

Documentação da API Mobile do ModularCompany

Visão Geral

Esta documentação descreve os endpoints específicos criados para aplicativos móveis se comunicarem com a API do ModularCompany. Esses endpoints foram projetados para funcionar com aplicativos React Native e outras plataformas móveis.

Base URL

```
https://modularcompany.vercel.app/api
```

Autenticação

Login

Endpoint: `/mobile-auth`

Método: `POST`

Body:

```
{
  "email": "usuario@exemplo.com",
  "password": "senha123"
}
```

Resposta de Sucesso (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "123456",
    "name": "Nome do Usuário",
    "email": "usuario@exemplo.com",
    "role": "EMPLOYEE",
    "companyId": "789012"
  }
}
```

Resposta de Erro (401):

```
{
  "error": "Credenciais inválidas"
}
```

Utilização do Token

Após receber o token, inclua-o no cabeçalho **Authorization** para todas as requisições subsequentes:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

O token expira após 24 horas, exigindo um novo login.

Endpoints

Perfil do Usuário

Endpoint: `/mobile-profile`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Resposta de Sucesso (200):

```
{
  "user": {
    "id": "123456",
    "name": "Nome do Usuário",
    "email": "usuario@exemplo.com",
    "role": "EMPLOYEE",
    "companyId": "789012",
    "hourlyRate": 50,
    "createdAt": "2023-01-15T10:30:00",
    "company": {
      "id": "789012",
      "name": "Empresa Exemplo",
      "plan": "STANDARD"
    }
  }
}
```

Registros de Horas

Listar Registros

Endpoint: `/mobile-time-entries`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters (opcionais):

- **startDate:** Data inicial no formato ISO (YYYY-MM-DD)
- **endDate:** Data final no formato ISO (YYYY-MM-DD)
- **userId:** ID do usuário específico (apenas para ADMIN, MANAGER e DEVELOPER)
- **approved:** Filtrar por status de aprovação (true/false)
- **rejected:** Filtrar por status de rejeição (true/false)
- **project:** Filtrar por nome do projeto
- **minHours:** Filtrar por horas mínimas
- **maxHours:** Filtrar por horas máximas
- **unpaid:** Filtrar apenas registros não pagos (true/false)
- **page:** Número da página (padrão: 1)
- **limit:** Registros por página (padrão: 50)
- **sortBy:** Campo para ordenação (date, hours, createdAt)
- **sortOrder:** Direção da ordenação (asc, desc)

Se não especificado, retorna registros do mês atual.

Controle de Acesso por Papel:

- **DEVELOPER:** Acesso completo a todos os registros do sistema
 - Pode ver registros de qualquer usuário
 - Pode filtrar por userId específico
- **ADMIN:** Acesso aos registros da sua empresa
 - Pode ver registros de todos os usuários da mesma empresa
 - Pode filtrar por userId específico (desde que o usuário pertença à mesma empresa)
 - Não pode ver registros de outras empresas
- **MANAGER:** Acesso aos registros da sua equipe
 - Pode ver registros de todos os usuários da mesma empresa
 - Pode filtrar por userId específico (desde que o usuário pertença à mesma empresa)
 - Não pode ver registros de outras empresas
- **EMPLOYEE:** Acesso restrito aos próprios registros

- Pode ver apenas seus próprios registros
- Não pode utilizar o parâmetro userId

Resposta de Sucesso (200):

```
{
  "timeEntries": [
    {
      "id": "entry123",
      "date": "2023-04-15",
      "startTime": "2023-04-15T09:00:00",
      "endTime": "2023-04-15T17:00:00",
      "totalHours": 8,
      "observation": "Desenvolvimento de features",
      "project": "ModularCompany",
      "approved": true,
      "rejected": null,
      "rejectionReason": null,
      "userId": "123456",
      "user": {
        "id": "123456",
        "name": "Nome do Usuário",
        "email": "usuario@exemplo.com",
        "hourlyRate": 50,
        "companyId": "789012"
      },
      "createdAt": "2023-04-15T08:50:00",
      "updatedAt": "2023-04-15T17:10:00"
    }
  ],
  "period": {
    "startDate": "2023-04-01",
    "endDate": "2023-04-30"
  },
  "pagination": {
    "total": 45,
    "page": 1,
    "limit": 50,
    "pages": 1
  },
  "stats": {
    "approved": 30,
    "pending": 10,
    "rejected": 5,
    "total": 45
  },
  "appliedFilters": {
    "approved": null,
    "rejected": null,
    "project": null,
    "minHours": null,
  }
}
```

```
"maxHours": null,  
"unpaid": null,  
"sortBy": "date",  
"sortOrder": "desc"  
}  
}
```

Exemplos de Uso:

1. **ADMIN** - Visualizar todos os registros da empresa no mês atual:

```
GET /mobile-time-entries
```

2. **ADMIN** - Visualizar registros de um funcionário específico:

```
GET /mobile-time-entries?userId=123456
```

3. **ADMIN/MANAGER** - Visualizar apenas registros aprovados em um período específico:

```
GET /mobile-time-entries?startDate=2023-05-01&endDate=2023-05-31&approved=true
```

4. **EMPLOYEE** - Visualizar seus próprios registros pendentes:

```
GET /mobile-time-entries?approved=null&rejected=null
```

Notas Importantes:

- Para ADMIN e MANAGER, se nenhum userId for especificado, serão retornados todos os registros de usuários da mesma empresa
- Para EMPLOYEE, o parâmetro userId é ignorado, pois só podem ver seus próprios registros
- O parâmetro unpaid=true filtra apenas registros que não foram associados a pagamentos

Criar Registro

Endpoint: /mobile-time-entries

Método: POST

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "date": "2023-04-16",
  "startTime": "2023-04-16T09:00:00",
  "endTime": "2023-04-16T17:00:00",
  "observation": "Implementação de nova feature",
  "project": "ModularCompany Mobile"
}
```

