

---

# RELATÓRIO DO TRABALHO PRÁTICO

---

Meta 2



# Índice

<b>Índice</b>	<b>1</b>
<b>Introdução</b>	<b>2</b>
<b>Arquitetura do Sistema</b>	<b>2</b>
Visão Geral	2
<b>Serviço RMI</b>	<b>2</b>
Server-side	2
getUsers	2
getGroups	2
addObserver	3
removeObserver	3
Client-side	3
onRegiste	3
onLogin	3
onInsertExpense	3
onDeleteExpense	3
<b>Serviço API REST</b>	<b>3</b>
Tabelas do endpoints	3
Autenticador	5
/register	5
POST	5
/login	6
POST	6
Grupos	6
/groups	6
GET	6
POST	7
/groups/{groupId}	8
GET	8
PUT	8
/groups/{groupId}/expenses	9
GET	9
POST	10
/groups/{groupId}/expenses/{expenseId}	11
DELETE	11
<b>Conclusão</b>	<b>12</b>

# Introdução

Neste trabalho pretende adicionar em Java 3 funcionalidades: *serviço API REST* e *serviço RMI*. Assim, foi criado 3 novos aplicativos: *ServerAPI*, *ClientAPI* e *ClientRMI*

As aplicações *ClientAPI* e *ClientRMI* são aplicações de consola simples para testar o *API REST* e *RMI* em Java sem precisar de utilizador alguma aplicação externo.

A aplicação *ServerAPI* utiliza *Spring Boot* framework para facilitar a criação e configuração dos endpoints para este trabalho.

A funcionalidade *RMI* está presente dentro do Servidor da meta 1 para assim puder receber as informações pretendidas.

## Arquitetura do Sistema

### Visão Geral

O servidor API vai buscar o ficheiro da base de dados da meta 1 e a classe *DataBaseManager* com a lógica já feita para aceder e fazer as operações CRUD na base de dados, mas não tendo nenhuma ligação com o servidor da meta 1.

O serviço RMI foi adicionado ao servidor da meta 1 para assim poder saber os eventos de registo, autenticação, inserção de despesa e eliminação de despesa.

Com isto, o servidor API e serviço RMI não têm nenhuma ligação entre eles.

O clientAPI é uma aplicação simples que faz os pedidos utilizado o package *java.net* para fazer os seus pedidos e mostrar os dados brutos no terminal sem ter a necessidade de transformar em objetos que não seriam utilizados em runtime no lado do cliente, só servem com fim de mostrar os dados.

O clientRMI é uma aplicação simples semelhante a um monitor com os 2 comandos de pedidos para listar os utilizadores e grupos registados.

## Serviço RMI

### Server-side

#### *getUsers*

Função que devolve a lista de utilizadores registado na base de dados

#### *getGroups*

Função que devolve a lista de grupos registado na base de dados

## addObserver

Função que adiciona o clientAPI que invocou à sua lista para assim puder notificar quando um dos eventos ocorre.

## removeObserver

Função que remove o clientAPI que ou fez o pedido de sair, ou caso de erro e não estar a funcionar quando tentar notificação o ocorrido evento.

## Client-side

### onRegiste

Função que mostrara informação do novo registo feito no servidor.

### onLogin

Função que mostrara informação da nova autenticação feita no servidor.

### onInsertExpense

Função que mostrara informação do novo registo de despesa feita no servidor.

### onDeleteExpense

Função que mostrara informação da eliminação da despesa feita no servidor.

## Serviço API REST

### Tabelas do endpoints

Method	URI	Operation	Description	Request Body	Response Body
POST	/register	Create	Regista um novo utilizador	User object	"User created" ou mensagem de erro associada ao estado http

Method	URI	Operation	Description	Request Body	Response Body
POST	/login	Authentication	Gerar token de autenticação	None (uses authentication object)	JWT
GET	/groups	Read	Obtenha todos os grupos de utilizadores com filtragem opcional (por nome, número de utilizadores)	None	List de PreviewGroupDTO
POST	/groups	Create	Crie um novo grupo	Group object	PreviewGroupDTO do grupo criado ou mensagem de erro associado ao estado http
GET	/groups/{groupid}	Read	Recuperar um grupo específico por ID	None	Group
PUT	/groups/{groupid}	Update	Atualizar as informações de um grupo existente	Group object	Group atualizado
GET	/groups/{groupid}/expenses	Read	Obtenha todas as despesas	None	Lista de Expense

