PROYECTO FINAL: "Data Forge" - Plataforma de Transformación de Datos Inteligente

🎯 VISIÓN FINAL

Una plataforma donde cualquier persona pueda transformar cualquier estructura de datos en cualquier otra estructura, mediante una interfaz visual intuitiva asistida por IA.

🏗️ ARQUITECTURA COMPLETA

FRONTEND (React + TypeScript)

```typescript
interface DataForgeApp {
  // 3 Vistas Principales
  views: {
    explorer: DataExplorer;      // Explorar datos fuente
    canvas: MappingCanvas;       // Mapeo visual
    workshop: TransformationWorkshop; // Transformaciones avanzadas
  };

  // Motor de IA integrado
  aiAssistant: {
    suggestMappings: () => MappingSuggestion[];
    detectPatterns: () => DataPattern[];
    learnFromUser: (corrections: UserCorrection[]) => void;
  };
}
```

BACKEND (Node.js + Python Hybrid)

```javascript
// Servicios principales
const services = {
  extraction: 'PDF, Excel, CSV, APIs, Databases',
  transformation: 'Graph-based data reshaping',
  intelligence: 'Machine learning for pattern recognition',
  collaboration: 'Real-time shared workspaces'
};
```

BASE DE DATOS (Políglota)

```
Neo4j   → Para relaciones de datos complejas
PostgreSQL → Para metadatos y usuarios
Redis   → Para caché y sesiones
S3/MinIO → Para almacenamiento de archivos
```

```

## 🚀 FUNCIONALIDADES CLAVE

### 1. DETECCIÓN DE PATRONES INTELIGENTE

```typescript
class PatternDetector {
  async analyze(data: any[]): Promise<DetectionResult> {
    // Detecta automáticamente:
    // - Jerarquías (por indentación, códigos, niveles)
    // - Tipos de datos (fechas, monedas, códigos)
    // - Relaciones (padre-hijo, uno-a-muchos)
    // - Duplicados y anomalías

    return {
      hierarchyLevels: this.findHierarchyLevels(data),
      dataTypes: this.inferDataTypes(data),
      relationships: this.buildRelationshipGraph(data),
      suggestions: this.generateTransformationSuggestions(data)
    };
  }
}
```

2. EDITOR DE TRANSFORMACIONES VISUAL

```jsx
<TransformBuilder>
  {/* Bloque de origen */}
  <DataSource
    preview={sourceData}
    onFieldSelect={(field) => this.startMapping(field)}
  />

  {/* Biblioteca de transformaciones */}
  <TransformationLibrary>
    <TransformCard name="Dividir" icon="split" />
    <TransformCard name="Unir" icon="merge" />
    <TransformCard name="Jerarquizar" icon="tree" />
    <TransformCard name="Calcular" icon="calculator" />
    <TransformCard name="Enriquecer" icon="magic" />
  </TransformationLibrary>

  {/* Pipeline visual */}
  <PipelineCanvas>
    <PipelineStep
      from="Código Cuenta"
```
```

```
      transform="splitByDot"
      to="Nivel 1, Nivel 2, Nivel 3"
    />
    <PipelineStep
      from="Nombres"
      transform="normalizeText"
      to="Nombre Normalizado"
    />
  </PipelineCanvas>

  {/* Vista previa en tiempo real */}
  <LivePreview
    before={originalData}
    after={transformedData}
    diff={showDifferences}
  />
</TransformBuilder>
```

3. MOTOR DE TRANSFORMACIÓN GRÁFICO

```javascript
class GraphTransformationEngine {
  transform(sourceGraph, transformationRules) {
    // Convierte datos a grafo
    const dataGraph = this.buildGraph(sourceData);

    // Aplica transformaciones como operaciones de grafo
    const transformedGraph = this.applyGraphOperations(
      dataGraph,
      transformationRules
    );

    // Convierte de vuelta a la estructura deseada
    return this.graphToStructure(transformedGraph, targetSchema);
  }

  // Operaciones disponibles
  operations = {
    'split-node': this.splitNode,      // Dividir nodo en varios
    'merge-nodes': this.mergeNodes,    // Unir nodos
    'create-edge': this.createEdge,    // Crear relación
    'propagate-values': this.propagate // Propagación de valores
  };
}
```

4. ASISTENTE DE IA CONTEXTUAL

```typescript
interface AIAssistant {
  // Basado en el tipo de documento detectado
  provideContextualHelp(documentType: string): HelpSuggestion[];

  // Sugiere transformaciones basadas en datos similares
  suggestBasedOnPatterns(
    currentData: any[],
    similarPatterns: Pattern[]
  ): Suggestion[];

  // Aprende de las correcciones del usuario
  learnFromFeedback(
    originalSuggestion: Suggestion,
    userCorrection: Correction
  ): void;

  // Ofrece ejemplos interactivos
  showInteractiveExample(
    problem: string,
    solution: string
  ): InteractiveDemo;
}
```