Resposta de Sucesso (201):

```
{
  "timeEntry": {
    "id": "entry124",
    "date": "2023-04-16",
    "startTime": "2023-04-16T09:00:00",
    "endTime": "2023-04-16T17:00:00",
    "totalHours": 8,
    "observation": "Implementação de nova feature",
    "project": "ModularCompany Mobile",
    "approved": null,
    "rejected": null,
    "rejectionReason": null,
    "userId": "123456",
    "createdAt": "2023-04-16T08:55:00",
    "updatedAt": "2023-04-16T08:55:00"
  }
}
```

Resposta de Erro - Conflito (409):

```
{
  "error": "Existe um conflito de horário com um registro existente",
  "conflictingEntry": {
    "id": "entry123",
    "date": "2023-04-16",
    "startTime": "08:30",
    "endTime": "12:30"
  }
}
```

Visualizar Registro

Endpoint: `/mobile-time-entries/[id]/view`

Método: GET

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Resposta de Sucesso:

```
{
  "timeEntry": {
    "id": "123456",
    "date": "2023-03-15",
    "startTime": "2023-03-15T09:00:00",
    "endTime": "2023-03-15T17:00:00",
    "totalHours": 8,
    "observation": "Trabalho no projeto A",
    "project": "Projeto A",
    "userId": "789012",
    "userName": "Nome do Usuário",
    "approved": true,
    "rejected": false,
    "rejectionReason": null,
    "createdAt": "2023-03-15T08:00:00",
    "updatedAt": "2023-03-15T18:00:00",
    "payment": {
      "id": "456789",
      "amount": 200,
      "date": "2023-03-30",
      "reference": "PAG123",
      "description": "Pagamento de março",
      "status": "completed"
    }
  }
}
```

Resposta de Erro:

```
{
  "error": "Registro não encontrado"
}
```

Editar Registro

Endpoint: /mobile-time-entries/[id]

Método: PUT

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "date": "2023-03-15",
  "startTime": "2023-03-15T10:00:00",
  "endTime": "2023-03-15T18:00:00",
  "observation": "Trabalho no projeto B",
  "project": "Projeto B"
}
```

Resposta de Sucesso:

```
{
  "timeEntry": {
    "id": "123456",
    "date": "2023-03-15",
    "startTime": "2023-03-15T10:00:00",
    "endTime": "2023-03-15T18:00:00",
    "totalHours": 8,
    "observation": "Trabalho no projeto B",
    "project": "Projeto B",
    "approved": null,
    "rejected": null,
    "rejectionReason": null,
    "createdAt": "2023-03-15T08:00:00",
    "updatedAt": "2023-03-15T09:00:00"
  }
}
```

Resposta de Erro:

```
{
  "error": "Não é possível editar um registro já aprovado"
}
```

ou

```
{
  "error": "Existe um conflito de horário com um registro existente",
}
```



```
"conflictingEntry": {
  "id": "789012",
  "date": "2023-03-15",
  "startTime": "09:00",
  "endTime": "11:00"
}
```

Excluir Registro

Endpoint: `/mobile-time-entries/[id]`

Método: `DELETE`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Resposta de Sucesso:

```
{
  "message": "Registro excluído com sucesso",
  "id": "123456"
}
```

Resposta de Erro:

```
{
  "error": "Não é possível excluir um registro já aprovado"
}
```

ou

```
{
  "error": "Este registro já está associado a um pagamento e não pode ser excluído"
}
```

Pagamentos

Listar Pagamentos (Funcionário)

Endpoint: `/mobile-payments`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Descrição: Retorna os pagamentos recebidos pelo **usuário autenticado** (funcionário).

Query Parameters (opcionais):

- **startDate:** Data inicial no formato ISO (YYYY-MM-DD)
- **endDate:** Data final no formato ISO (YYYY-MM-DD)
- **status:** Status do pagamento (ex: "completed", "pending")

Se não especificado, retorna pagamentos do mês atual.

Resposta de Sucesso (200):

```
{
  "payments": [
    {
      "id": "payment123",
      "amount": 400,
      "date": "2023-04-30",
      "description": "Pagamento mensal - Abril 2023",
      "reference": "PAG-2023-04",
      "paymentMethod": "bank_transfer",
      "status": "completed",
      "periodStart": "2023-04-01",
      "periodEnd": "2023-04-30",
      "totalHours": 40,
      "createdBy": {
        "id": "manager456",
        "name": "Gerente da Silva"
      },
      "timeEntries": [
        {
          "id": "entry123",
          "date": "2023-04-15",
          "totalHours": 8,
          "observation": "Desenvolvimento de features",
          "project": "ModularCompany",
          "amount": 80
        },
        {
          "id": "entry124",
          "date": "2023-04-16",
          "totalHours": 8,
```

```

        "observation": "Implementação de API",
        "project": "ModularCompany",
        "amount": 80
    }
],
"createdAt": "2023-04-30T18:00:00",
"updatedAt": "2023-04-30T18:00:00"
}
],
"period": {
    "startDate": "2023-04-01",
    "endDate": "2023-04-30"
}
}

```

Visualizar Pagamento (Funcionário)

Endpoint: `/mobile-payments/[id]`

Método: GET

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Descrição: Retorna os detalhes de um pagamento específico recebido pelo **usuário autenticado**.

Resposta de Sucesso (200):