Method	URI	Operation	Description	Request Body	Response Body
			do grupo com a filtragem opcional (por título, valor, data, registo, pagador, utilizadores associados)		
POST	/groups/{groupid}/expenses	Create	Criar uma nova despesa num grupo	Expense object	Expense criado
DELETE	/groups/{groupid}/expenses/{expenseid}	Delete	Excluir uma despesa específica de um grupo	None	"Expense deleted" ou mensagem de erro associado ao estado http

## Autenticador

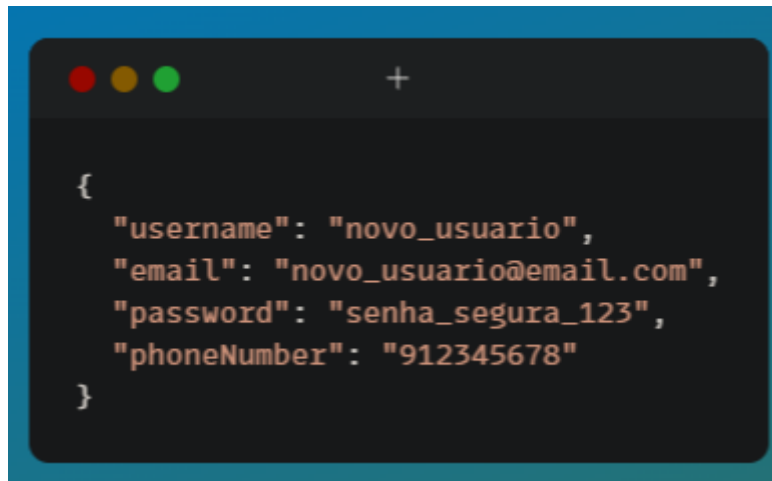
O controlador *AuthController* é responsável pelos endpoints de autenticação.

### /register

#### POST

**Descrição:** Regista um novo utilizador

**Pedido:**



## /login

### POST

**Descrição:** Autentica utilizador o email e password.

**Resposta:** JWT

## Grupos

O controlador *GroupController* é responsável pelos endpoints dos grupos e das despesas, pois as despesas só existem dentro de grupos, por isso não faz sentido existir um controlador dedicado para despesas quando são dependes do grupo.

## /groups

### GET

**Descrição:** Lista os grupos, omitindo algumas informações para minimizar a informação enviada, em que o utilizador autenticado faz parte e tempo a mecânica de filtro da pesquisa de grupos.

**Parâmetros de Filtragem:**

- name: Filtrar por nome do grupo
- numUsers: Filtrar pelo número de utilizadores

**Resposta:**

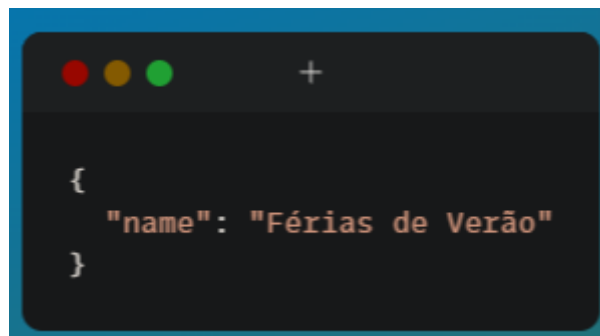


```
[
  {
    "id": 1,
    "name": "Viagem com Amigos",
    "numUsers": 4
  },
  {
    "id": 2,
    "name": "Casa",
    "numUsers": 3
  }
]
```

## POST

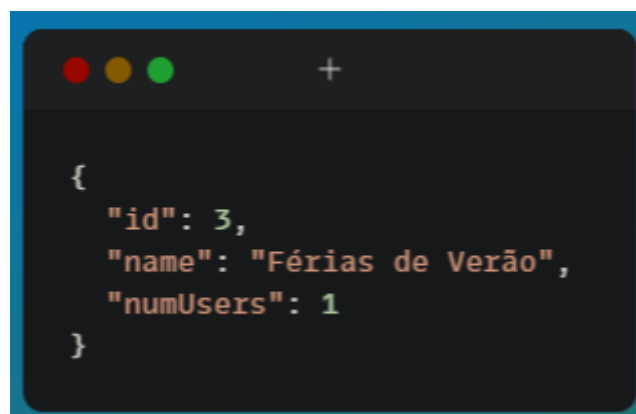
**Descrição:** Insere um novo grupo indicando o nome desse grupo e automaticamente o utilizador autenticado é associado a esse grupo criado.

