📋 Formularios Dinámicos - Documentación Técnica

🎯 Descripción General

El sistema de formularios dinámicos permite crear, gestionar y procesar formularios configurables que se adaptan automáticamente a los esquemas JSON esperados por el frontend. Esta implementación utiliza **arquitectura hexagonal** para mantener la separación de responsabilidades y está completamente integrada con la base de datos SQL Server.

🏗️ Arquitectura

```
src/forms/
├── domain/
│   ├── entities/           # Entidades del dominio (Form, FormField, FormSubmission)
│   └── ports/              # Interfaces/Puertos (repositorios y servicios)
├── application/
│   └── services/           # Lógica de aplicación (FormsService, FormSubmissionService)
└── infrastructure/
    ├── controllers/        # Controladores REST (FormsController)
    ├── entities/           # Entidades de TypeORM
    ├── repositories/       # Implementaciones de repositorios
    ├── services/           # Servicios de infraestructura y validación
    └── dto/               # Data Transfer Objects
```

📊 Formularios Disponibles

1. **Alta de Automotor** (`vehicle-registration`)

- **Título**: Alta de Automotor
- **Subtitle**: Sistema de registro de nuevos vehículos - Formulario AUFA0030_CBA
- **Formulario**: AUFA0030_CBA
- **Categoría**: vehiculos
- **Endpoint**: `POST http://localhost:3000/api/automotor/alta-modificacion`

- **Método**: POST

Secciones organizadas:

1. **Datos Básicos del Vehículo** (8 campos)
2. **Información RNPA** (8 campos)
3. **Registro y Documentación** (7 campos)
4. **Propietario Inicial** (6 campos)
5. **Datos Técnicos** (8 campos)
6. **Documentación y Control** (5 campos)

Total: 42 campos configurados

2. Transferencia de Automotor (`vehicle-transfer`)

- **Título**: Transferencia de Automotor
- **Subtitle**: Sistema de consulta y transferencia de vehículos - Formulario AUFA0090_CBA - TAXIDESI
- **Formulario**: AUFA0090_CBA - TAXIDESI
- **Categoría**: vehiculos
- **Endpoint**: `PUT http://localhost:3000/api/automotor/transferencia`
- **Método**: PUT

Secciones organizadas:

1. **Datos del Vehículo** (17 campos)
2. **Datos de la Transferencia** (5 campos)
3. **Propietarios** (10 campos)
4. **Documentación y Observaciones** (3 campos)

🔌 API Endpoints

📋 Gestión de Formularios

`GET /forms`

Retorna todos los formularios activos con paginación y filtros.

Query Parameters:

- `page` (opcional): Número de página (mínimo 1)

- `limit` (opcional): Elementos por página (1-100)
- `category` (opcional): Filtrar por categoría
- `isActive` (opcional): Filtrar por estado activo
- `search` (opcional): Búsqueda por título o descripción

Respuesta:


```
{
  "value": [
    {
      "id": "vehicle-registration",
      "title": "Alta de Automotor",
      "description": "AUFA0030_CBA - Registre un nuevo vehículo en el sistema tributario",
      "category": "vehiculos",
      "isActive": true,
      "createdAt": "2025-07-17T14:38:04.880Z",
      "updatedAt": "2025-07-17T14:38:04.880Z"
    }
  ],
  "Count": 2
}
```


`GET /forms/:id`

Retorna un formulario específico con todos sus campos y configuración completa.

