Contratos de Integração

Visão Geral

As integrações do Sistema Matriz seguem o padrão **Adapter**. Cada integração define um contrato (interface TypeScript) que pode ter múltiplas implementações (mock, real, etc).

CalendarAdapter

Interface

```typescript
interface CalendarAdapter {
  listEvents(
    tenantId: string,
    userId: string,
    startDate: Date,
    endDate: Date
  ): Promise<CalendarEvent[]>;

  createEvent(
    tenantId: string,
    userId: string,
    event: CreateEventDto
  ): Promise<CalendarEvent>;

  updateEvent(
    tenantId: string,
    eventId: string,
    updates: UpdateEventDto
  ): Promise<CalendarEvent>;

  deleteEvent(
    tenantId: string,
    eventId: string
  ): Promise<void>;

  findFreeSlots(
    tenantId: string,
    userId: string,
    date: Date,
    duration: number
  ): Promise<TimeSlot[]>;
}
```

DTOs

```typescript
interface CalendarEvent {
  id: string;
  tenantId: string;
  userId: string;
  title: string;
  description?: string;
  startTime: Date;
  endTime: Date;
  isPrivate: boolean;
  status: 'confirmed' | 'tentative' | 'cancelled';
  category?: string;
  location?: string;
  reminder?: number; // minutos antes
}

interface CreateEventDto {
  title: string;
  description?: string;
  startTime: Date;
  endTime: Date;
  isPrivate?: boolean;
  category?: string;
  location?: string;
  reminder?: number;
}

interface UpdateEventDto {
  title?: string;
  description?: string;
  startTime?: Date;
  endTime?: Date;
  isPrivate?: boolean;
  status?: 'confirmed' | 'tentative' | 'cancelled';
  category?: string;
  location?: string;
  reminder?: number;
}

interface TimeSlot {
  start: Date;
  end: Date;
  duration: number;
}
```

Exemplo de Uso

```
import { calendarAdapter } from '@/lib/integrations/calendar-adapter';

// Listar eventos
const events = await calendarAdapter.listEvents(
  tenantId,
  userId,
  new Date('2026-01-29'),
  new Date('2026-01-30')
);

// Criar evento
const newEvent = await calendarAdapter.createEvent(tenantId, userId, {
  title: 'Reunião',
  startTime: new Date('2026-01-30T14:00:00'),
  endTime: new Date('2026-01-30T15:00:00'),
  category: 'work'
});

// Encontrar horários livres
const freeSlots = await calendarAdapter.findFreeSlots(
  tenantId,
  userId,
  new Date('2026-01-30'),
  60 // 60 minutos
);
```

FinanceAdapter

Interface

```
interface FinanceAdapter {
  createTransaction(
    tenantId: string,
    userId: string,
    transaction: CreateTransactionDto
  ): Promise<FinancialTransaction>;

  listTransactions(
    tenantId: string,
    userId: string,
    filters: TransactionFilters
  ): Promise<FinancialTransaction[]>;

  categorizeTransaction(
    description: string
  ): Promise<CategorySuggestion>;
}
```

DTOs

```typescript
interface FinancialTransaction {
  id: string;
  tenantId: string;
  userId: string;
  type: 'pix' | 'card' | 'boleto' | 'transfer' | 'cash';
  amount: number;
  category: string;
  description?: string;
  date: Date;
  status: 'pending' | 'confirmed' | 'cancelled';
  merchant?: string;
  installments?: number;
  isRecurring: boolean;
}

interface CreateTransactionDto {
  type: 'pix' | 'card' | 'boleto' | 'transfer' | 'cash';
  amount: number;
  category: string;
  description?: string;
  date: Date;
  merchant?: string;
  installments?: number;
  isRecurring?: boolean;
}

interface TransactionFilters {
  startDate?: Date;
  endDate?: Date;
  type?: string;
  category?: string;
  minAmount?: number;
  maxAmount?: number;
}

interface CategorySuggestion {
  category: string;
  confidence: number; // 0-1
}
```

Categorias Padrão

- Alimentação
- Transporte
- Saúde
- Educação
- Lazer
- Moradia
- Impostos
- SaaS/Assinaturas
- Vestuário
- Comunicação
- Outros

Exemplo de Uso

```
import { financeAdapter } from '@/lib/integrations/finance-adapter';

// Criar transação
const transaction = await financeAdapter.createTransaction(tenantId, userId, {
  type: 'pix',
  amount: 150.00,
  category: 'Transporte',
  description: 'Uber',
  date: new Date()
});

// Listar transações
const transactions = await financeAdapter.listTransactions(tenantId, userId, {
  startDate: new Date('2026-01-01'),
  endDate: new Date('2026-01-31'),
  category: 'Alimentação'
});

// Sugerir categoria
const suggestion = await financeAdapter.categorizeTransaction('uber viagem trabalho');
// { category: 'Transporte', confidence: 0.85 }
```

Como Substituir Mock por Integração Real

Passo 1: Criar Nova Implementação

```
// lib/integrations/adapters/google-calendar-adapter.ts
import type { CalendarAdapter, CalendarEvent, ... } from '@/lib/types';

export class GoogleCalendarAdapter implements CalendarAdapter {
  private oauth2Client: OAuth2Client;

  constructor(credentials: GoogleCredentials) {
    this.oauth2Client = new OAuth2Client(credentials);
  }

  async listEvents(...) {
    const calendar = google.calendar({ version: 'v3', auth: this.oauth2Client });
    const response = await calendar.events.list({ ... });
    return this.normalizeEvents(response.data.items);
  }

  // ... implementar outros métodos
}
```

Passo 2: Configurar Factory

```
// lib/integrations/factory.ts
import { MockCalendarAdapter } from './mocks/calendar-adapter';
import { GoogleCalendarAdapter } from './adapters/google-calendar-adapter';

export function createCalendarAdapter(tenantConfig: TenantIntegrationConfig) {
  if (tenantConfig.integrationSlug === 'integration.google.calendar') {
    if (tenantConfig.credentialsRef) {
      return new GoogleCalendarAdapter(getCredentials(tenantConfig.credentialsRef));
    }
  }
  // Fallback para mock
  return new MockCalendarAdapter();
}
```

Passo 3: Atualizar Agente

```
// lib/agents/secretary-agent.ts
import { createCalendarAdapter } from '@/lib/integrations/factory';

export async function processSecretaryMessage(message, context) {
  const calendarAdapter = await createCalendarAdapter(
    context.tenantIntegrationConfig
  );

  // Usar adapter normalmente
  const events = await calendarAdapter.listEvents(...);
}
```

Padrões de Normalização

Cada adapter deve normalizar dados externos para o formato interno:

```
class GoogleCalendarAdapter {
  private normalizeEvent(googleEvent: GoogleEvent): CalendarEvent {
    return {
      id: googleEvent.id,
      title: googleEvent.summary || 'Sem título',
      startTime: new Date(googleEvent.start.dateTime || googleEvent.start.date),
      endTime: new Date(googleEvent.end.dateTime || googleEvent.end.date),
      isPrivate: googleEvent.visibility === 'private',
      // ... outros campos
    };
  }
}
```