Contents

SECTION 23: CANVAS & ARTIFACTS (v3.6.0)

23.1 Canvas Overview

Rich content editing with artifacts for code, documents, and visualizations.

23.2 Canvas Database Schema

```sql
-- migrations/032_canvas_artifacts.sql

CREATE TABLE canvases (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    chat_id UUID REFERENCES chats(id),
    canvas_name VARCHAR(200),
    canvas_type VARCHAR(50) NOT NULL DEFAULT 'general',
    content JSONB NOT NULL DEFAULT '{}',
    version INTEGER DEFAULT 1,
    is_published BOOLEAN DEFAULT false,
    published_url VARCHAR(500),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE artifacts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    canvas_id UUID NOT NULL REFERENCES canvases(id) ON DELETE CASCADE,
    artifact_type VARCHAR(50) NOT NULL,
    title VARCHAR(200),
    content TEXT NOT NULL,
    language VARCHAR(50),
    position_x INTEGER DEFAULT 0,
    position_y INTEGER DEFAULT 0,
    width INTEGER DEFAULT 400,
    height INTEGER DEFAULT 300,
```

```
    z_index INTEGER DEFAULT 0,
    is_collapsed BOOLEAN DEFAULT false,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE artifact_versions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    artifact_id UUID NOT NULL REFERENCES artifacts(id) ON DELETE CASCADE,
    version INTEGER NOT NULL,
    content TEXT NOT NULL,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_canvases_user ON canvases(tenant_id, user_id);
CREATE INDEX idx_artifacts_canvas ON artifacts(canvas_id);
CREATE INDEX idx_artifact_versions ON artifact_versions(artifact_id, version DESC);

ALTER TABLE canvases ENABLE ROW LEVEL SECURITY;
ALTER TABLE artifacts ENABLE ROW LEVEL SECURITY;
ALTER TABLE artifact_versions ENABLE ROW LEVEL SECURITY;

CREATE POLICY canvases_isolation ON canvases USING (tenant_id = current_setting('app.current_te
CREATE POLICY artifacts_isolation ON artifacts USING (
    canvas_id IN (SELECT id FROM canvases WHERE tenant_id = current_setting('app.current_tenant
);
CREATE POLICY artifact_versions_isolation ON artifact_versions USING (
    artifact_id IN (SELECT a.id FROM artifacts a JOIN canvases c ON a.canvas_id = c.id WHERE c
);
```

23.3 Canvas Service

```
// packages/core/src/services/canvas-service.ts

import { Pool } from 'pg';

type ArtifactType = 'code' | 'markdown' | 'mermaid' | 'html' | 'svg' | 'json' | 'table';

interface ArtifactCreate {
    type: ArtifactType;
    title?: string;
    content: string;
    language?: string;
    position?: { x: number; y: number };
    size?: { width: number; height: number };
}
```

```typescript
export class CanvasService {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async createCanvas(
        tenantId: string,
        userId: string,
        options?: { name?: string; chatId?: string; type?: string }
    ): Promise<string> {
        const result = await this.pool.query(`
            INSERT INTO canvases (tenant_id, user_id, canvas_name, chat_id, canvas_type)
            VALUES ($1, $2, $3, $4, $5)
            RETURNING id
        `, [tenantId, userId, options?.name, options?.chatId, options?.type || 'general']);

        return result.rows[0].id;
    }

    async addArtifact(canvasId: string, artifact: ArtifactCreate, createdBy?: string): Promise<
        const result = await this.pool.query(`
            INSERT INTO artifacts (canvas_id, artifact_type, title, content, language, positio
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)
            RETURNING id
        `, [
            canvasId,
            artifact.type,
            artifact.title,
            artifact.content,
            artifact.language,
            artifact.position?.x || 0,
            artifact.position?.y || 0,
            artifact.size?.width || 400,
            artifact.size?.height || 300
        ]);

        const artifactId = result.rows[0].id;

        // Create initial version
        await this.pool.query(`
            INSERT INTO artifact_versions (artifact_id, version, content, created_by)
            VALUES ($1, 1, $2, $3)
        `, [artifactId, artifact.content, createdBy]);

        return artifactId;
```

```
}

async updateArtifact(artifactId: string, content: string, updatedBy?: string): Promise<voi
    // Get current version
    const current = await this.pool.query(
        `SELECT COALESCE(MAX(version), 0) as max_version FROM artifact_versions WHERE arti
        [artifactId]
    );
    const newVersion = current.rows[0].max_version + 1;

    // Update artifact
    await this.pool.query(`
        UPDATE artifacts SET content = $2, updated_at = NOW() WHERE id = $1
    `, [artifactId, content]);

    // Save version
    await this.pool.query(`
        INSERT INTO artifact_versions (artifact_id, version, content, created_by)
        VALUES ($1, $2, $3, $4)
    `, [artifactId, newVersion, content, updatedBy]);
}

async getCanvas(canvasId: string): Promise<any> {
    const canvas = await this.pool.query(`SELECT * FROM canvases WHERE id = $1`, [canvasId]
    const artifacts = await this.pool.query(`SELECT * FROM artifacts WHERE canvas_id = $1 (

    return {
        ...canvas.rows[0],
        artifacts: artifacts.rows
    };
}

async getArtifactVersions(artifactId: string): Promise<any[]> {
    const result = await this.pool.query(`
        SELECT av.*, u.email as created_by_email
        FROM artifact_versions av
        LEFT JOIN users u ON av.created_by = u.id
        WHERE av.artifact_id = $1
        ORDER BY av.version DESC
    `, [artifactId]);

    return result.rows;
}

async moveArtifact(artifactId: string, x: number, y: number): Promise<void> {
    await this.pool.query(`
        UPDATE artifacts SET position_x = $2, position_y = $3, updated_at = NOW() WHERE id
    `, [artifactId, x, y]);
```

```
    }

    async resizeArtifact(artifactId: string, width: number, height: number): Promise<void> {
        await this.pool.query(`
            UPDATE artifacts SET width = $2, height = $3, updated_at = NOW() WHERE id = $1
        `, [artifactId, width, height]);
    }

    async deleteArtifact(artifactId: string): Promise<void> {
        await this.pool.query(`DELETE FROM artifacts WHERE id = $1`, [artifactId]);
    }

    async publishCanvas(canvasId: string): Promise<string> {
        const publishedUrl = `https://canvas.radiant.ai/${canvasId}`;

        await this.pool.query(`
            UPDATE canvases SET is_published = true, published_url = $2, updated_at = NOW() WHE
        `, [canvasId, publishedUrl]);

        return publishedUrl;
    }
}
```