Ejemplo: `GET /forms/vehicle-registration`

Respuesta:

```
{
  "id": "vehicle-registration",
  "title": "Alta de Automotor",
  "subtitle": "Sistema de registro de nuevos vehículos - Formulario AUFA0030_CBA",
  "description": "AUFA0030_CBA - Registre un nuevo vehículo en el sistema tributario",
  "category": "vehiculos",
  "isActive": true,
  "version": "1.0",
  "buttonsConfig": {
    "submit": "Registrar Vehículo",
    "cancel": "Cancelar",
    "showBack": true,
    "backRoute": "/home"
  },
  "layoutConfig": {
    "cols": 2,
    "gap": "16px",
    "rowHeight": "85px"
  },
  "submissionEndpoint": "http://localhost:3000/api/automotor/alta-modificacion",
  "submissionMethod": "POST",
  "submissionSchema": {
    "type": "object",
    "mapping": {
      "vehiculo": {
        "type": "object",
        "mapping": {
          "patente": { "source": "patente" },
          "tipoVehiculo": { "source": "tipo_vehiculo" },
          "anioFabricacion": { "source": "anio_fabricacion", "type": "number" }
        }
      },
      "propietario": {
        "type": "object",
        "mapping": {
          "cuit": { "source": "cuit_propietario" },
          "porcentajePropiedad": { "source": "porcentaje_propiedad", "type": "number" },
          "esResponsable": { "source": "es_responsable", "type": "boolean" }
        }
      }
    }
  },
  "sectionsConfig": [
```

```
      {
        "title": "Datos Básicos del Vehículo",
        "fields": ["patente", "tipo_vehiculo", "anio_fabricacion", "origen_rnpa"]
      }
    ],
    "fields": [
      {
        "id": "uuid",
        "formId": "vehicle-registration",
        "name": "patente",
        "type": "plate",
        "label": "Dominio (Patente)",
        "description": "ATR_DOMINIO - Patente del vehículo a registrar",
        "placeholder": "AA111AA o AAA111",
        "required": true,
        "help": "Ingrese la nueva patente asignada al vehículo",
        "orderIndex": 1,
        "gridPosition": { "row": 1, "col": 1 }
      }
    ]
  }
```


`GET /forms/category/:category`

Retorna formularios por categoría específica.

Ejemplo: `GET /forms/category/vehiculos`

📝 Envío de Formularios

`POST /forms/:id/submit`

Envía un formulario con transformación automática de datos.

Ejemplo: `POST /forms/vehicle-registration/submit`

Request Body:

```
{
  "patente": "BB456CC",
  "tipo_vehiculo": "AUTO",
  "anio_fabricacion": 2020,
  "cuit_propietario": "20-12345678-9",
  "porcentaje_propiedad": 100,
  "es_responsable": true,
  "numero_motor": "XYZ123456",
  "numero_chasis": "ABC789012",
  "userAgent": "Mozilla/5.0...",
  "ipAddress": "192.168.1.1",
  "userId": "user123",
  "sessionId": "session456"
}
```


Transformación automática según `submissionSchema` :


```
{
  "vehiculo": {
    "patente": "BB456CC",
    "tipoVehiculo": "AUTO",
    "anioFabricacion": 2020,
    "numeroMotor": "XYZ123456",
    "numeroChasis": "ABC789012"
  },
  "propietario": {
    "cuit": "20-12345678-9",
    "porcentajePropiedad": 100,
    "esResponsable": true
  },
  "registro": {
    "usuarioAlta": "system",
    "fechaAltaVehiculo": "2025-07-17T14:38:05.000Z"
  },
  "documentacion": {
    "documentosPresentados": [],
    "verificacionPolicial": false
  }
}
```

📊 Historial y Estadísticas

`GET /forms/:id/submissions`

Obtiene el historial de envíos de un formulario específico.

Ejemplo: `GET /forms/vehicle-registration/submissions`

Respuesta:


```
[
  {
    "id": "uuid",
    "formId": "vehicle-registration",
    "data": { "patente": "BB456CC", "..." },
    "transformedData": { "vehiculo": { "patente": "BB456CC" } },
    "response": { "success": true },
    "userAgent": "Mozilla/5.0...",
    "ipAddress": "192.168.1.1",
    "submittedAt": "2025-07-17T14:38:05.000Z"
  }
]
```


`GET /forms/:id/stats`

Obtiene estadísticas de uso de un formulario.

Ejemplo: `GET /forms/vehicle-registration/stats`

Respuesta:


```
{
  "totalSubmissions": 150,
  "successfulSubmissions": 145,
  "failedSubmissions": 5,
  "averageCompletionTime": "00:05:30",
  "mostUsedFields": ["patente", "cuit_propietario"],
  "submissionsByDay": [
    { "date": "2025-07-17", "count": 25 }
  ]
}
```

🔍 Validación y Utilidades

`POST /forms/validate`

Valida un campo específico sin enviar el formulario completo.

Request Body:

```
{
  "endpoint": "/api/automotor",
  "value": "BB456CC"
}
```

Respuesta:

```
{
  "isValid": true,
  "data": {
    "tipo_rnpa_descripcion": "Automóvil"
  },
  "errors": []
}
```

`POST /forms/autocomplete`

Obtiene datos de autocompletado para campos con lookup automático.

