

Uma imagem com Tipo de letra, captura de ecrã, Gráficos, logótipo

Descrição gerada automaticamente

**Elementos do Grupo:**

Gonçalo Fernandes – 20231215

Pedro Dias – 20230038

Tiago Rato – 20230931

Repositório no GitHub: <https://github.com/PedroDias0/ShareBill-Grupo-5>

**Quadro de Tarefas e Contributos:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarefa** | **Pedro Dias** | **Tiago Rato** | **Gonçalo Fernandes** |
| **Desenvolvimento da Api** | **0%** | **0%** | **100%** |
| **Design** | **0%** | **100%** | **0%** |
| **Base de dados** | **100%** | **0%** | **0%** |
| **Desenvolvimento da aplicação** | **0%** | **100%** | **0%** |
| **Words e powerpoints** | **100%** | **0%** | **0%** |

**Palavras-chave:**

Divisão de despesas, Gestão financeira, Share bill, Despesas compartilhadas, Grupos de amigos.

**Breve descrição da App e do problema que pretende resolver:**

O Share bill é uma aplicação que facilita a partilha de despesas entre grupos de pessoas, sejam amigos, familiares ou colegas de trabalho. Muitas vezes, a divisão de despesas em eventos sociais, viagens ou mesmo na gestão de uma casa partilhada, torna-se confusa e gera mal entendidos. O Share bill automatiza este processo, calculando de forma justa quanto cada pessoa deve pagar ou receber, ajudando a evitar conflitos e simplificando a organização financeira entre grupos.

**Descrição dos objetivos e da motivação do trabalho a realizar:**

O objetivo principal do projeto Share bill é desenvolver uma aplicação que simplifique a gestão de despesas compartilhadas, permitindo que os utilizadores adicionem e acompanhem despesas de forma eficiente. A motivação do trabalho surge da necessidade crescente de ferramentas digitais que automatizem a gestão financeira e melhorem a comunicação em situações de partilha de custos, como viagens ou eventos sociais. Esta aplicação será uma solução útil para reduzir o tempo e os conflitos associados à partilha de despesas.

**Identificação do Público-Alvo:**

O público-alvo do Share bill são grupos de amigos, estudantes, famílias, colegas de trabalho ou qualquer pessoa que precise de dividir despesas de forma justa e organizada. A aplicação destina-se a utilizadores que valorizam uma solução digital prática e intuitiva para gerir finanças pessoais e coletivas.

**Pesquisa de mercado sobre aplicações semelhantes ou com as quais pode comparar:**

1. **Splitwise** –A principal inspiração para o Share bill, permitindo dividir despesas entre grupos. É amplamente usada e reconhecida, mas o Share bill pretende oferecer uma interface mais simples e mais funções.

2. **Tricount** – Focado em simplificar a divisão de despesas para grupos, particularmente durante viagens. Similar ao Share bill, mas com menos funcionalidades avançadas.

3. **Settle Up** – Outra aplicação de divisão de contas, com foco em simplicidade e partilha de despesas de maneira rápida.

**Guia de Teste 1: Criar um Grupo para Dividir Despesas**

**• Objetivo:** Verificar se o utilizador consegue criar um grupo para dividir despesas corretamente.

**• Passos:**

1. O utilizador faz login na aplicação.

2. O utilizador seleciona "Criar novo grupo".

3. O utilizador insere o nome do grupo ("Viagem para Lisboa") e convida membros (escolhendo contatos).

4. O utilizador confirma a criação do grupo.

5. O sistema exibe o grupo recém-criado na lista de grupos.

**Guia de Teste 2:** **Adicionar uma Despesa ao Grupo**

**• Objetivo:** Verificar se o utilizador consegue adicionar uma despesa ao grupo de forma correta.

**• Passos:**

1. O utilizador seleciona um grupo existente ("Viagem para Lisboa").

2. O utilizador escolhe "Adicionar despesa" no grupo.

3. O utilizador insere o valor da despesa (€200) e a descrição (Alojamento).

4. O utilizador seleciona quem pagou e como deseja dividir a despesa (divisão igual).

5. O utilizador confirma a operação.

6. O sistema atualiza o saldo de cada membro do grupo.

**Guia de Teste 3: Marcar uma Dívida como Paga**

**• Objetivo:** Verificar se o utilizador consegue marcar uma dívida como paga.

**• Passos:**

1. O utilizador visualiza uma dívida pendente no grupo ("Pedro deve €50 a João"). 2. O utilizador clica na dívida e escolhe "Marcar como paga".