5. BIBLIOTECA DE PLANTILLAS SECTORIALES

```javascript
const industryTemplates = {
  accounting: {
    planDeCuentas: {
      sourcePatterns: ['codigo.nombre.tipo', 'jerarquia-por-sangria'],
      targetSchemas: {
        sqlNormalized: '3 tablas relacionadas',
        jsonHierarchical: 'JSON anidado',
        excelMatrix: 'Matriz de cuentas'
      },
      commonTransformations: [
        'split-codigo-by-dot',
        'build-account-tree',
        'calculate-totals'
      ]
    },
    balanceGeneral: {
      // Patrones específicos para balances
    }
  },
```

```
  inventory: {
    stockManagement: {
      // Plantillas para inventario
    }
  },
  sales: {
    invoices: {
      // Transformaciones para facturas
    }
  }
};
```

🔥 FLUJO DE TRABAJO COMPLETO

PASO 1: CARGA INTELIGENTE

```
Usuario sube archivo → Sistema:
1. Detecta tipo (PDF, Excel, CSV)
2. Extrae estructura automáticamente
3. Pregunta: "¿Qué tipo de documento es?"
   - Plan de Cuentas
   - Factura/Recibo
   - Inventario
   - Lista de Contactos
   - Otro (configurable)
```

PASO 2: ANÁLISIS AUTOMÁTICO

```javascript
// El sistema analiza y muestra:
const analysisResults = {
  detectedStructure: 'Tabular con jerarquía implícita',
  dataQuality: {
    completeness: '95%',
    consistency: '87%',
    duplicates: '12 filas'
  },
  suggestedActions: [
    'Separar columna "Código" en niveles',
    'Normalizar nombres de cuentas',
    'Crear relaciones padre-hijo'
  ]
};
```

PASO 3: MAPEO VISUAL

```
Interfaz dividida en 3 paneles:

[PANEL IZQUIERDO] Datos originales
- Estructura detectada
- Vista previa con colores por tipo
- Capacidad de seleccionar/seleccionar áreas

[PANEL CENTRAL] Área de transformación
- Biblioteca de componentes arrastrables
- Líneas de conexión visuales
- Validaciones en tiempo real

[PANEL DERECHO] Vista previa de destino
- Múltiples formatos: Tabla, JSON, SQL, Gráfico
- Código generado automáticamente
- Opciones de exportación
```

PASO 4: TRANSFORMACIÓN AVANZADA

```javascript
// Usuario puede crear transformaciones complejas
const transformationRecipe = {
  name: 'Normalizar Plan de Cuentas',
  steps: [
    {
      action: 'split',
      target: 'codigo',
      by: '.',
      into: ['nivel1', 'nivel2', 'nivel3']
    },
    {
      action: 'create_hierarchy',
      parent: 'nivel1',
      child: 'nivel2',
      relationship: 'parent_of'
    },
    {
      action: 'derive',
      formula: 'codigo_completo = nivel1 + "." + nivel2',
      target: 'codigo_normalizado'
    }
  ]
};
```

PASO 5: EXPORTACIÓN Y AUTOMATIZACIÓN

```
Formatos de exportación:
- Excel/CSV/JSON
- SQL (CREATE TABLE + INSERT)
- Código (Python, JavaScript, Java)
- API REST endpoint
- Webhook automático

Automatización:
- "Guardar como plantilla"
- "Programar ejecución periódica"
- "Desencadenar por nuevo archivo en carpeta"
- "Integrar con Slack/Teams"
```

💡 INNOVACIONES TÉCNICAS ÚNICAS

1. GRAPHQL PARA TRANSFORMACIONES

```graphql
# Los usuarios pueden definir transformaciones con GraphQL
type Transformation {
  input: DataStructure!
  operations: [Operation!]!
  output: DataStructure!
}

type Operation {
  name: String!
  params: JSON!
  description: String
}

# Ejemplo: Transformar plan de cuentas
mutation TransformPlanDeCuentas {
  transform(
    input: $excelFile,
    operations: [
      { name: "splitColumn", params: { column: "codigo", delimiter: "." } },
      { name: "buildTree", params: { parentField: "nivel1", childField: "nivel2" } }
    ]
  ) {
    success
    result
    preview
```

```
    exportFormats
  }
}
```

2. EXTENSIONES POR COMUNIDAD

```javascript
// Los usuarios pueden crear y compartir extensiones
class CustomTransformation {
  static metadata = {
    name: 'PlanDeCuentas a QuickBooks',
    author: 'Juan Pérez',
    version: '1.0',
    category: 'Accounting'
  };

  static transform(inputData) {
    // Lógica personalizada
    return transformedData;
  }

  static test() {
    // Casos de prueba incluidos
    return testResults;
  }
}

// Marketplace de extensiones
const marketplace = [
  'SAP-to-Odoo Transformer',
  'Excel-to-QuickBooks Migrator',
  'PDF-Invoice-to-JSON Parser'
];
```

