1. POST /login - Autenticar usuário

Descrição: Realiza o login de um usuário no sistema. Suporta resposta em HTML ou JSON.
Método HTTP: POST
Endpoint: `/controller.do?action=login`

Cabeçalhos opcionais:
`Accept: application/json` ou parâmetro `?format=json` para resposta JSON.

Corpo da requisição (form-urlencoded):

```
email=usuario@email.com
password=123456
```

Corpo da requisição (JSON):

```
{
  "email": "usuario@email.com",
  "password": "123456"
}
```

Códigos de resposta:

- **200 OK**
    - Login bem-sucedido (HTML redirecionado ou JSON retornado com dados do usuário).

JSON de resposta:

```
{
  "message": "Login successful",
  "userName": "João da Silva",
  "userEmail": "joao@email.com"
}
```

    -
- **302 Found**
    - Redireciona para a página principal (`transactions-page`) em caso de login via formulário.
- **401 Unauthorized**

- Login inválido (credenciais incorretas).

JSON de erro:

```
{
  "error": "Invalid login"
}
```

- **500 Internal Server Error**
  - Erro interno ao tentar autenticar o usuário.

2. POST /register - Criar novo usuário

Descrição: Cadastra um novo usuário no sistema. Suporta resposta em HTML ou JSON.
Método HTTP: POST
Endpoint: `/controller.do?action=register`

Cabeçalhos opcionais:
`Accept: application/json` ou parâmetro `?format=json` para resposta JSON.

Corpo da requisição (form-urlencoded):

```
name=João da Silva
email=joao@email.com
password=123456
```

Corpo da requisição (JSON):

```
{
  "name": "João da Silva",
  "email": "joao@email.com",
  "password": "123456"
}
```

Códigos de resposta:

- **201 Created** – Usuário cadastrado com sucesso.
  JSON de sucesso:

```
{
  "message": "User registered successfully.",
  "userId": 42
}
```

- **302 Found** – Redirecionamento para a página de transações em caso de sucesso via formulário HTML.

- **409 Conflict** – E-mail já está em uso.
  JSON de erro:

```
{
  "error": "Email already in use."
}
```

- **500 Internal Server Error** – Erro ao tentar registrar o usuário.
  JSON de erro:

```
{
  "error": "Failed to complete the registration. Please try again."
}
```

3. GET /logout - Finalizar sessão do usuário

Descrição: Finaliza a sessão do usuário autenticado. Suporta resposta em HTML ou JSON.
Método HTTP: GET
Endpoint: `/controller.do?action=logout`

Cabeçalhos opcionais:
`Accept: application/json` ou parâmetro `?format=json` para resposta JSON.

Códigos de resposta:

200 OK – Logout realizado com sucesso.
JSON de sucesso:

```
{
  "message": "Logout successful"
}
```

200 OK (HTML) – Encaminha para a página `index.jsp` após logout.

4. GET /transactions - Listar transações do usuário

Descrição: Retorna uma lista de transações realizadas pelo usuário logado, com suporte a filtros e paginação. Resposta em JSON ou HTML.
Método HTTP: GET
Endpoint: `/controller.do?action=transactions-page`

Parâmetros de consulta (query params):

- `year` (opcional): Ano para filtrar transações (ex: `2025`)
- `month` (opcional): Mês para filtrar transações (1 a 12)
- `type` (opcional): Tipo da transação (`INCOME` ou `EXPENSE`)
- `category` (opcional): Categoria da transação (ex: `FOOD`, `ENTERTAINMENT`, `HEALTH`, etc)
- `page` (opcional): Número da página (padrão: `1`)
- `size` (opcional): Quantidade de itens por página (padrão: `10`)

Cabeçalhos opcionais:

- `Accept: application/json` ou `?format=json` (não implementado nesse comando) para resposta JSON.

Exemplo de resposta JSON (200 OK):

```
{
  "transactions": [
    {
      "id": 42,
      "payerId": 1,
      "receiverId": 3,
      "price": 100.0,
      "description": "Compra no mercado",
      "type": "EXPENSE",
      "category": "FOOD",
      "dateTime": "2025-07-13T10:45:00"
    }
  ],
  "currentPage": 1,
  "totalPages": 5
}
```

Códigos de resposta:

- **200 OK** – Lista de transações retornada com sucesso.
- **400 Bad Request** – Parâmetro inválido (ex: `year`, `month`, `type`, `category`)
- **401 Unauthorized** – Usuário não está autenticado.
- **500 Internal Server Error** – Erro interno ao buscar os dados.

5. GET /transactions-summary - Resumo financeiro do usuário

Descrição: Retorna um resumo das transações financeiras do usuário logado, incluindo total de receitas, despesas, saldo atual e agrupamento por categoria.
Método HTTP: GET
Endpoint: `/controller.do?action=transactions-summary-page`

Cabeçalhos opcionais:

- `Accept: application/json` — para obter resposta em JSON

Exemplo de resposta JSON (200 OK):

```
{
  "totalIncome": 5000.00,
  "totalExpenses": 3200.00,
  "currentBalance": 1800.00,
  "expensesByCategory": {
    "FOOD": 1200.00,
    "RENT": 1500.00,
    "ENTERTAINMENT": 500.00
  },
  "incomeByCategory": {
    "SALARY": 4000.00,
    "FREELANCE": 1000.00
  }
}
```

Códigos de resposta:

- **200 OK** – Resumo retornado com sucesso.
- **401 Unauthorized** – Usuário não está autenticado.
- **500 Internal Server Error** – Erro ao buscar o resumo no banco.