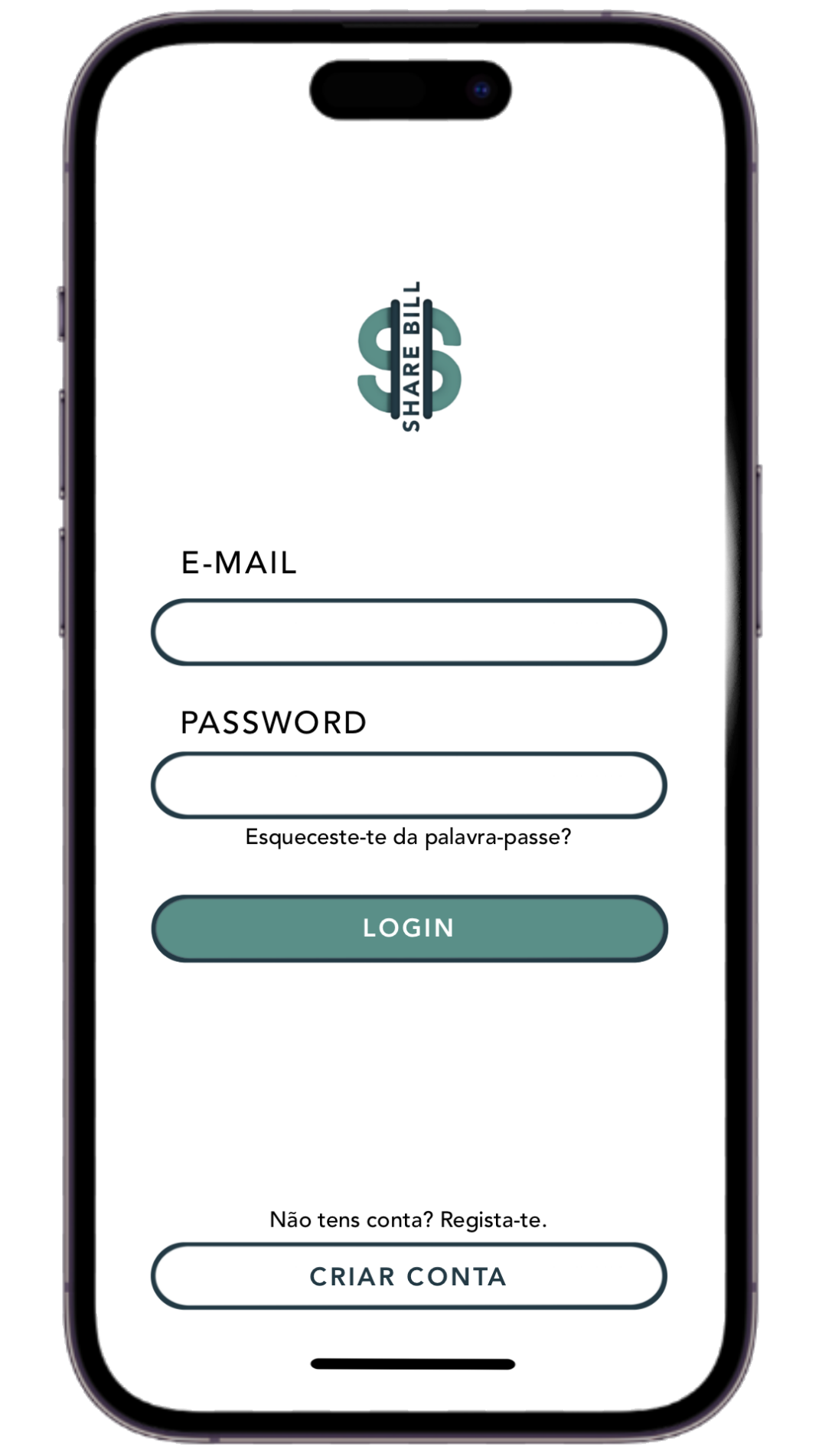
3. O sistema pergunta ao utilizador se deseja confirmar a operação. 4. O utilizador confirma. 5. O sistema atualiza o estado da dívida para "Paga".