Request Body:

```
{
  "endpoint": "/api/contribuyente",
  "value": "20-12345678",
  "context": {
    "formId": "vehicle-registration",
    "fieldName": "cuit_propietario"
  }
}
```

Respuesta:

```
{
  "suggestions": [
    {
      "value": "20-12345678-9",
      "label": "JUAN PEREZ",
      "additionalData": {
        "nombre_completo": "JUAN PEREZ",
        "tipo_persona": "FISICA"
      }
    }
  ]
}
```


`POST /forms/dependent-options`

Obtiene opciones dependientes basadas en la selección de otros campos.

Request Body:


```
{
  "endpoint": "/api/modelos",
  "parentValue": "FORD"
}
```


Respuesta:


```
{
  "options": [
    { "id": "FOCUS", "label": "Focus" },
    { "id": "FIESTA", "label": "Fiesta" },
    { "id": "ECOSPORT", "label": "EcoSport" }
  ]
}
```

🔧 Configuración de Campos

Tipos de Campo Soportados

Tipo	Descripción	Validaciones	Eventos
`plate`	Patentes de vehículos	Formato AA111AA o AAA111	onValidation
`cuit`	CUIT/CUIL	Formato XX-XXXXXXXX-X	onValidation
`text`	Texto libre	maxLength, pattern	onChange, onBlur
`number`	Números	min, max, integer	onChange
`date`	Fechas	dateRange, format	onChange
`datetime`	Fecha y hora	dateTimeRange	onChange
`select`	Lista desplegable	options, required	onChange
`radio`	Botones de radio	options, required	onChange
`multiselect`	Selección múltiple	options, minItems, maxItems	onChange
`checkbox`	Casilla de verificación	required	onChange
`textarea`	Texto largo	maxLength, rows	onChange, onBlur
`percentage`	Porcentajes	min: 0, max: 100	onChange
`money`	Valores monetarios	currency, precision	onChange
`hidden`	Campos ocultos	defaultValue	-

Estructura de Campo


```
{
  "id": "uuid-generado",
  "formId": "vehicle-registration",
  "name": "patente",
  "type": "plate",
  "label": "Dominio (Patente)",
  "description": "ATR_DOMINIO - Patente del vehículo a registrar",
  "placeholder": "AA111AA o AAA111",
  "required": true,
  "readonly": false,
  "help": "Ingrese la nueva patente asignada al vehículo",
  "defaultValue": null,
  "min": null,
  "max": null,
  "maxLength": null,
  "orderIndex": 1,
  "gridPosition": { "row": 1, "col": 1, "colspan": 1 },
  "options": [],
  "validationRules": {
    "customValidation": "validateNewPlate"
  },
  "events": {
    "onValidation": {
      "action": "both",
      "endpoint": "/api/automotor",
      "fields": {
        "tipo_rnpa_descripcion": "tipo_rnpa_descripcion"
      },
      "debounceTime": 500
    }
  }
}
```

Validaciones Avanzadas

Validaciones Personalizadas

```
{
  "validationRules": {
    "customValidation": "validateNewPlate",
    "pattern": "^[A-Z]{2}\\d{3}[A-Z]{2}$",
    "message": "Formato de patente inválido",
    "min": 1900,
    "max": 2025,
    "required": true,
    "unique": {
      "table": "AUTOMOTORES",
      "field": "patente",
      "message": "Ya existe un vehículo con esta patente"
    }
  }
}
```

Eventos de Campo

```
{
  "events": {
    "onValidation": {
      "action": "both",
      "endpoint": "/api/automotor",
      "fields": {
        "modelo_descripcion": "modelo_descripcion",
        "marca_descripcion": "marca_descripcion"
      },
      "debounceTime": 800
    },
    "onChange": "calculateTotal",
    "onBlur": "validateField"
  }
}
```

🗄️ Base de Datos

Estructura de Tablas

Tabla `forms`

```sql
CREATE TABLE forms (
  id VARCHAR(100) PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  subtitle VARCHAR(500),
  description VARCHAR(500),
  category VARCHAR(100) NOT NULL,
  isActive BIT DEFAULT 1,
  version VARCHAR(20),
  buttonsConfig NVARCHAR(MAX),
  layoutConfig NVARCHAR(MAX),
  submissionEndpoint VARCHAR(500),
  submissionMethod VARCHAR(10),
  submissionSchema NVARCHAR(MAX),
  sectionsConfig NVARCHAR(MAX),
  validationsConfig NVARCHAR(MAX),
  eventsConfig NVARCHAR(MAX),
  createdAt DATETIME2 DEFAULT GETDATE(),
  updatedAt DATETIME2 DEFAULT GETDATE()
);
```

