1.Requisitos, projeto e interface

1.1.Mudanças em relação à etapa anterior

As alterações que pensamos foi em relação ao funcionamento do lançamento de “dívidas”. Após conversar com a professora, percebemos que a funcionalidade é mais complexa do que havíamos considerado inicialmente, visto que existiriam os casos em que mais de uma pessoa fazem o pagamento por um “evento”, o que é diferente do que havíamos definido anteriormente (onde levamos em consideração apenas o caso de uma pessoa fazer o pagamento). Neste caso, precisaríamos fazer, também, uma função de “desconto”, em relação a cada uma das pessoas que fizeram o pagamento, no momento de lançar as dívidas.

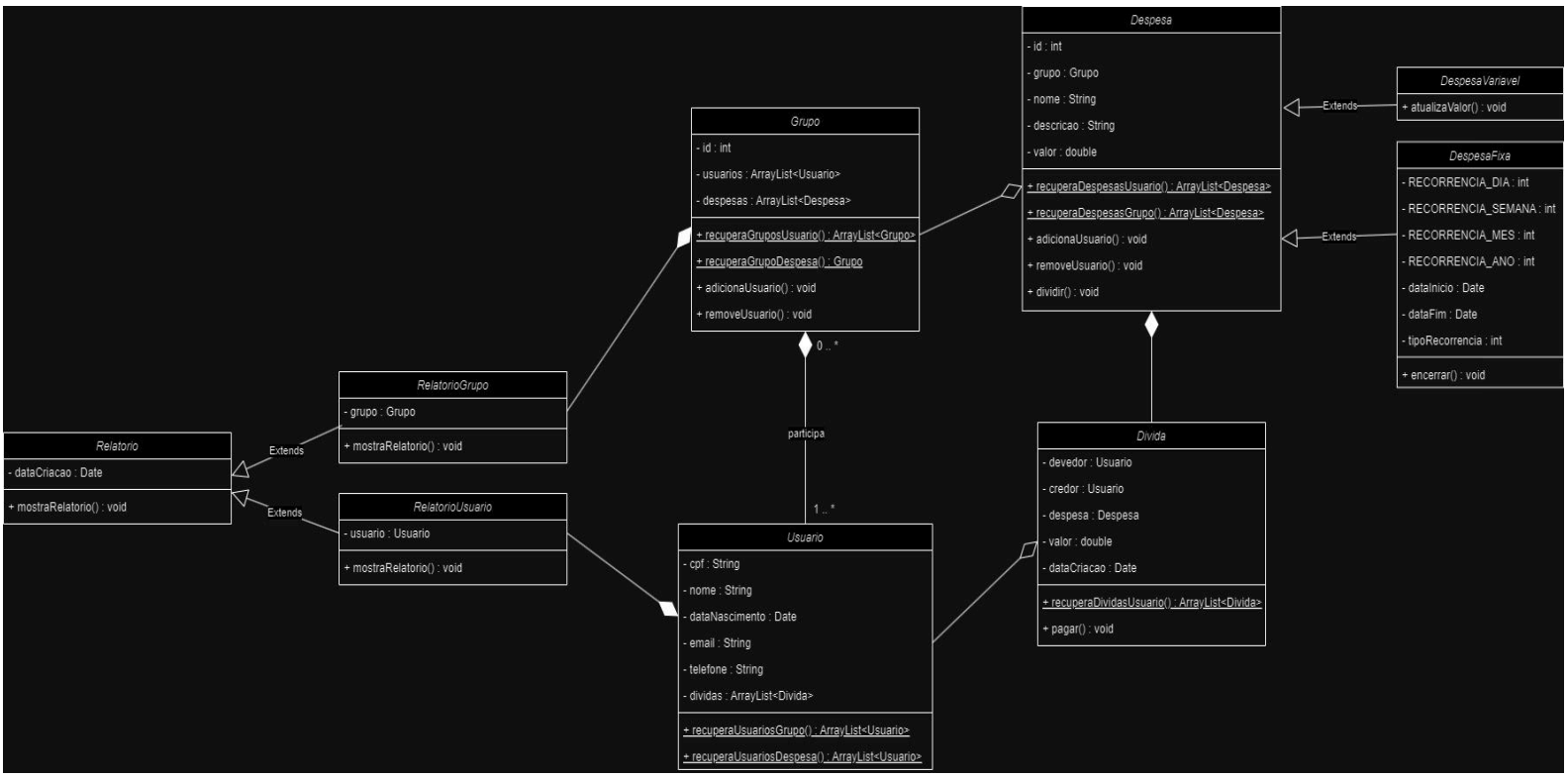
1.2.Requisitos

Requisitos Funcionais	Requisitos Não Funcionais
RF-1: O usuário deve ser capaz de cadastrar uma conta.	**RNF-1:** O sistema deve ser desenvolvido apenas com ferramentas gratuitas.
RF-2: O usuário deve ser capaz de fazer login na sua conta.	**RNF-2:** O sistema deve ser capaz de persistir os dados.
RF-3: O usuário deve ser capaz de criar grupos.	**RNF-3:** O sistema deve gerar mensagens de sucesso e erro ao manipular dados.
RF-4: Dentro de um grupo, o usuário deve ser capaz de incluir outros usuários.	**RNF-4:** As operações mais utilizadas não devem estar a mais de 3 cliques após o login.
RF-5: Dentro de um grupo, o usuário deve ser capaz de criar novas despesas.	**RNF-5:** As operações feitas no sistema devem ser processadas em menos de 3 segundos.
RF-6: Ao criar uma despesa, devem ser geradas as dívidas dos participantes envolvidos conforme os gastos informados.	
RF-7: O usuário deve conseguir visualizar as suas dívidas. E realizar seu pagamento.	
RF-8: O usuário deve conseguir gerar	

relatórios das dívidas de sua conta e despesas de seus grupo.	