**Design:**

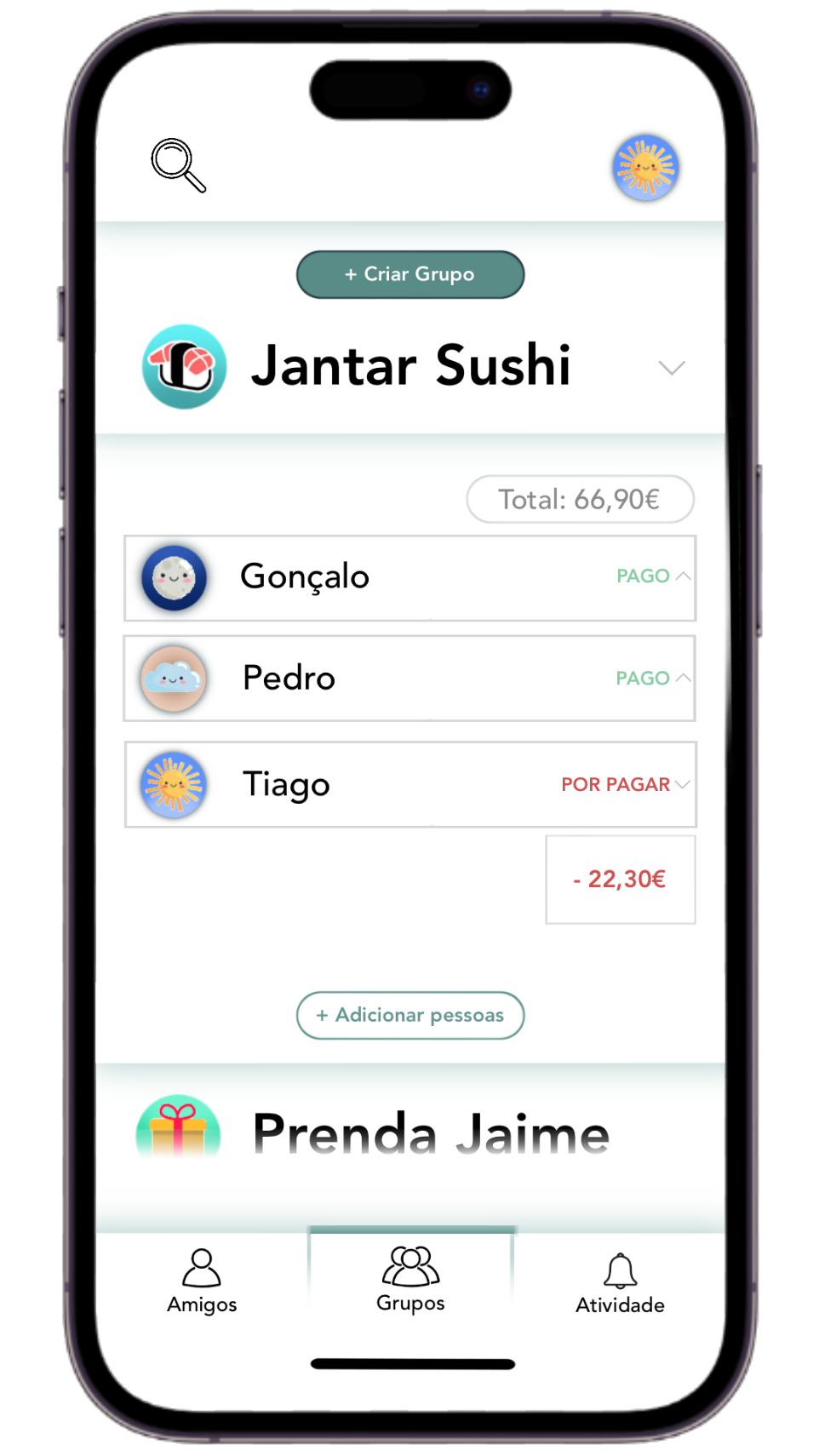
Uma imagem com captura de ecrã, texto, Telemóvel, Dispositivo móvel