```

{
  "payment": {
    "id": "payment123",
    "amount": 400,
    "date": "2023-04-30",
    "description": "Pagamento mensal - Abril 2023",
    "reference": "PAG-2023-04",
    "paymentMethod": "bank_transfer",
    "status": "completed",
    "periodStart": "2023-04-01",
    "periodEnd": "2023-04-30",
    "totalHours": 40,
    "createdBy": {
      "id": "manager456",
      "name": "Gerente da Silva",
      "email": "gerente@exemplo.com"
    },
  },
  "timeEntries": [
    {

```

```

    "id": "entry123",
    "date": "2023-04-15",
    "startTime": "2023-04-15T09:00:00",
    "endTime": "2023-04-15T17:00:00",
    "totalHours": 8,
    "observation": "Desenvolvimento de features",
    "project": "ModularCompany",
    "amount": 80
  },
  {
    "id": "entry124",
    "date": "2023-04-16",
    "startTime": "2023-04-16T09:00:00",
    "endTime": "2023-04-16T17:00:00",
    "totalHours": 8,
    "observation": "Implementação de API",
    "project": "ModularCompany",
    "amount": 80
  }
],
"createdAt": "2023-04-30T18:00:00",
"updatedAt": "2023-04-30T18:00:00"
}
}

```

Resposta de Erro:

```

{
  "error": "Pagamento não encontrado"
}

```

ou

```

{
  "error": "Você não tem permissão para visualizar este pagamento"
}

```

Confirmar Recebimento (Funcionário)

Endpoint: `/mobile-payments/[id]/confirm`

Método: PUT

Headers:

```
Authorization: Bearer ... (Token do Funcionário que recebeu o pagamento)
```

Descrição: Permite que o **Funcionário** confirme o recebimento de um pagamento que está com status 'pending' ou 'awaiting_confirmation'. Apenas o destinatário do pagamento pode confirmar.

Body: (Não requer corpo)

Resposta de Sucesso (200):

```
{
  "payment": {
    "id": "payment-uuid-confirmado",
    "amount": 1500.50,
    "date": "2023-05-31",
    "description": "Pagamento referente às horas trabalhadas em Maio",
    "reference": "Pagamento Maio/2023",
    "paymentMethod": "bank_transfer",
    "status": "completed", // Status atualizado
    "confirmedAt": "2023-06-01T10:00:00", // Data/hora da confirmação
    "periodStart": "2023-05-01",
    "periodEnd": "2023-05-31",
    "user": { "id": "uuid-do-funcionario", "name": "Nome Funcionário" },
    "creator": { "id": "uuid-do-admin", "name": "Nome Admin/Manager" },
    "timeEntries": [
      // ... (detalhes dos time entries associados)
    ]
  }
}
```

Respostas de Erro:

- **400 Bad Request:** Pagamento já está confirmado ou não pode ser confirmado (status inválido).
- **401 Unauthorized:** Token inválido ou expirado.
- **403 Forbidden:** Usuário autenticado não é o destinatário do pagamento.
- **404 Not Found:** Pagamento com o ID fornecido não encontrado.
- **500 Internal Server Error:** Erro inesperado no servidor.

Criar Pagamento (Admin/Manager)

Endpoint: `/mobile-payments`

Método: `POST`

Headers:

```
Authorization: Bearer ... (Token de Admin ou Manager)
```

Descrição: Permite que um **Administrador** ou **Gerente** crie um novo registro de pagamento para um funcionário de sua empresa, associando-o a registros de horas específicos.

Body:

```
{
  "userId": "uuid-do-funcionario",
  "amount": 1500.50, // Valor total do pagamento
  "date": "2023-05-31", // Data do pagamento
  "paymentMethod": "bank_transfer", // Ex: bank_transfer, pix, cash
  "reference": "Pagamento Maio/2023", // Referência opcional
  "description": "Pagamento referente às horas trabalhadas em Maio", //
  Descrição opcional
  "status": "pending", // Status inicial (pending, completed, etc.)
  "periodStart": "2023-05-01", // Início do período coberto
  "periodEnd": "2023-05-31", // Fim do período coberto
  "timeEntryIds": [
    "uuid-registro-hora-1",
    "uuid-registro-hora-2"
    // Array com os IDs dos TimeEntry aprovados e não pagos a serem incluídos
  ]
}
```

Resposta de Sucesso (201):

```
{
  "payment": {
    "id": "payment-uuid-criado",
    "amount": 1500.50,
    "date": "2023-05-31",
    "description": "Pagamento referente às horas trabalhadas em Maio",
    "reference": "Pagamento Maio/2023",
    "paymentMethod": "bank_transfer",
    "status": "pending",
    "periodStart": "2023-05-01",
    "periodEnd": "2023-05-31",
    "user": { "id": "uuid-do-funcionario", "name": "Nome Funcionário" },
    "creator": { "id": "uuid-do-admin", "name": "Nome Admin/Manager" },
    "timeEntries": [
      { "id": "uuid-registro-hora-1", "date": "2023-05-10", "totalHours": 8,
"amount": 750.25 },
      { "id": "uuid-registro-hora-2", "date": "2023-05-11", "totalHours": 8,
"amount": 750.25 }
    ]
  }
}
```

Respostas de Erro:

- **400 Bad Request:** Dados inválidos no corpo da requisição (ver [details](#) na resposta).
- **401 Unauthorized:** Token inválido ou expirado.
- **403 Forbidden:** Usuário não tem permissão (não é Admin/Manager) ou funcionário alvo não pertence à empresa.
- **404 Not Found:** Usuário alvo ([userId](#)) não encontrado.
- **409 Conflict:** Um ou mais [timeEntryIds](#) já estão associados a outro pagamento.
- **500 Internal Server Error:** Erro inesperado no servidor.

Verificar Saldo do Usuário

Endpoint: [/mobile-users/balance](#)

Método: [GET](#)

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters (opcionais):

- [startDate](#): Data inicial no formato ISO (YYYY-MM-DD)
- [endDate](#): Data final no formato ISO (YYYY-MM-DD)

Se não especificado, retorna o saldo do mês atual.

Resposta de Sucesso (200):