Em relação aos requisitos funcionais, a prioridade foi definida seguindo o fluxo geral de dependências dos dados dentro do sistema: para criar um grupo, é necessário ter usuários cadastrados; para criar eventos, é necessário ter um grupo, etc. Já para os requisitos não funcionais, avaliamos que a nossa menor prioridade é o desempenho, focando mais em garantir que o sistema consiga manipular os dados de forma robusta, sem ter despesas, e atinja alguns critérios que julgamos importantes em termos de usabilidade.

1.3.Projeto



Disponível em: https://github.com/RafaelPetr/INF01120_Grupo2/blob/main/uml/UML.jpg

O diagrama foi definido após ser feita uma análise de como o SplitWise é estruturado, visando construir um sistema de menor escopo, mas que implementa as funcionalidades mais relevantes ao trabalho. Definimos a classe de “usuário”, que compõem grupos para que seja possível gerar as despesas. As despesas, por sua vez, podem ser de dois tipos (classes filhas): fixo (que ocorre recorrentemente em um dado período de tempo) ou variável (que ocorre só uma vez). As despesas são utilizadas para montar os objetos do tipo “dívida”, que, além da despesa que a gerou, armazena o usuário devedor e o credor. Além disso, pela funcionalidade de geração de relatórios, definimos uma classe geral de “relatório” que é estendida

para poder sobrescrever a função que o imprime, podendo, assim, construir uma lógica diferente para mostrar dados de usuário ou grupo.

1.4.Interface com o usuário

1 - perfil do usuário: mostra uma lista de grupos e um botão que leva para a lista de dívidas do usuário

2 - lista de dívidas do usuário: mostra uma lista de dívidas do usuário com um botão de pagamento do lado de cada item

3 - interface de grupo: mostra uma lista de despesas do grupo e botões para adicionar despesa, remover despesa e outro que leva para uma lista de usuários

4 - lista de usuários: mostra os usuários cadastrados no grupo e botões para adicionar e remover