Descrição gerada automaticamente

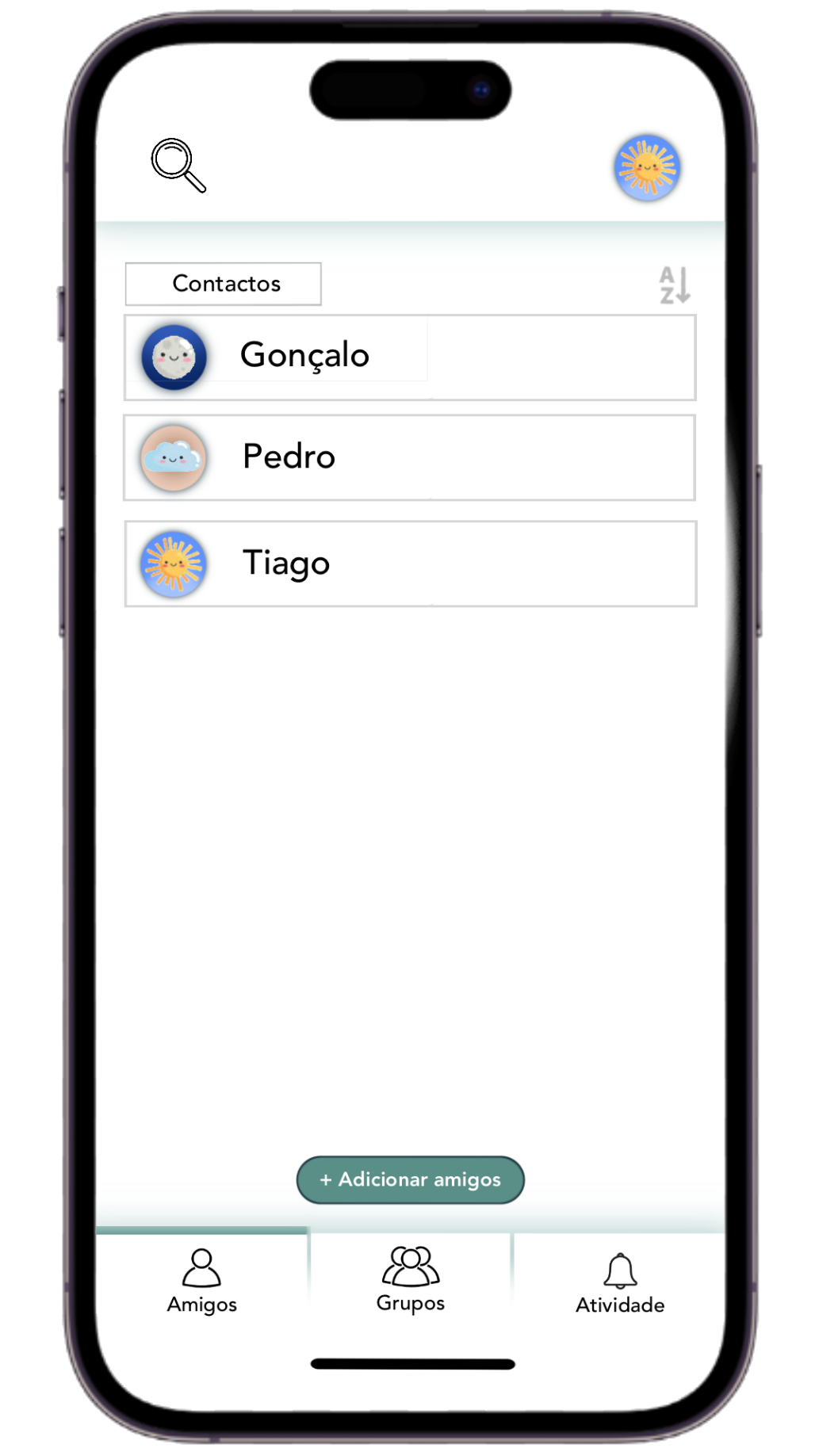
Criar Conta



Login



Grupos



Amigos



Atividades

**Descrição genérica da solução a implementar:**

O Share bill será uma aplicação mobile e web que permitirá a criação de grupos de despesas, a adição de pagamentos e a divisão automática dos custos entre os participantes. A solução inclui funcionalidades como lembretes automáticos, registo detalhado de transações e relatórios claros.

