllGroupsView.class)
public record AllGroupsResponse(boolean success, ArrayList<Group> content) { }
```

Exemplo de *Response*

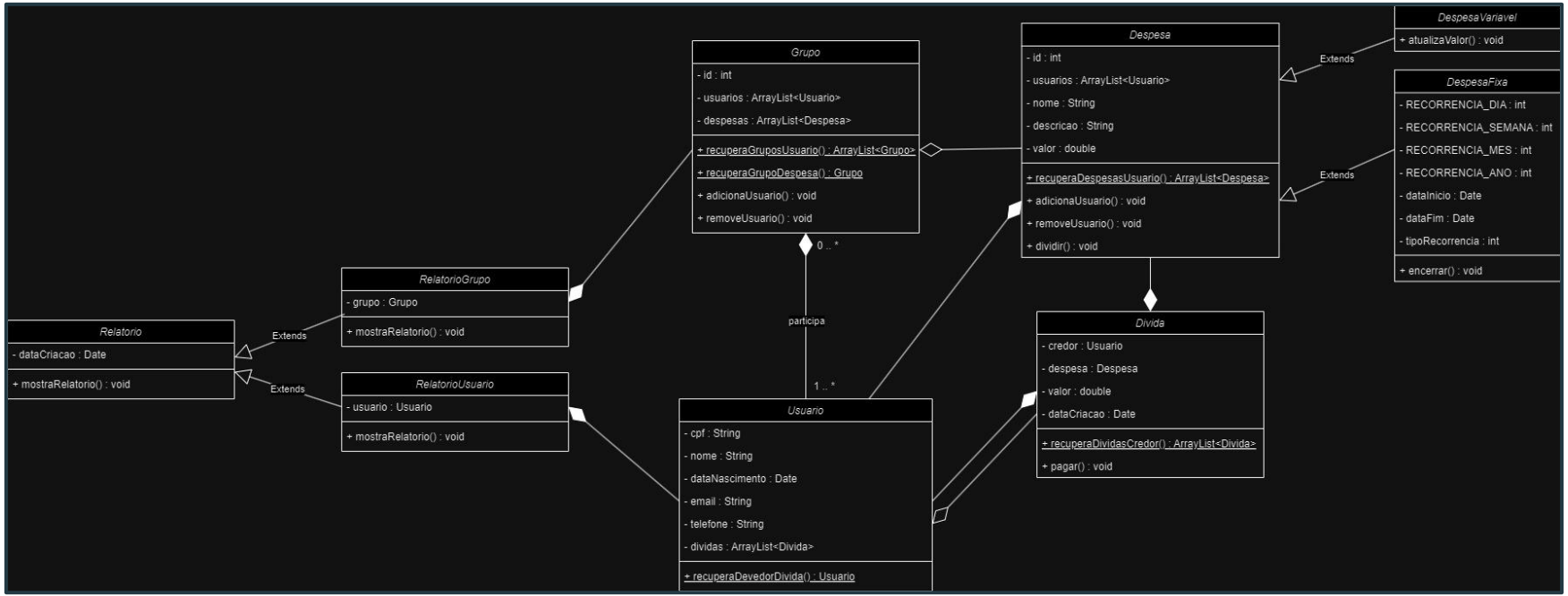
```
public interface GroupViews {
    public interface SingleGroupView {};
    public interface AllGroupsView {};
    public interface ExpensesView {};
}
```

Exemplo de *View*

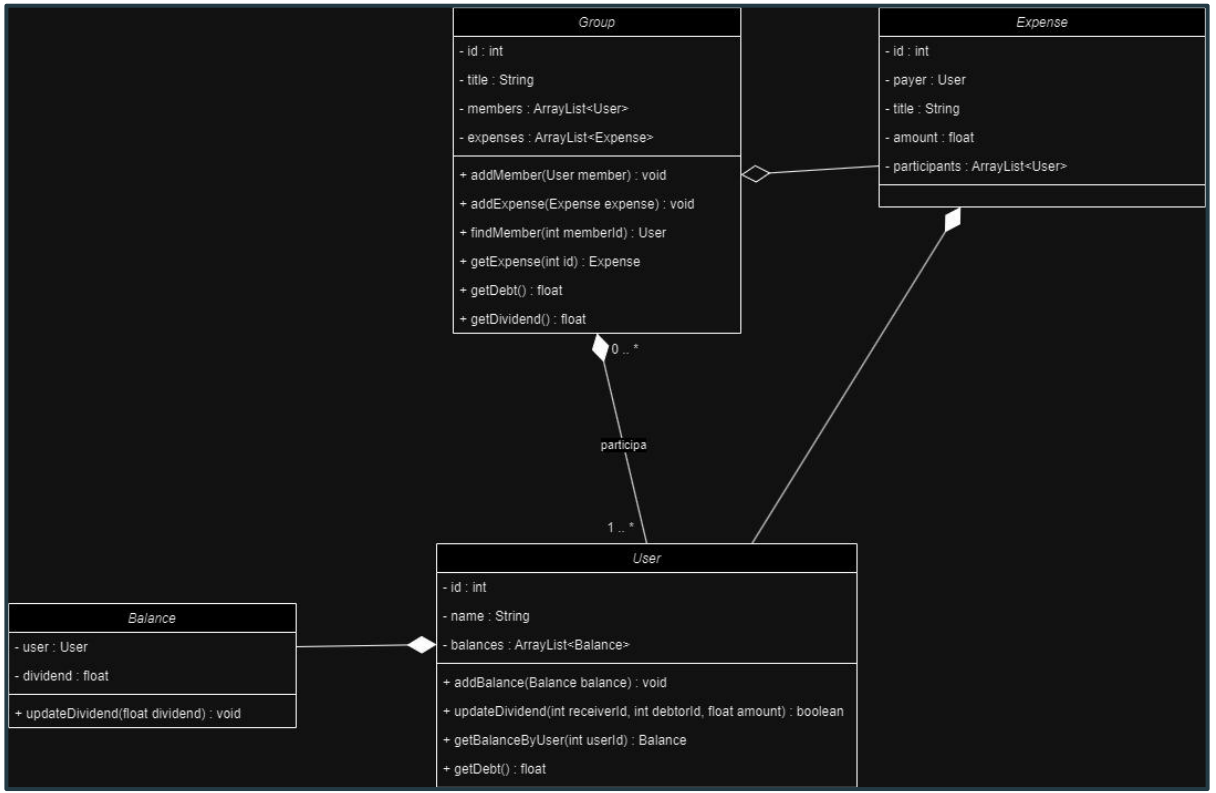
Implementação: API

- Possui 4 *models*;
- Possui 4 *controllers*;
- Possui 8 *endpoints*, sendo eles do tipo GET, POST ou DELETE;

Models: Diagrama da Etapa 1



Models: Diagrama Atual



Testes

- Cálculos monetários;
- Buscas de *models* por meio de IDs;
- Validação de resposta de um *endpoint*;
- Implementados em *models* e *controllers*
- O sistema possui 30 testes unitários;

```
// class UserTest
// Verifica se o updateDividend do usuário somou o terceiro parâmetro e arredondou para
//duas casas decimais

@Test
public void testAddDividend() {
    this.user.updateDividend(0, 1, 100.92156f);
    assertEquals(balance.getDividend(), 100.92f);
}
```

Exemplo de teste em *model*

```
// class GroupControllerTest
// Verifica se o updateDividend do usuário somou o terceiro parâmetro e arredondou para
//duas casas decimais

// Dado um ID existente, verifica se a resposta retorna sucesso
@Test
public void testGetGroupSuccess() {
    SingleGroupResponse response = GroupController.getGroup(0);
    assertTrue(response.success());
}

// Dado um ID inexistente, verifica se a resposta retorna falha
@Test
public void testGetGroupFail() {
    SingleGroupResponse response = GroupController.getGroup(1);
    assertFalse(response.success());
}
```

Exemplo de teste em *controller*

Demonstração

Agradecemos a atenção de todos!
Dúvidas?