**Pedido:**



```
{
  "name": "Férias de Verão"
}
```

**Resposta:**



```
{
  "id": 3,
  "name": "Férias de Verão",
  "numUsers": 1
}
```



## /groups/{groupId}

### GET

**Descrição:** Mostra os dados completos do grupo selecionado utilizando o id desse grupo.

**Variável de caminho:**

- groupId: id do grupo

**Resposta:**



```
{
  "id": 1,
  "name": "Viagem com Amigos",
  "numUsers": 4,
  "members": [
    {"id": 1, "email": "joao@email.com"},
    {"id": 2, "email": "maria@email.com"}
  ],
  "expenses": [
    {
      "id": 5,
      "title": "Aluguer de Carro",
      "amount": 250,
      "date": "2024-07-15"
    }
  ]
}
```

### PUT

**Descrição:** Atualiza o nome do grupo selecionado pelo id.

**Variável de caminho:**

- groupId: id do grupo

**Pedido:**

```
{
  "name": "Viagem com Amigos 2024"
}
```

**Resposta:**

```
{
  "id": 1,
  "name": "Viagem com Amigos 2024",
  "numUsers": 4
}
```

## /groups/{groupId}/expenses

### GET

**Descrição:** Lista as despesas do grupo em que o utilizador autenticado faz parte e tempo a mecânica de filtro da pesquisa de despesas

**Variável de caminho:**

- groupId: id do grupo

**Parâmetros de Filtragem:**

- title: Filtrar por título da despesa
- amount: Filtrar por valor
- date: Filtrar por data
- register: Filtrar por utilizador que registou
- payer: Filtrar por utilizador que pagou
- associated: Filtrar por utilizadores associados

```
[
  {
    "id": 5,
    "title": "Aluguer de Carro",
    "amount": 250,
    "date": "2024-07-15",
    "registerByUser": "joao@email.com",
    "payerUser": "maria@email.com",
    "associatedUsersList": [
      "joao@email.com",
      "pedro@email.com"
    ]
  },
  {
    "id": 6,
    "title": "Jantar",
    "amount": 120,
    "date": "2024-07-16",
    "registerByUser": "pedro@email.com",
    "payerUser": "joao@email.com",
    "associatedUsersList": [
      "maria@email.com",
      "ana@email.com"
    ]
  }
]
```

## POST

**Descrição:** Insere uma nova despesa ao grupo selecionado pelo id.

**Variável de caminho:**

- groupId: id do grupo

**Pedido:**

```
{
  "title": "Gasolina",
  "amount": 75,
  "date": "2024-07-20",
  "payerUser": "maria@email.com",
  "associatedUsersList": [
    "joao@email.com",
    "pedro@email.com"
  ]
}
```

Resposta:

```
{
  "id": 7,
  "title": "Gasolina",
  "amount": 75,
  "date": "2024-07-20",
  "registerByUser": "joao@email.com",
  "payerUser": "maria@email.com",
  "associatedUsersList": [
    "joao@email.com",
    "pedro@email.com"
  ]
}
```

[/groups/{groupId}/expenses/{expenseId}](#)

## DELETE

**Descrição:** Apaga a despesa selecionada pelo seu id no grupo selecionado pelo seu id.

**Variável de caminho:**

- groupId: id do grupo
- expenseId: id da despesa que vai ser eliminada

## Conclusão

Na entrega deste trabalho foi fornecido um *wrapper* do *gradle* para poder executar os comandos *gradle* numa máquina em que não esteja instalada o *gradle*.

Foi tomada essa decisão, pois o tamanho total do projeto seria maior que a capacidade possível que o InforEstudante consegue aguentar (capacidade máxima: 100MB e o projeto com executáveis ou ficheiros *bytecodes* tem >200MB e o ficheiro comprimido não era o suficiente para diminuir mais de metade)

Antes da entrega, foi testado este mesmo *wrapper* numa máquina nova com Linux só com Java 21 JDK e foi executada como deve ser, sendo esperado que execute como esperado.

Todo o funcionamento do *gradle wrapper* está presente na [documentação do gradle](#).

O JSON não permite o uso de dados do tipo *double* e como não era o ideal mudar a classe *Expense* para dividir o que seria o número inteiro e décimas, foi optado o uso de números inteiros para simplificar o trabalho, mas não sendo uma situação realista.

Como o pedido desta meta era simples, foi decidido criar uns endpoints para explorar outras funcionalidades, assim não precisar de utilizar outras formas para modificar melhor os dados.