**Enquadramento nas diversas Unidades Curriculares:**

Programação de Dispositivos Móveis – Ensina a criar interfaces amigáveis para os utilizadores;

Programação Orientada a Objetos – fornece técnicas para modelar o comportamento e as interações dos objetos tornando o código mais organizado;

Bases de Dados- Responsável pela gestão eficiente dos dados da aplicação, como informações sobre utilizadores, grupos e despesas;

Competências Comunicacionais Responsável pelos planeamentos dos Words e powerpoints;

Matemática Discreta – contribui no desenvolvimento de algoritmos que organizamos dados e ajuda a tornar o código seja mais eficiente;

**Requisitos Técnicos para desenvolvimento do projeto:** https://developer.android.com/studio?hl=pt-br

<https://www.mysql.com>

<https://www.mamp.info/en/windows/>

**Arquitetura da Solução Android Studio**

- Desenvolvimento da interface;

Mysql – Armazenamento consulta e manipulação de dados;

Mamp - Servidor local, testes de integração;

Figma - Design

Rest:

## 1. AmigosController

Este controlador é responsável pela gestão de amizades no sistema. As funcionalidades incluem:

- Listar amizades aceites (`GET /api/amigos`).

- Listar amigos de um utilizador específico (`GET /api/amigos/user/{userId}`).

- Listar amizades pendentes de um utilizador (`GET /api/amigos/user/{userId}/pending`).

- Obter uma amizade específica pelo ID (`GET /api/amigos/{id}`).

- Criar uma nova amizade (`POST /api/amigos`).

- Eliminar uma amizade por ID (`DELETE /api/amigos/{id}`).

- Aceitar um pedido de amizade (`PATCH /api/amigos/{utilizador1Id}/accept/{utilizador2Id}`).

## 2. AtividadesController

Este controlador gere as operações relacionadas com atividades. As funcionalidades incluem:

- Listar todas as atividades (`GET /api/atividades`).

- Obter uma atividade por ID (`GET /api/atividades/{id}`).

## 3. CategoriasController

Gere as categorias. Inclui:

- Listar todas as categorias (`GET /api/categorias`).

- Obter uma categoria por ID (`GET /api/categorias/{id}`).

## 4. DespesasController

Este controlador lida com a gestão de despesas. As funcionalidades incluem:

- Listar todas as despesas (`GET /api/despesas`).

- Obter uma despesa específica pelo ID (`GET /api/despesas/{id}`).

## 5. GrupoController

Este controlador é responsável pela gestão de grupos:

- Listar todos os grupos (`GET /api/grupo`).

- Obter detalhes de um grupo por ID (`GET /api/grupo/{id}`).

## 6. UgController

Lida com operações entre utilizadores e grupos, incluindo:

- Criar um novo grupo (`POST /api/grupos/create`).

- Adicionar um utilizador a um grupo (`POST /api/grupos/{grupoId}/add-user/{userId}`).

- Remover um utilizador de um grupo (`DELETE /api/grupos/{grupoId}/remove-user/{userId}`).

- Eliminar um grupo (`DELETE /api/grupos/{grupoId}`).

- Listar utilizadores num grupo (`GET /api/grupos/{grupoId}/users`).

- Listar os grupos de um utilizador (`GET /api/grupos/user/{userId}/groups`).

## 7. UserController

Este controlador gere os utilizadores, incluindo:

- Listar todos os utilizadores (`GET /api/user`).

- Obter um utilizador por ID (`GET /api/user/{id}`).

