事業会社専用ページの4タブ形式実装計画

> 📋 **ステータス**: 計画中
> 📅 **作成日**: 2025-12-11
> 👤 **用途**: 事業会社の専用ページを組織と同じ4タブ形式にする実装手順の検討

概要

事業会社の専用ページを組織の専用ページと同じように4つのタブ形式に変更します。

タブ構成

1. **事業会社紹介** - 事業会社の基本情報と紹介文
2. **注力事業** - 事業会社が注力している事業領域
3. **注力施策** - 事業会社の注力施策（カード形式で追加可能）
4. **議事録** - 事業会社関連の議事録（カード形式で追加可能）

現在の実装状況

組織の詳細ページ

- **パス**: `/app/organization/detail/page.tsx`
- **タブ構成**:
  - `introduction`: 組織紹介
  - `focusAreas`: 注力領域
  - `focusInitiatives`: 注力施策（カード追加可能）
  - `meetingNotes`: 議事録（カード追加可能）

事業会社の詳細ページ

- **パス**: `/app/companies/detail/page.tsx`
- **現在の機能**: 組織表示関係の管理のみ
- **タブ構造**: なし

データベーステーブル構造

組織関連テーブル

```
-- 組織コンテンツ
CREATE TABLE organizationContents (
    id TEXT PRIMARY KEY,
    organizationId TEXT NOT NULL,
    introduction TEXT,
    focusAreas TEXT,
    meetingNotes TEXT,
    createdAt TEXT,
    updatedAt TEXT,
```

```
    FOREIGN KEY (organizationId) REFERENCES organizations(id)
);

-- 注力施策
CREATE TABLE focusInitiatives (
    id TEXT PRIMARY KEY,
    organizationId TEXT NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    content TEXT,
    themeIds TEXT,
    topicIds TEXT,
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (organizationId) REFERENCES organizations(id)
);

-- 議事録
CREATE TABLE meetingNotes (
    id TEXT PRIMARY KEY,
    organizationId TEXT NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    content TEXT,
    chromaSynced INTEGER DEFAULT 0,
    chromaSyncError TEXT,
    lastChromaSyncAttempt TEXT,
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (organizationId) REFERENCES organizations(id)
);
```

事業会社テーブル

```
CREATE TABLE companies (
    id TEXT PRIMARY KEY,
    code TEXT NOT NULL UNIQUE,
    name TEXT NOT NULL,
    nameShort TEXT,
    category TEXT NOT NULL,
    organizationId TEXT NOT NULL,
    company TEXT,
    division TEXT,
    department TEXT,
    region TEXT NOT NULL,
    position INTEGER DEFAULT 0,
    createdAt TEXT NOT NULL,
    updatedAt TEXT NOT NULL,
    FOREIGN KEY (organizationId) REFERENCES organizations(id)
);
```

データベース設計の比較検討

オプション1: 既存テーブルを拡張（推奨）

アプローチ

既存のテーブルに`companyId`カラムを追加し、組織と事業会社の両方で使用可能にする。

メリット

- ✅ **コードの再利用性が高い**: 組織用の既存ロジックを流用可能
- ✅ **統一されたデータ構造**: 組織と事業会社で同じデータ構造を維持
- ✅ **クエリの簡素化**: 組織と事業会社を統合して検索・集計が容易
- ✅ **RAG検索の統合が容易**: 検索対象範囲の調整が簡単
- ✅ **保守性が高い**: 1つのテーブルを管理すればよい

デメリット

- ⚠ **外部キー制約の複雑化**: `organizationId`と`companyId`の両方を扱う必要がある
- ⚠ **NULL値の扱い**: どちらか一方がNULLになるため、NOT NULL制約が使えない
- ⚠ **既存データへの影響**: 既存の組織データに`companyId`がNULLとして追加される

実装例

```
-- focusInitiativesテーブルを拡張
ALTER TABLE focusInitiatives ADD COLUMN companyId TEXT;
CREATE INDEX idx_focusInitiatives_companyId ON
focusInitiatives(companyId);
-- 外部キー制約は追加しない（NULLを許可するため）

-- meetingNotesテーブルを拡張
ALTER TABLE meetingNotes ADD COLUMN companyId TEXT;
CREATE INDEX idx_meetingNotes_companyId ON meetingNotes(companyId);

-- organizationContentsテーブルを拡張（事業会社コンテンツ用）
CREATE TABLE companyContents (
    id TEXT PRIMARY KEY,
    companyId TEXT NOT NULL,
    introduction TEXT,
    focusBusinesses TEXT,  -- 注力事業（組織のfocusAreasに相当）
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (companyId) REFERENCES companies(id)
);
```

オプション2: 新しいテーブルを作成

アプローチ

事業会社専用の新しいテーブルを作成する。

メリット

- ✅ **データの分離**: 組織と事業会社のデータが明確に分離される
- ✅ **外部キー制約が明確**: `companyId`をNOT NULLにできる
- ✅ **既存データへの影響なし**: 組織の既存テーブルに変更を加えない