```
{
  "balance": {
    "totalApprovedHours": 80,
    "paidHours": 60,
    "unpaidHours": 20,
    "hourlyRate": 50,
    "totalAmountDue": 4000,
    "totalPaid": 3000,
    "balance": 1000,
    "currency": "BRL"
  },
  "period": {
    "startDate": "2023-04-01",
    "endDate": "2023-04-30"
  }
}
```

Notificações

Listar Notificações

Endpoint: `/mobile-notifications`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters (opcionais):

- **read:** Filtrar por status de leitura (`true` ou `false`)
- **limit:** Número máximo de notificações por página (padrão: 50)
- **page:** Número da página para paginação (padrão: 1)

Resposta de Sucesso (200):

```
{
  "notifications": [
    {
      "id": "notif123",
      "title": "Registro aprovado",
      "message": "Seu registro do dia 15/04/2023 foi aprovado",
      "type": "success",
      "read": false,
      "relatedId": "entry123",
      "relatedType": "timeEntry",
      "createdAt": "2023-04-16T10:30:00"
    }
  ],
  "pagination": {
    "total": 25,
    "page": 1,
    "limit": 50,
    "pages": 1
  },
  "unreadCount": 3
}
```

Marcar Notificação como Lida

Endpoint: `/mobile-notifications`

Método: `PUT`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```


Body (marcar uma notificação específica):

```
{
  "id": "notif123",
  "read": true
}
```

Body (marcar todas como lidas):

```
{
  "all": true
}
```

Resposta de Sucesso (200):

```
{
  "success": true,
  "message": "Notificação marcada como lida"
}
```

ou

```
{
  "success": true,
  "message": "Todas as notificações foram marcadas como lidas"
}
```

Excluir Notificação

Endpoint: `/mobile-notifications`

Método: `DELETE`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters:

- **id:** ID da notificação a ser excluída

Resposta de Sucesso (200):

```
{
  "success": true,
  "message": "Notificação excluída com sucesso"
}
```

Gerenciamento de Perfil

Atualizar Perfil

Endpoint: `/mobile-profile`

Método: `PUT`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "name": "Novo Nome",
  "email": "novo.email@exemplo.com"
}
```

Resposta de Sucesso (200):

```
{
  "user": {
    "id": "123456",
    "name": "Novo Nome",
    "email": "novo.email@exemplo.com",
    "role": "EMPLOYEE",
    "companyId": "789012",
    "hourlyRate": 50,
    "createdAt": "2023-01-15T10:30:00",
    "company": {
      "id": "789012",
      "name": "Empresa Exemplo",
      "plan": "STANDARD"
    }
  },
  "message": "Perfil atualizado com sucesso"
}
```

Alterar Senha

Endpoint: `/mobile-auth/change-password`

Método: `POST`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "currentPassword": "senhaAtual123",
  "newPassword": "novaSenha456",
  "confirmPassword": "novaSenha456"
}
```

Resposta de Sucesso (200):

```
{
  "success": true,
  "message": "Senha alterada com sucesso"
}
```

Resposta de Erro (400):

```
{
  "error": "Senha atual incorreta"
}
```

ou

```
{
  "error": "As senhas não coincidem"
}
```

Registros de Horas (Funcionalidades Avançadas)

Listar Registros com Filtros Avançados e Paginação

Endpoint: `/mobile-time-entries`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters (opcionais):

- `startDate`: Data inicial (formato YYYY-MM-DD)
- `endDate`: Data final (formato YYYY-MM-DD)
- `approved`: Filtrar por status de aprovação (true/false)
- `rejected`: Filtrar por status de rejeição (true/false)
- `project`: Filtrar por nome do projeto
- `minHours`: Filtrar por horas mínimas
- `maxHours`: Filtrar por horas máximas
- `unpaid`: Filtrar apenas registros não pagos (true/false)
- `page`: Número da página (padrão: 1)
- `limit`: Registros por página (padrão: 50)
- `sortBy`: Campo para ordenação (date, hours, createdAt)
- `sortOrder`: Direção da ordenação (asc, desc)

Resposta de Sucesso (200):

```
{
  "timeEntries": [...],
  "period": {
    "startDate": "2023-04-01",
    "endDate": "2023-04-30"
  },
  "pagination": {
    "total": 45,
    "page": 1,
    "limit": 50,
    "pages": 1
  },
  "stats": {
    "approved": 30,
    "pending": 10,
    "rejected": 5,
    "total": 45
  },
  "appliedFilters": {
    "approved": "true",
    "project": "ModularCompany",
    "sortBy": "date",
    "sortOrder": "desc"
  }
}
```

```
}  
}
```

Detalhes de Rejeição de Registro

Endpoint: `/mobile-time-entries/[id]/rejection`

Método: `GET`

Headers:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Resposta de Sucesso (200):