Tabla `form_fields`

```sql
CREATE TABLE form_fields (
  id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  formId VARCHAR(100) NOT NULL,
  name VARCHAR(100) NOT NULL,
  type VARCHAR(50) NOT NULL,
  label VARCHAR(255) NOT NULL,
  description VARCHAR(500),
  placeholder VARCHAR(255),
  required BIT DEFAULT 0,
  readonly BIT DEFAULT 0,
  help VARCHAR(500),
  defaultValue VARCHAR(255),
  min INT,
  max INT,
  maxLength INT,
  orderIndex INT NOT NULL,
  gridPosition NVARCHAR(MAX),
  options NVARCHAR(MAX),
  validationRules NVARCHAR(MAX),
  events NVARCHAR(MAX),
  createdAt DATETIME2 DEFAULT GETDATE(),
  updatedAt DATETIME2 DEFAULT GETDATE(),
  FOREIGN KEY (formId) REFERENCES forms(id)
);
```

Tabla `form_submissions`

```sql
CREATE TABLE form_submissions (
  id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  formId VARCHAR(100) NOT NULL,
  data NVARCHAR(MAX) NOT NULL,
  transformedData NVARCHAR(MAX),
  response NVARCHAR(MAX),
  userAgent VARCHAR(500),
  ipAddress VARCHAR(45),
  userId VARCHAR(100),
  sessionId VARCHAR(100),
  submittedAt DATETIME2 DEFAULT GETDATE(),
  FOREIGN KEY (formId) REFERENCES forms(id)
);
```

🚀 Inicialización y Seed

Servicios de Poblado Automático

`FormSeedService`

- Crea los formularios base con configuración completa
- Se ejecuta automáticamente al iniciar la aplicación
- Elimina y recrea formularios para mantener configuración actualizada

`FormFieldSeedService`

- Crea todos los campos de los formularios con configuración detallada
- Maneja 42 campos para vehicle-registration
- Configura validaciones, eventos y propiedades específicas

Proceso de Inicialización

1. **Verificación**: Comprueba formularios existentes
2. **Limpieza**: Elimina formularios obsoletos
3. **Creación**: Inserta formularios actualizados
4. **Campos**: Crea todos los campos con configuración
5. **Verificación**: Confirma inserción exitosa

🔄 Integración con Frontend

Flujo de Trabajo Típico

1. **Obtener Formularios**: `GET /forms` para listar formularios disponibles
2. **Cargar Formulario**: `GET /forms/:id` para obtener configuración completa
3. **Renderizar Dinámicamente**: Usar configuración para generar UI
4. **Validación en Tiempo Real**: `POST /forms/validate` para validar campos
5. **Autocompletado**: `POST /forms/autocomplete` para lookup automático
6. **Envío**: `POST /forms/:id/submit` con transformación automática
7. **Seguimiento**: `GET /forms/:id/submissions` para historial

Características para Frontend

- **Configuración Dinámica**: No necesita hardcodear formularios

- **Validación Consistente**: Mismas reglas en frontend y backend
- **Transformación Automática**: Convierte datos al formato esperado
- **Eventos Configurables**: onBlur, onChange, onValidation
- **Autocompletado Inteligente**: Lookup automático con debounce
- **Responsive Layout**: Configuración de grid adaptable

🛠️ Desarrollo y Extensión

Agregar Nuevo Formulario

1. **Definir en FormSeedService**:

```
private async createNewForm(): Promise<void> {
  const form = {
    id: 'new-form',
    title: 'Nuevo Formulario',
    // ... configuración completa
  };
  await this.formRepository.create(form);
}
```

2. **Crear Campos en FormFieldSeedService**:

```
private async createNewFormFields(): Promise<void> {
  const fields = [
    {
      formId: 'new-form',
      name: 'campo1',
      type: 'text',
      // ... configuración del campo
    }
  ];
  // ... insertar campos
}
```

3. **Configurar Endpoints**: Definir endpoints de destino para envío
4. **Agregar Validaciones**: Implementar reglas específicas
5. **Testear**: Verificar funcionamiento completo

Tipos de Campo Personalizados

Para agregar nuevos tipos de campo:

1. **Actualizar enum** de tipos permitidos
2. **Implementar validación** específica
3. **Configurar eventos** necesarios
4. **Documentar uso** y propiedades