デメリット

- ❌ **コードの重複**: 組織用と事業会社用で似たようなコードを2つ書く必要がある
- ❌ **保守性の低下**: 2つのテーブルを管理する必要がある
- ❌ **RAG検索の統合が複雑**: 2つのテーブルから検索する必要がある
- ❌ **クエリの複雑化**: 組織と事業会社を統合して検索する場合、UNIONが必要

実装例

```
-- 事業会社コンテンツテーブル
CREATE TABLE companyContents (
    id TEXT PRIMARY KEY,
    companyId TEXT NOT NULL,
    introduction TEXT,
    focusBusinesses TEXT,
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (companyId) REFERENCES companies(id)
);

-- 事業会社注力施策テーブル
CREATE TABLE companyFocusInitiatives (
    id TEXT PRIMARY KEY,
    companyId TEXT NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    content TEXT,
    themeIds TEXT,
    topicIds TEXT,
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (companyId) REFERENCES companies(id)
);

-- 事業会社議事録テーブル
CREATE TABLE companyMeetingNotes (
    id TEXT PRIMARY KEY,
    companyId TEXT NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
```

```
    content TEXT,
    chromaSynced INTEGER DEFAULT 0,
    chromaSyncError TEXT,
    lastChromaSyncAttempt TEXT,
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (companyId) REFERENCES companies(id)
);
```

推奨アプローチ: オプション1（既存テーブル拡張）

理由

1. **RAG検索の統合が容易**: 将来的に「組織だけ」「事業会社だけ」「組織＋事業会社」などの検索対象範囲を調整する際、1つのテーブルから検索する方が簡単
2. **コードの再利用性**: 組織用の既存ロジックを流用できるため、開発効率が高い
3. **統一されたデータ構造**: 組織と事業会社で同じデータ構造を維持できる

実装方針

1. データベーススキーマの拡張

```
-- focusInitiativesテーブルにcompanyIdを追加
ALTER TABLE focusInitiatives ADD COLUMN companyId TEXT;
CREATE INDEX idx_focusInitiatives_companyId ON
focusInitiatives(companyId);

-- meetingNotesテーブルにcompanyIdを追加
ALTER TABLE meetingNotes ADD COLUMN companyId TEXT;
CREATE INDEX idx_meetingNotes_companyId ON meetingNotes(companyId);

-- 事業会社コンテンツテーブルを作成
CREATE TABLE companyContents (
    id TEXT PRIMARY KEY,
    companyId TEXT NOT NULL,
    introduction TEXT,
    focusBusinesses TEXT,  -- 注力事業（組織のfocusAreasに相当）
    createdAt TEXT,
    updatedAt TEXT,
    FOREIGN KEY (companyId) REFERENCES companies(id)
);

CREATE INDEX idx_companyContents_companyId ON
companyContents(companyId);
```

2. データ型の定義

```
// lib/companiesApi.ts または lib/orgApi.ts に追加

export interface CompanyContent {
  id: string;
  companyId: string;
  introduction?: string;
  focusBusinesses?: string;  // 注力事業
  createdAt?: string;
  updatedAt?: string;
}

export interface CompanyFocusInitiative {
  id: string;
  companyId: string;  // organizationIdの代わり
  title: string;
  description?: string;
  content?: string;
  themeIds?: string;
  topicIds?: string;
  createdAt?: string;
  updatedAt?: string;
}

export interface CompanyMeetingNote {
  id: string;
  companyId: string;  // organizationIdの代わり
  title: string;
  description?: string;
  content?: string;
  chromaSynced?: number;
  chromaSyncError?: string;
  lastChromaSyncAttempt?: string;
  createdAt?: string;
  updatedAt?: string;
}
```

3. API関数の追加

組織用のAPI関数を参考に、事業会社用のAPI関数を作成：

```
// lib/companiesApi.ts に追加

// 事業会社コンテンツの取得・保存
export async function getCompanyContent(companyId: string):
Promise<CompanyContent | null>
export async function saveCompanyContent(content: CompanyContent):
Promise<string>

// 事業会社の注力施策の取得・保存・削除
```

```
export async function getCompanyFocusInitiatives(companyId: string):
Promise<CompanyFocusInitiative[]>
export async function saveCompanyFocusInitiative(initiative:
CompanyFocusInitiative): Promise<string>
export async function deleteCompanyFocusInitiative(initiativeId:
string): Promise<void>
export async function generateUniqueCompanyInitiativeId():
Promise<string>

// 事業会社の議事録の取得・保存・削除
export async function getCompanyMeetingNotes(companyId: string):
Promise<CompanyMeetingNote[]>
export async function saveCompanyMeetingNote(note: CompanyMeetingNote):
Promise<string>
export async function deleteCompanyMeetingNote(noteId: string):
Promise<void>
export async function generateUniqueCompanyMeetingNoteId():
Promise<string>
```