3. COLABORACIÓN EN TIEMPO REAL

```typescript
// Múltiples usuarios pueden trabajar juntos
interface CollaborativeSession {
  sessionId: string;
  participants: User[];
  transformations: SharedTransformation[];
  chat: Message[];
  versionHistory: Version[];

  // Funcionalidades colaborativas
```

```
  liveCursors: CursorPosition[]; // Cursor de otros usuarios
  comments: Comment[];           // Comentarios en datos específicos
  approvalWorkflow: Approval[];  // Flujo de aprobación
}
```

4. MODOS DE INTERACCIÓN MÚLTIPLES

```
Modo "Asistente": Paso a paso guiado para principiantes
Modo "Experto": Canvas libre para usuarios avanzados
Modo "Código": Editor de código para desarrolladores
Modo "Automático": IA sugiere transformaciones completas
Modo "Comparación": Side-by-side de múltiples enfoques
```

📊 EJEMPLO COMPLETO: PLAN DE CUENTAS

PROBLEMA ORIGINAL:

```excel
Código  | Nombre              | Tipo
1       | ACTIVO              | Grupo
1.01    | Caja y Bancos       | Subgrupo
1.01.01 | Caja                | Cuenta
1.01.02 | Bancos              | Cuenta
1.02    | Inventarios         | Subgrupo
```

TRANSFORMACIÓN VISUAL:

```
1. Usuario selecciona columna "Código"
2. Arrastra componente "Split by Delimiter"
3. Configura: Delimiter = ".", Output = 3 columns
4. Arrastra componente "Build Hierarchy"
5. Conecta: Parent = nivel1, Children = nivel2, nivel3
6. Vista previa muestra árbol jerárquico
```

RESULTADOS:

```json
// Opción 1: JSON jerárquico
{
  "codigo": "1",
  "nombre": "ACTIVO",
  "tipo": "Grupo",
```

```
  "children": [
    {
      "codigo": "1.01",
      "nombre": "Caja y Bancos",
      "tipo": "Subgrupo",
      "children": [
        { "codigo": "1.01.01", "nombre": "Caja", "tipo": "Cuenta" },
        { "codigo": "1.01.02", "nombre": "Bancos", "tipo": "Cuenta" }
      ]
    }
  ]
}

// Opción 2: SQL normalizado
CREATE TABLE grupos (id, codigo, nombre, tipo);
CREATE TABLE subgrupos (id, grupo_id, codigo, nombre, tipo);
CREATE TABLE cuentas (id, subgrupo_id, codigo, nombre, tipo);

// Opción 3: API REST
GET /api/accounts/1.01.01
{
  "path": ["ACTIVO", "Caja y Bancos", "Caja"],
  "full_code": "1.01.01",
  "type": "Cuenta"
}
```

🚀 PLAN DE IMPLEMENTACIÓN EN 90 DÍAS

SEMANAS 1-4: MVP BÁSICO

```
✅ Frontend: Componentes básicos de UI
✅ Backend: API para subida/descarga
✅ Transformaciones: Split, Join, Rename
✅ Exportación: CSV, JSON
```

SEMANAS 5-8: INTELIGENCIA BÁSICA

```
🔧 Detección automática de tipos de datos
🔧 Sugerencias de transformaciones simples
🔧 Plantillas predefinidas (5-10 comunes)
🔧 Historial de operaciones
```

SEMANAS 9-12: TRANSFORMACIONES AVANZADAS

```
⚡ Motor de transformaciones gráficas
⚡ Jerarquías y relaciones
⚡ Scripting personalizado
⚡ Conexión a APIs externas
```

SEMANAS 13-16: COLABORACIÓN

```
👥 Compartición de trabajos
👥 Comentarios y revisión
👥 Control de versiones
👥 Roles y permisos
```

SEMANAS 17-20: IA AVANZADA

```
🤖 Aprendizaje de patrones
🤖 Generación de código
🤖 Corrección automática
🤖 Recomendaciones contextuales
```

ECOSISTEMA:

```
Data Forge Core → Plataforma principal
Data Forge Connect → Conectores empresariales
Data Forge AI → Asistente avanzado de IA
Data Forge Teams → Colaboración empresarial
Data Forge Automate → Automatización de flujos
```

🌟 CONCLUSIÓN FINAL

Data Forge no es solo otra herramienta de ETL. Es un entorno de desarrollo visual para transformación de datos donde:

1. Cualquier persona puede reestructurar datos sin programar
2. La IA asiste pero el usuario mantiene el control
3. Se aprende con cada uso y mejora continuamente
4. Se colabora y comparte conocimiento colectivo