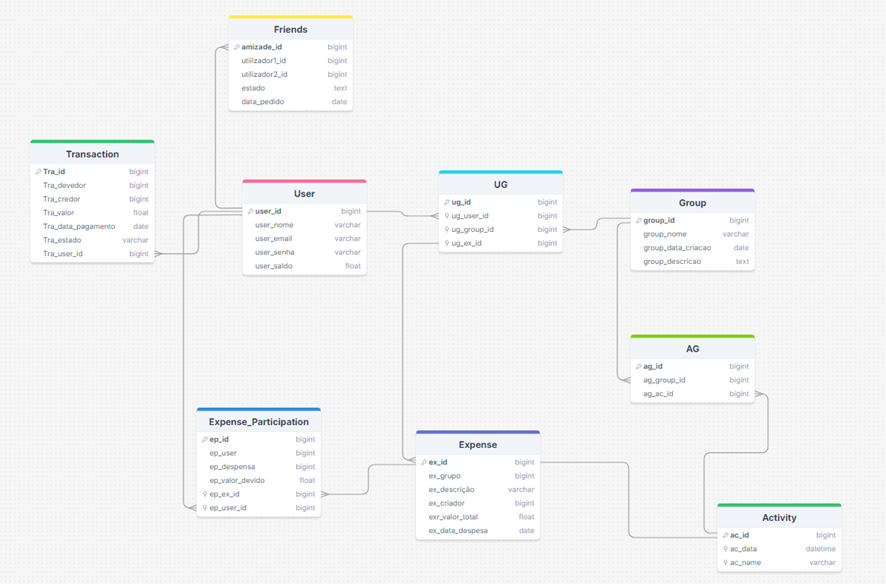
- Procurar utilizadores por nome (`GET /api/user/search`).

- Registar um novo utilizador (`POST /api/user/register`).

- Iniciar sessão e gerar token JWT (`POST /api/user/login`).

- Validar token JWT (`GET /api/user/validate`).

Modelo Entidade-Relação



1. Tabela User (Utilizadores)

Representa os utilizadores do sistema. Atributos: user\_id: Identificador único do utilizador. user\_nome: Nome do utilizador. user\_email: Email do utilizador. user\_senha: Palavra-passe do utilizador. user\_saldo: Saldo associado ao utilizador.

1. Tabela Friends (Amizades)

Representa as relações de amizade entre utilizadores. Atributos: amizade\_id: Identificador único da amizade. utilizador\_1\_id e utilizador\_2\_id: IDs dos dois utilizadores que são amigos. estado: Estado da amizade (por exemplo, "pendente" ou "aceite"). data\_pedido: Data em que o pedido de amizade foi feito. Relacionamento: Liga dois registos da tabela User para definir as amizades.

1. Tabela Group (Grupos)

Representa os grupos do sistema. Atributos: group\_id: Identificador único do grupo. group\_nome: Nome do grupo. group\_data\_criacao: Data de criação do grupo. group\_descricao: Descrição do grupo.

1. Tabela UG (Utilizadores em Grupos)

Representa a associação entre utilizadores e grupos. Atributos: ug\_id: Identificador único da associação. ug\_user\_id: ID do utilizador (chave estrangeira para User). ug\_group\_id: ID do grupo (chave estrangeira para Group). ug\_ex\_id: Relacionamento adicional, possivelmente associado a despesas ou outras atividades. Relacionamento: Liga a tabela User à tabela Group para registar que utilizadores pertencem a quais grupos.

1. Tabela Expense (Despesas)

Regista despesas realizadas em grupos. Atributos: ex\_id: Identificador único da despesa. ex\_grupo: ID do grupo associado à despesa. ex\_descricao: Descrição da despesa. ex\_criador: ID do utilizador que criou a despesa. ex\_valor\_total: Valor total da despesa. ex\_data\_despesa: Data em que a despesa foi realizada. Relacionamento: Liga despesas à tabela Group e ao criador da despesa na tabela User.

1. Tabela Expense\_Participation (Participação nas Despesas)

Representa a partilha de despesas entre utilizadores. Atributos: ep\_id: Identificador único da participação. ep\_despesa: ID da despesa (chave estrangeira para Expense). ep\_valor\_devido: Valor devido pelo utilizador. ep\_user\_id: ID do utilizador associado (chave estrangeira para User). Relacionamento: Liga a tabela Expense à tabela User para indicar como as despesas são divididas.