4. フロントエンドの実装

4.1 タブ構造の追加

`/app/companies/detail/page.tsx`を組織の詳細ページと同じ構造に変更：

```
type TabType = 'introduction' | 'focusBusinesses' | 'focusInitiatives' |
'meetingNotes';

const [activeTab, setActiveTab] = useState<TabType>('introduction');
const [companyContent, setCompanyContent] = useState<CompanyContent |
null>(null);
const [focusInitiatives, setFocusInitiatives] =
useState<CompanyFocusInitiative[]>([]);
const [meetingNotes, setMeetingNotes] = useState<CompanyMeetingNote[]>
([]);
```

4.2 タブUIの実装

組織の詳細ページと同じタブUIを実装：

```
<div style={{ display: 'flex', borderBottom: '1px solid #E5E7EB',
marginBottom: '24px' }}>
  <button onClick={() => handleTabChange('introduction')}>
    事業会社紹介
  </button>
  <button onClick={() => handleTabChange('focusBusinesses')}>
    注力事業
```

```
    </button>
    <button onClick={() => handleTabChange('focusInitiatives')}>
      注力施策 ({focusInitiatives.length})
    </button>
    <button onClick={() => handleTabChange('meetingNotes')}>
      議事録 ({meetingNotes.length})
    </button>
  </div>
```

4.3 各タブのコンテンツ実装

- **事業会社紹介タブ**: 事業会社の基本情報と紹介文の編集
- **注力事業タブ**: 注力事業のテキスト編集（組織のfocusAreasと同じ）
- **注力施策タブ**: カード形式で注力施策を追加・編集・削除（組織と同じ機能）
- **議事録タブ**: カード形式で議事録を追加・編集・削除（組織と同じ機能）

5. RAG検索の統合

将来的にRAG検索で検索対象範囲を調整できるようにする：

```
// lib/knowledgeGraphRAG.ts の拡張

export async function searchKnowledgeGraph(
  queryText: string,
  limit: number = 10,
  filters?: {
    organizationId?: string;
    companyId?: string;  // 新規追加
    searchScope?: 'organization' | 'company' | 'both';  // 新規追加
    // ... 既存のフィルター
  },
  // ...
)
```

検索対象範囲の調整UI:

```
<select value={searchScope} onChange={(e) =>
setSearchScope(e.target.value)}>
  <option value="organization">組織のみ</option>
  <option value="company">事業会社のみ</option>
  <option value="both">組織＋事業会社</option>
</select>
```

実装手順

フェーズ1: データベーススキーマの拡張

1. `src-tauri/src/database/mod.rs`にスキーマ変更を追加
    - `focusInitiatives`テーブルに`companyId`カラムを追加
    - `meetingNotes`テーブルに`companyId`カラムを追加
    - `companyContents`テーブルを作成
    - インデックスを追加
2. マイグレーション処理の実装
    - 既存データへの影響を確認
    - 必要に応じてデータ移行スクリプトを作成

フェーズ2: バックエンドAPIの実装

1. `lib/companiesApi.ts`にAPI関数を追加
    - 事業会社コンテンツの取得・保存
    - 事業会社の注力施策の取得・保存・削除
    - 事業会社の議事録の取得・保存・削除
2. Rust側のAPIエンドポイントを追加（必要に応じて）
    - `src-tauri/src/commands/companies.rs`にコマンドを追加

フェーズ3: フロントエンドの実装

1. タブ構造の実装
    - `app/companies/detail/page.tsx`を4タブ形式に変更
    - タブの状態管理を追加
2. 各タブのコンテンツ実装
    - 事業会社紹介タブ
    - 注力事業タブ
    - 注力施策タブ（組織と同じ機能）
    - 議事録タブ（組織と同じ機能）
3. モーダルとフォームの実装
    - 注力施策追加・編集モーダル
    - 議事録追加・編集モーダル

フェーズ4: RAG検索の統合（将来実装）

1. `lib/knowledgeGraphRAG.ts`の拡張
    - `companyId`フィルターの追加
    - `searchScope`パラメータの追加
2. RAG検索ページのUI拡張

    - 検索対象範囲の選択UIを追加

3. ChromaDBのコレクション構造の検討
    - 事業会社用のコレクションを作成するか、既存のコレクションを拡張するか

注意事項

データ整合性

- `focusInitiatives`と`meetingNotes`テーブルで、`organizationId`と`companyId`の両方がNULLになることを防ぐ必要がある
- チェック制約またはアプリケーション側でのバリデーションが必要

パフォーマンス

- `companyId`カラムにインデックスを追加して検索性能を確保
- 組織と事業会社を統合して検索する場合のクエリ最適化

既存機能への影響

- 組織の既存機能に影響を与えないよう注意
- 既存の組織データは`companyId`がNULLのまま動作する必要がある

参考資料

- `/app/organization/detail/page.tsx`: 組織の詳細ページの実装
- `/lib/orgApi.ts`: 組織関連のAPI関数
- `/lib/companiesApi.ts`: 事業会社関連のAPI関数
- `/src-tauri/src/database/mod.rs`: データベーススキーマ定義