```
{  
  "timeEntry": {  
    "id": "entry123",  
    "date": "2023-04-15",  
    "startTime": "2023-04-15T09:00:00",  
    "endTime": "2023-04-15T17:00:00",  
    "totalHours": 8,  
    "observation": "Desenvolvimento de features",  
    "project": "ModularCompany",  
    "rejected": true,  
    "rejectionReason": "Sobreposição com outra atividade",  
    "rejectedAt": "2023-04-15T18:30:00",  
    "history": [  
      {  
        "id": "notif123",  
        "title": "Registro rejeitado",  
        "message": "Seu registro foi rejeitado: Sobreposição com outra  
atividade",  
        "createdAt": "2023-04-15T18:30:00"  
      }  
    ]  
  }  
}
```

Relatórios

Exportar Relatório

Endpoint: `/mobile-reports/export`

Método: `POST`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Body:

```
{
  "startDate": "2023-04-01",
  "endDate": "2023-04-30",
  "includeRejected": false,
  "format": "detailed"
}
```

Resposta de Sucesso (200):

```
{
  "report": {
    "title": "Relatório de Horas - 01/04/2023 a 30/04/2023",
    "user": {
      "id": "123456",
      "name": "Nome do Usuário",
      "email": "usuario@exemplo.com",
      "hourlyRate": 50
    },
    "company": {
      "id": "789012",
      "name": "Empresa Exemplo"
    },
    "period": {
      "startDate": "2023-04-01",
      "endDate": "2023-04-30"
    },
    "summary": {
      "totalEntries": 22,
      "totalHours": 176,
      "approvedEntries": 20,
      "approvedHours": 160,
      "totalValue": 8000,
      "paidHours": 120,
      "paidAmount": 6000,
      "unpaidHours": 40,
      "unpaidAmount": 2000
    },
    "entries": [
      {
        "id": "entry123",
        "date": "2023-04-01",
```

```

    "startTime": "09:00",
    "endTime": "17:00",
    "totalHours": 8,
    "value": 400,
    "observation": "Desenvolvimento de features",
    "project": "ModularCompany",
    "status": "Aprovado",
    "paid": true
  }
],
"payments": [
  {
    "id": "payment123",
    "date": "2023-04-15",
    "amount": 4000,
    "description": "Pagamento quinzenal",
    "reference": "REF123",
    "status": "completed",
    "entriesCount": 10
  }
],
"generatedAt": "2023-05-01T10:30:00"
}
}

```

Recuperação de Senha

Solicitar Recuperação

Endpoint: `/mobile-auth/forgot-password`

Método: `POST`

Body:

```

{
  "email": "usuario@exemplo.com"
}

```

Resposta de Sucesso (200):

```

{
  "success": true,
  "message": "Se o email estiver cadastrado, enviaremos instruções para recuperar sua senha"
}

```

Observações:

- Por razões de segurança, a resposta é a mesma independentemente de o email existir ou não
- O email enviado contém um link de redefinição válido por 1 hora

Redefinir Senha com Token

Endpoint: `/mobile-auth/reset-password`

Método: `POST`

Body:

```
{
  "token": "7f58a9d72e54c3b2a1d5e6f8c7b9a0e2d4f6a8b5c3e2d1f4a7e9c8b5a3f2e1d",
  "newPassword": "novaSenha456",
  "confirmPassword": "novaSenha456"
}
```

Resposta de Sucesso (200):

```
{
  "success": true,
  "message": "Senha redefinida com sucesso"
}
```

Resposta de Erro (400):

```
{
  "error": "Token inválido ou expirado"
}
```

ou

```
{
  "error": "As senhas não coincidem"
}
```

Dashboard

Endpoint: `/mobile-dashboard`

Método: `GET`

Headers:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Resposta de Sucesso (200):

```
{
  "dashboard": {
    "user": {
      "id": "123456",
      "name": "Nome do Usuário",
      "email": "usuario@exemplo.com",
      "hourlyRate": 50,
      "role": "EMPLOYEE",
      "company": {
        "id": "789012",
        "name": "Empresa Exemplo",
        "plan": "STANDARD"
      }
    },
    "notifications": {
      "unreadCount": 3
    },
    "currentMonth": {
      "totalHours": 160,
      "approvedHours": 120,
      "rejectedHours": 8,
      "pendingHours": 32,
      "pendingEntries": 4,
      "estimatedValue": 6000,
      "period": {
        "startDate": "2023-05-01",
        "endDate": "2023-05-31"
      }
    },
    "comparison": {
      "lastMonth": {
        "totalHours": 168,
        "approvedHours": 168,
        "rejectedHours": 0,
        "pendingHours": 0,
        "estimatedValue": 8400
      },
      "percentageChange": -4.76
    },
    "balance": {
      "pendingAmount": 2000,
      "currency": "BRL"
    }
  },
}
```

```

    "recentPayments": [
      {
        "id": "payment123",
        "date": "2023-04-30",
        "amount": 8400,
        "description": "Pagamento mensal - Abril 2023",
        "status": "completed"
      }
    ]
  }
}

```

Observações:

- O endpoint fornece uma visão resumida de dados importantes para o usuário
- **percentageChange** é a variação percentual de horas em relação ao mês anterior
- **pendingAmount** é o valor ainda não pago referente a registros já aprovados

Projetos

Endpoint: `/mobile-projects`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Resposta de Sucesso (200):

```

{
  "projects": {
    "official": [
      {
        "id": "proj123",
        "name": "Desenvolvimento Web",
        "description": "Projeto de desenvolvimento web com React"
      },
      {
        "id": "proj456",
        "name": "App Mobile",
        "description": "Aplicativo móvel com React Native"
      }
    ],
    "informal": [
      {
        "name": "Manutenção"
      }
    ]
  }
}

```