6. POST /transactions - Criar nova transação

Descrição: Registra uma nova transação financeira entre usuários. O usuário deve estar autenticado.
Método HTTP: POST
Endpoint: `/controller.do?action=new-transaction`

Cabeçalhos opcionais:

- `Accept: application/json` — para receber a resposta em JSON

Corpo da requisição (JSON):

```
{
  "receiverEmail": "usuario2@example.com",
  "price": "250.75",
  "description": "Compra de livros",
  "type": "EXPENSE",
  "category": "EDUCATION"
}
```

> Também é possível enviar via formulário `application/x-www-form-urlencoded` com os mesmos campos.

Códigos de resposta:

- **201 Created** – Transação criada com sucesso.
- **400 Bad Request** – Campos ausentes, valores inválidos ou tentativa de transferência para o próprio e-mail.
- **401 Unauthorized** – Usuário não autenticado.

- **404 Not Found** – E-mail do destinatário não encontrado.
- **500 Internal Server Error** – Erro ao persistir a transação.

7. GET /transactions/edit - Buscar dados de uma transação para edição

Descrição: Recupera os dados de uma transação específica para edição. Retorna os dados em formato JSON ou exibe a tela de edição, dependendo do cabeçalho `Accept`.
Método HTTP: GET
Endpoint: `/controller.do?action=edit-transaction-page`

Parâmetros de consulta:

- `id` (obrigatório) — ID da transação a ser consultada

Cabeçalhos opcionais:

- `Accept: application/json` — para resposta JSON

Exemplo de requisição:

```
GET /controller.do?action=edit-transaction-page&id=5
Accept: application/json
```

Resposta (JSON):

```
{
  "id": 5,
  "payerId": 1,
  "receiverId": 2,
  "price": 250.75,
  "description": "Compra de livros",
  "type": "EXPENSE",
  "category": "EDUCATION",
  "dateTime": "2025-07-13T10:23:00",
  "receiverEmail": "usuario2@example.com"
}
```

Códigos de resposta:

- **200 OK** – Transação encontrada com sucesso.
- **400 Bad Request** – Parâmetro `id` ausente ou inválido.
- **404 Not Found** – Transação não encontrada.
- **500 Internal Server Error** – Erro inesperado no servidor.

8. PUT /transactions - Editar transação

Descrição: Atualiza os dados de uma transação existente. Disponível em formato JSON e também via formulário HTML.
Método HTTP: PUT
Endpoint: `/controller.do?action=edit-transaction`

Corpo da requisição (JSON):

```
{
  "id": 5,
  "receiverEmail": "usuario2@example.com",
  "price": "150.50",
  "description": "Pagamento mensal",
  "type": "EXPENSE",
  "category": "HOUSING"
}
```

Todos os campos são obrigatórios.

Resposta de sucesso (JSON):

```
{
  "message": "Transaction updated successfully."
}
```

Códigos de resposta:

- **200 OK** – Transação atualizada com sucesso.

- **400 Bad Request** – Campos ausentes, inválidos ou operação não autorizada.
- **401 Unauthorized** – Usuário não autenticado.
- **404 Not Found** – Transação ou destinatário não encontrado.
- **500 Internal Server Error** – Erro inesperado no servidor.

9. DELETE /transactions - Excluir transação

Descrição: Remove uma transação cadastrada pelo usuário autenticado.
Método HTTP: DELETE
Endpoint: `/controller.do?action=delete-transaction`

Corpo da requisição (JSON):

```
{
  "id": 5
}
```

> Alternativamente, o ID também pode ser enviado como parâmetro de URL:
> `/controller.do?action=delete-transaction&id=5`

Códigos de resposta:

- **204 No Content** – Transação excluída com sucesso.
- **400 Bad Request** – ID inválido ou ausente.
- **401 Unauthorized** – Usuário não autenticado.
- **403 Forbidden** – Tentativa de exclusão de transação pertencente a outro usuário.
- **404 Not Found** – Transação não encontrada.
- **500 Internal Server Error** – Falha ao excluir a transação.

10. GET /transactions/id - Buscar transação por ID

Descrição: Retorna os dados de uma transação específica pertencente ao usuário autenticado.
Método HTTP: GET
Endpoint: `/transacoes/id?id={id}`

Parâmetros de consulta:

- `id` (long): ID da transação a ser buscada.

Cabeçalhos esperados:

- `Accept: application/json`
- `Cookie: JSESSIONID=...` (para sessão autenticada)

Exemplo de requisição:

```
GET /transacoes/id?id=123
Accept: application/json
```

Resposta (JSON):

```
{
  "id": 123,
  "payerId": 1,
  "receiverId": 2,
  "price": 150.00,
  "description": "Aluguel",
  "type": "EXPENSE",
  "category": "HOUSING",
  "dateTime": "2025-07-13T10:40:00",
  "receiverEmail": "exemplo@dominio.com"
}
```

Códigos de resposta:

- **`200 OK`** – Transação encontrada com sucesso.
- **`400 Bad Request`** – ID inválido ou ausente.
- **`401 Unauthorized`** – Usuário não autenticado.
- **`403 Forbidden`** – Tentativa de acesso a transação de outro usuário.
- **`404 Not Found`** – Transação não encontrada.
- **`500 Internal Server Error`** – Erro inesperado no servidor.