1. Tabela Transaction (Transações)

Regista transações financeiras entre utilizadores. Atributos: Tra\_id: Identificador único da transação. Tra\_devedor: ID do utilizador que deve (chave estrangeira para User). Tra\_credor: ID do utilizador que irá receber (chave estrangeira para User). Tra\_valor: Valor da transação. Tra\_data\_pagamento: Data do pagamento. Tra\_estado: Estado da transação (por exemplo, "pendente" ou "concluída"). Tra\_user\_id: ID do utilizador que registou a transação. Relacionamento: Liga dois registos da tabela User para registar transferências financeiras entre eles.

1. Tabela AG (Atividades em Grupos)

Regista a relação entre grupos e atividades. Atributos: ag\_id: Identificador único. ag\_group\_id: ID do grupo (chave estrangeira para Group). ag\_ac\_id: ID da atividade (chave estrangeira para Activity).

1. Tabela Activity (Atividades)

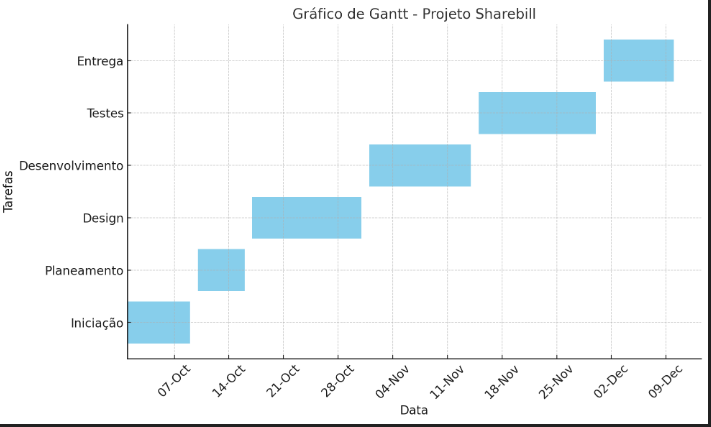
Regista atividades realizadas em grupos. Atributos: ac\_id: Identificador único da atividade. ac\_data: Data da atividade. ac\_name: Nome da atividade. Resumo dos Relacionamentos: Utilizadores têm amigos através da tabela Friends. Utilizadores pertencem a grupos através da tabela UG. Grupos têm despesas registadas na tabela Expense. Despesas são divididas entre os utilizadores na tabela Expense\_Participation. Transações financeiras entre utilizadores são geridas na tabela Transaction. Grupos podem ter atividades específicas, registadas nas tabelas AG e Activity.

**Tecnologias do Projeto:**

Kotlin;

Java;

Mysql;





Na primeira entrega do pre-projeto fizemos este gráfico de Gantt e distribuímos as tarefas neste modo, mas na distribuição final ficou diferente, o Pedro Dias ficou a tratar da base de dados e dos words do projeto, Gonçalo Fernandes ficou a tratar do desenvolvimento da API, e o Tiago Rato ficou com a parte do desenvolvimento mobile.

**Auto-avaliação da implementação do projeto:**

A nossa Auto-avaliação sobre o Projeto é que não conseguimos fazer a parte do Guia 3 porque gerimos muito mal o tempo e estivemos a fazer as coisas a ultima da hora, na parte dos words alguns deles não ficaram completamente completos porque não gerimos bem o tempo, no final achamos que deviamos ter melhorado melhor o nosso tempo para fazer o projeto e terminar por completo as nossas tarefas.

**Conclusão:**

O projeto Share bill visa criar uma aplicação que resolva de forma eficaz os problemas associados à partilha de despesas entre grupos. Os principais objetivos incluem a simplificação do processo de divisão de contas, a automatização de lembretes de pagamentos e a geração de relatórios detalhados. A aplicação será testada e otimizada para garantir uma experiência de utilizador clara e eficiente.

**Bibliografia:**

1. "Splitwise: Dividing Expenses for Friends and Groups." <https://www.splitwise.com>

2. "Tricount: Expense Sharing Made Easy." <https://www.tricount.com>

3. "Settle Up: Share Expenses with Friends." https://settleup.io