📝 Notas Técnicas

- **UUIDs**: Identificadores únicos para campos
- **JSON Storage**: Configuraciones almacenadas como JSON strings
- **Validación Dual**: Frontend y backend
- **Transformación Automática**: Schema mapping configurable
- **Auditoría Completa**: Tracking de todos los envíos
- **Performance**: Índices en campos frecuentemente consultados
- **Escalabilidad**: Arquitectura preparada para múltiples formularios

🔒 Seguridad

- **Validación Server-Side**: Todas las validaciones se ejecutan en el backend
- **Sanitización**: Datos de entrada sanitizados antes del procesamiento
- **Auditoría**: Registro completo de envíos con IP y User Agent
- **Rate Limiting**: Control de frecuencia de envíos
- **CORS**: Configuración adecuada para requests cross-origin

🚀 Estado Actual

✅ Sistema Completamente Funcional

- 2 formularios operativos (Alta y Transferencia)
- 42+ campos configurados con validaciones
- API REST completa con documentación Swagger
- Base de datos poblada automáticamente
- Transformación de datos funcionando
- Integración con backend de automotores

🎯 **Listo para producción y uso en frontend dinámico.**

"cuit_propietario": "20-12345678-9",
"porcentaje_propiedad": 100,
"es_responsable": true,
"numero_motor": "ABC123456",
"numero_chasis": "DEF789012"
}

Transformación automática según `submissionSchema`:

```json
{
  "ovpId": "AA123BB",
  "tipoOperacion": "ALTA",
  "propietario": {
    "cuit": "20-12345678-9",
    "porcentajePropiedad": 100,
    "esResponsable": true
  },
  "vehiculo": {
    "patente": "AA123BB",
    "anioFabricacion": 2020,
    "numeroMotor": "ABC123456",
    "numeroChasis": "DEF789012"
  }
}
```

Validación

`POST /forms/validate`

Valida datos de formulario sin enviar.

Request Body:

```
{
  "formId": "vehicle-registration",
  "data": {
    "patente": "AA123BB",
    "cuit_propietario": "20-12345678-9"
  }
}
```

Utilidades

`POST /forms/autocomplete`

Obtiene datos de autocompletado para campos específicos.

Request Body:

```
{
  "field": "patente",
  "query": "AA123",
  "formId": "vehicle-registration"
}
```

`POST /forms/dependent-options`

Obtiene opciones dependientes basadas en otros campos.

Request Body:

```
{
  "field": "modelo",
  "dependencies": {
    "marca": "FORD"
  },
  "formId": "vehicle-registration"
}
```

🔧 Configuración de Campos

Tipos de Campo Soportados

- `text` : Campos de texto
- `number` : Campos numéricos
- `date` : Selectores de fecha
- `select` : Listas desplegables
- `checkbox` : Casillas de verificación
- `radio` : Botones de radio

Validaciones

Cada campo puede tener reglas de validación en `validationConfig`:

```
{
  "pattern": "^[A-Z]{2}\\d{3}[A-Z]{2}$",
  "message": "Formato de patente inválido",
  "min": 1900,
  "max": 2025
}
```

Eventos

Los campos pueden tener eventos configurados en `eventsConfig`:

```
{
  "onBlur": "patenteLookup",
  "onChange": "calculateValue"
}
```

🗄️ Base de Datos

Tabla `forms`

Almacena la configuración de formularios.

Tabla `form_fields`

Almacena los campos de cada formulario con su configuración.

Tabla `form_submissions`

Almacena los envíos de formularios para auditoría.

🚀 Inicialización

Los formularios se pueblan automáticamente al iniciar la aplicación mediante:

- `FormSeedService`: Crea los formularios base

- `FormFieldSeedService` : Crea los campos de los formularios

🔄 Integración con Frontend

El sistema está diseñado para ser consumido por frontends que esperan:

1. Configuración dinámica de formularios
2. Validación en tiempo real
3. Transformación automática de datos
4. Eventos de autocompletado y validación

🛠️ Desarrollo

Para agregar nuevos formularios:

1. **Crear formulario** en `FormSeedService`
2. **Definir campos** en `FormFieldSeedService`
3. **Configurar transformación** en `submissionSchema`
4. **Agregar validaciones** específicas
5. **Testear endpoints** de integración

📝 Notas Técnicas

- Los formularios usan **UUID** como identificadores
- Las configuraciones se almacenan como **JSON strings**
- La validación se ejecuta tanto en frontend como backend
- Los datos se transforman automáticamente antes del envío al endpoint destino
- El sistema es extensible para agregar nuevos tipos de campo y validaciones