```
{
  {
    "name": "Reuniões"
  },
  {
    "name": "Suporte ao Cliente"
  }
]
}
```

Observações:

- **official** - Projetos formais cadastrados na plataforma
- **informal** - Projetos informais baseados em strings usadas em registros de horas

Feedback

Enviar Feedback

Endpoint: `/mobile-feedback`

Método: `POST`

Headers:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Body:

```
{
  "type": "feature",
  "title": "Implementar dark mode",
  "description": "Seria útil ter um modo escuro para usar o app à noite",
  "priority": "medium",
  "metadata": {
    "device": "iPhone 12",
    "osVersion": "iOS 16.2"
  }
}
```

Resposta de Sucesso (201):

```
{
  "success": true,
  "feedbackId": "feedback123",
}
```

```
"message": "Feedback recebido com sucesso. Obrigado pela sua contribuição!"
}
```

Tipos de Feedback:

- **bug** - Problema ou erro encontrado
- **feature** - Solicitação de nova funcionalidade
- **suggestion** - Sugestão de melhoria
- **other** - Outros tipos de feedback

Prioridades:

- **low** - Baixa prioridade
- **medium** - Prioridade média (padrão)
- **high** - Alta prioridade

Listar Feedbacks

Endpoint: `/mobile-feedback`

Método: `GET`

Headers:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Query Parameters (opcionais):

- **page**: Número da página (padrão: 1)
- **limit**: Registros por página (padrão: 20)

Resposta de Sucesso (200):

```
{
  "feedbacks": [
    {
      "id": "feedback123",
      "type": "feature",
      "title": "Implementar dark mode",
      "description": "Seria útil ter um modo escuro para usar o app à noite",
      "priority": "medium",
      "createdAt": "2023-05-15T10:30:00",
      "status": "pending"
    }
  ],
  "pagination": {
    "total": 5,
    "page": 1,
  }
}
```

```
    "limit": 20,  
    "pages": 1  
  }  
}
```

Administração (Endpoints para Admin/Manager)

Listar Usuários da Empresa

Endpoint: `/mobile-admin/users`

Método: `GET`

Headers:

Authorization: Bearer ... (Token de Admin ou Manager)

Descrição: Retorna uma lista de todos os usuários (funcionários, gerentes, etc.) pertencentes à mesma empresa do Admin/Manager autenticado. Útil para selecionar um funcionário ao criar um pagamento.

Resposta de Sucesso (200):

```
{  
  "users": [  
    {  
      "id": "uuid-user-1",  
      "name": "Funcionário Um",  
      "email": "um@empresa.com",  
      "role": "EMPLOYEE",  
      "hourlyRate": 25.50  
    },  
    {  
      "id": "uuid-user-2",  
      "name": "Gerente Silva",  
      "email": "silva@empresa.com",  
      "role": "MANAGER",  
      "hourlyRate": null  
    }  
  ]  
}
```

Respostas de Erro:

- **401 Unauthorized:** Token inválido ou expirado.
- **403 Forbidden:** Usuário não tem permissão (não é Admin/Manager).
- **400 Bad Request:** Usuário autenticado não está associado a uma empresa.
- **500 Internal Server Error:** Erro inesperado no servidor.

Implementação no React Native

Exemplos de Código

Configuração do Cliente API

```
// api.js
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';

const API_URL = 'https://modularcompany.vercel.app/api';

const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Interceptor para adicionar token a todas as requisições
api.interceptors.request.use(
  async (config) => {
    const token = await AsyncStorage.getItem('@ModularCompany:token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

export default api;
```

Login

```
// authService.js
import api from './api';
import AsyncStorage from '@react-native-async-storage/async-storage';

const login = async (email: string, password: string) => {
  try {
    const response = await api.post('/mobile-auth', { email, password });

    // Armazenar o token e dados do usuário
    await AsyncStorage.setItem('token', response.data.token);
    await AsyncStorage.setItem('userData', JSON.stringify(response.data.user));

    return response.data;
  }
};
```

```

    } catch (error) {
      console.error('Erro de login:', error);
      throw error;
    }
  };
};

```

Manipulação de Registros de Horas

```

// timeEntryService.js
import api from './api';
import { format } from 'date-fns';

// Buscar registros de horas
const fetchTimeEntries = async (startDate?: Date, endDate?: Date) => {
  try {
    let url = '/mobile-time-entries';
    const params = new URLSearchParams();

    if (startDate) {
      params.append('startDate', format(startDate, 'yyyy-MM-dd'));
    }

    if (endDate) {
      params.append('endDate', format(endDate, 'yyyy-MM-dd'));
    }

    if (params.toString()) {
      url += `?${params.toString()}`;
    }

    const response = await api.get(url);
    return response.data;
  } catch (error) {
    console.error('Erro ao buscar registros de horas:', error);
    throw error;
  }
};

// Visualizar um registro específico
const viewTimeEntry = async (id: string) => {
  try {
    const response = await api.get(`/mobile-time-entries/${id}/view`);
    return response.data;
  } catch (error) {
    console.error('Erro ao buscar registro específico:', error);
    throw error;
  }
};

// Criar registro de horas

```

```

const createTimeEntry = async (timeEntryData: {
  date: string;
  startTime: string;
  endTime: string;
  observation?: string;
  project?: string;
}) => {
  try {
    const response = await api.post('/mobile-time-entries', timeEntryData);
    return response.data;
  } catch (error) {
    console.error('Erro ao criar registro de horas:', error);
    throw error;
  }
};

// Editar registro de horas
const updateTimeEntry = async (
  id: string,
  timeEntryData: {
    date: string;
    startTime: string;
    endTime: string;
    observation?: string;
    project?: string;
  }
) => {
  try {
    const response = await api.put(`/mobile-time-entries/${id}`,
timeEntryData);
    return response.data;
  } catch (error) {
    console.error('Erro ao atualizar registro de horas:', error);
    throw error;
  }
};

// Excluir registro de horas
const deleteTimeEntry = async (id: string) => {
  try {
    const response = await api.delete(`/mobile-time-entries/${id}`);
    return response.data;
  } catch (error) {
    console.error('Erro ao excluir registro de horas:', error);
    throw error;
  }
};

```

Manipulação de Pagamentos


```

// paymentService.js
import api from './api';
import { format } from 'date-fns';

// Buscar pagamentos
const fetchPayments = async (startDate?: Date, endDate?: Date, status?: string)
=> {
  try {
    let url = '/mobile-payments';
    const params = new URLSearchParams();

    if (startDate) {
      params.append('startDate', format(startDate, 'yyyy-MM-dd'));
    }

    if (endDate) {
      params.append('endDate', format(endDate, 'yyyy-MM-dd'));
    }

    if (status) {
      params.append('status', status);
    }

    if (params.toString()) {
      url += `?${params.toString()}`;
    }

    const response = await api.get(url);
    return response.data;
  } catch (error) {
    console.error('Erro ao buscar pagamentos:', error);
    throw error;
  }
};

// Visualizar um pagamento específico
const viewPayment = async (id: string) => {
  try {
    const response = await api.get(`/mobile-payments/${id}`);
    return response.data;
  } catch (error) {
    console.error('Erro ao buscar pagamento específico:', error);
    throw error;
  }
};

// Verificar saldo do usuário
const getUserBalance = async (startDate?: Date, endDate?: Date) => {
  try {
    let url = '/mobile-users/balance';
    const params = new URLSearchParams();
  }
};

```

```

    if (startDate) {
      params.append('startDate', format(startDate, 'yyyy-MM-dd'));
    }

    if (endDate) {
      params.append('endDate', format(endDate, 'yyyy-MM-dd'));
    }

    if (params.toString()) {
      url += `?${params.toString()}`;
    }

    const response = await api.get(url);
    return response.data;
  } catch (error) {
    console.error('Erro ao verificar saldo:', error);
    throw error;
  }
};

```

Exemplo de Uso em Componentes

```

// PaymentsScreen.jsx
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, ActivityIndicator, StyleSheet } from 'react-native';
import { fetchPayments } from '../services/paymentService';

const PaymentsScreen = () => {
  const [payments, setPayments] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    loadPayments();
  }, []);

  const loadPayments = async () => {
    try {
      setLoading(true);
      setError(null);
      const result = await fetchPayments();
      setPayments(result.payments);
    } catch (err) {
      setError('Não foi possível carregar os pagamentos');
      console.error(err);
    } finally {
      setLoading(false);
    }
  };
};

```

```

if (loading) {
  return (
    <View style={styles.centered}>
      <ActivityIndicator size="large" color="#0066CC" />
    </View>
  );
}

if (error) {
  return (
    <View style={styles.centered}>
      <Text style={styles.error}>{error}</Text>
    </View>
  );
}

return (
  <View style={styles.container}>
    <Text style={styles.title}>Meus Pagamentos</Text>
    <FlatList
      data={payments}
      keyExtractor={({item}) => item.id}
      renderItem={({ item }) => (
        <View style={styles.paymentCard}>
          <Text style={styles.paymentDate}>{item.date}</Text>
          <Text style={styles.paymentAmount}>R$ {item.amount.toFixed(2)}
        </Text>

          <Text style={styles.paymentDesc}>{item.description}</Text>
          <Text style={styles.paymentStatus}>
            Status: {item.status === 'completed' ? 'Concluído' : 'Pendente'}
          </Text>
          <Text>Total de horas: {item.totalHours}h</Text>
        </View>
      )}
    />
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    backgroundColor: '#f5f5f5',
  },
  centered: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  title: {
    fontSize: 22,

```

```

    fontWeight: 'bold',
    marginBottom: 16,
  },
  error: {
    color: 'red',
    fontSize: 16,
  },
  paymentCard: {
    backgroundColor: 'white',
    borderRadius: 8,
    padding: 16,
    marginBottom: 12,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 4,
    elevation: 2,
  },
  paymentDate: {
    fontSize: 14,
    color: '#666',
  },
  paymentAmount: {
    fontSize: 20,
    fontWeight: 'bold',
    marginVertical: 8,
  },
  paymentDesc: {
    fontSize: 16,
    marginBottom: 8,
  },
  paymentStatus: {
    fontSize: 14,
    color: '#333',
    marginBottom: 8,
  },
});

```

```
export default PaymentsScreen;
```

```

// BalanceScreen.jsx
import React, { useState, useEffect } from 'react';
import { View, Text, ActivityIndicator, StyleSheet, TouchableOpacity } from
'react-native';
import { getUserBalance } from '../services/paymentService';
import { format } from 'date-fns';
import { ptBR } from 'date-fns/locale';

const BalanceScreen = () => {
  const [balanceData, setBalanceData] = useState(null);

```

```

const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
const [period, setPeriod] = useState({ startDate: null, endDate: null });

useEffect(() => {
  loadBalance();
}, []);

const loadBalance = async () => {
  try {
    setLoading(true);
    setError(null);
    const result = await getUserBalance();
    setBalanceData(result.balance);
    setPeriod(result.period);
  } catch (err) {
    setError('Não foi possível carregar o saldo');
    console.error(err);
  } finally {
    setLoading(false);
  }
};

if (loading) {
  return (
    <View style={styles.centered}>
      <ActivityIndicator size="large" color="#0066CC" />
    </View>
  );
}

if (error) {
  return (
    <View style={styles.centered}>
      <Text style={styles.error}>{error}</Text>
      <TouchableOpacity style={styles.button} onPress={loadBalance}>
        <Text style={styles.buttonText}>Tentar novamente</Text>
      </TouchableOpacity>
    </View>
  );
}

return (
  <View style={styles.container}>
    <Text style={styles.title}>Meu Saldo</Text>

    <Text style={styles.periodText}>
      Período: {period?.startDate && format(new Date(period.startDate),
'dd/MM/yyyy', { locale: ptBR })} -
      {period?.endDate && format(new Date(period.endDate), 'dd/MM/yyyy', {
locale: ptBR })}
    </Text>
  </View>
);

```

```

<View style={styles.card}>
  <Text style={styles.balanceTitle}>Saldo a receber</Text>
  <Text style={styles.balanceValue}>
    R$ {balanceData?.balance.toFixed(2)}
  </Text>
</View>

<View style={styles.statsContainer}>
  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Total de horas aprovadas</Text>
    <Text style={styles.statValue}>
{balanceData?.totalApprovedHours}h</Text>
  </View>

  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Horas pagas</Text>
    <Text style={styles.statValue}>{balanceData?.paidHours}h</Text>
  </View>

  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Horas não pagas</Text>
    <Text style={styles.statValue}>{balanceData?.unpaidHours}h</Text>
  </View>

  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Valor/hora</Text>
    <Text style={styles.statValue}>R$
{balanceData?.hourlyRate.toFixed(2)}</Text>
  </View>

  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Total devido</Text>
    <Text style={styles.statValue}>R$
{balanceData?.totalAmountDue.toFixed(2)}</Text>
  </View>

  <View style={styles.statCard}>
    <Text style={styles.statLabel}>Total pago</Text>
    <Text style={styles.statValue}>R$ {balanceData?.totalPaid.toFixed(2)}
</Text>
  </View>
</View>
);
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    backgroundColor: '#f5f5f5',
  },
  centered: {

```

```

    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  title: {
    fontSize: 22,
    fontWeight: 'bold',
    marginBottom: 8,
  },
  periodText: {
    fontSize: 14,
    color: '#666',
    marginBottom: 16,
  },
  error: {
    color: 'red',
    fontSize: 16,
    marginBottom: 16,
  },
  button: {
    backgroundColor: '#0066CC',
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 4,
  },
  buttonText: {
    color: 'white',
    fontSize: 16,
    fontWeight: '500',
  },
  card: {
    backgroundColor: 'white',
    borderRadius: 8,
    padding: 20,
    marginBottom: 16,
    alignItems: 'center',
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 4,
    elevation: 2,
  },
  balanceTitle: {
    fontSize: 16,
    color: '#666',
    marginBottom: 8,
  },
  balanceValue: {
    fontSize: 32,
    fontWeight: 'bold',
    color: '#0066CC',
  },
  statsContainer: {

```

```

    flexDirection: 'row',
    flexWrap: 'wrap',
    justifyContent: 'space-between',
  },
  statCard: {
    backgroundColor: 'white',
    borderRadius: 8,
    padding: 16,
    marginBottom: 12,
    width: '48%',
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 1 },
    shadowOpacity: 0.1,
    shadowRadius: 2,
    elevation: 1,
  },
  statLabel: {
    fontSize: 14,
    color: '#666',
    marginBottom: 8,
  },
  statValue: {
    fontSize: 18,
    fontWeight: 'bold',
  },
});

export default BalanceScreen;

```

Observações Importantes

1. **Tratamento de Erros:** Sempre verifique os códigos de status e trate os erros adequadamente nas chamadas a todos os endpoints. Implemente uma experiência de usuário que informe claramente quando ocorrer um problema.
2. **Expiração do Token:** O token expira após 24 horas. Implemente lógica para detectar a expiração e redirecionar para a tela de login, preservando o contexto do usuário quando possível.
3. **Formatação de Datas:** As datas são enviadas e recebidas em formato ISO. Use bibliotecas como `date-fns` para manipulação e formatação consistente de datas.
4. **CORS:** Todos os endpoints possuem CORS configurado para permitir requisições de qualquer origem, facilitando o desenvolvimento de aplicações móveis.
5. **Validação:** A API implementa validação básica no servidor, mas é recomendável validar os dados também no cliente para melhorar a experiência do usuário.
6. **Cache:** Considere implementar estratégias de cache para dados que não mudam frequentemente, como o perfil do usuário, para reduzir requisições desnecessárias.

7. **Gerenciamento de Estado:** Em aplicações React Native maiores, considere usar gerenciadores de estado como Redux ou Context API para compartilhar dados entre componentes.
8. **Teste de Conectividade:** Implemente verificações de conectividade para garantir uma boa experiência em ambientes com conexão instável, incluindo funcionalidades offline quando possível.
9. **Paginação:** Para listas grandes de pagamentos ou registros de horas, a API suporta paginação para melhorar o desempenho. Implemente a lógica de "carregar mais" em suas listagens.
10. **Segurança:** Nunca armazene o token em locais não seguros. Utilize o AsyncStorage apenas para dados não sensíveis e considere bibliotecas como `react-native-keychain` para informações sensíveis.

Suporte

Para questões e problemas relacionados à API, entre em contato com a equipe de desenvolvimento do ModularCompany através do e-mail fabricappsdrumblow@gmail.com ou abra uma issue no repositório do projeto.