

Voici le code d'une application web qui me permet de générer un manifest iiif importable dans Transkribus, à partir de données publiées dans Nakala.

Voici le prompt de départ :

J'aimerais une petite application qui me permette de générer des manifests en fonction du DOI et de l'ID d'un fichier Nakala.

L'application me demanderait les informations nécessaires : DOI, ID de chaque page, identifiant de la lettre, et génèrerait un manifest iiif exactement sur le modèle de celui que je viens de te donner.

j'aimerais que l'application puisse récupérer les dimensions des images automatiquement. Par exemple, pour la lettre SPA\_2490, pour obtenir les dimensions exactes de chaque page, on peut consulter le fichier info.json de chaque image via l'API IIIF de Nakala.

Voici des exemples d'URLs à utiliser :

Pour la Page 1 :

<https://api.nakala.fr/iiif/10.34847/nkl.ca86te47/58b91ba17fb30e3d27881f56105322dfec65f1a6/info.json>

Pour la Page 2 :

<https://api.nakala.fr/iiif/10.34847/nkl.ca86te47/e17b90833e6cf131a1934d8d772b621143a0efb/info.json>

Dans la réponse JSON, on trouve les propriétés width et height qui donneront les dimensions exactes de chaque image.

J'aimerais aussi que le titre soit le même que l'ID Lettre, soit SPA\_XXXX et qu'il reprenne le début du titre global de la lettre. Voici un exemple : dans la colonne F «<http://nakala.fr/terms#title>» de mon CSV je peux lire des données qui ont toutes les même schéma comme par exemple «SPA\_2469 Letter from Sergey Sergeevich Prokofiev to Grzegorz Fitelberg». J'aimerais que l'ID Lettre et le Titre soient alors «SPA\_2469», pré-remplis, avec possibilité de les modifier si besoin. Voici ma clef d'API : 607d230a-347b-2b03-2000-eee194bf74de



Pour éliminer les problèmes CORS récurrent, voici l'optique adoptée :




## Saisie assistée

### Comment ça marche :





- 1 **Bouton "Ouvrir URLs info.json"** → Ouvre automatiquement tous les onglets nécessaires
- 2 **Champ "Coller les dimensions ici"** → Accepte plusieurs formats :
  - **JSON complet** : `{"width":2692,"height":3430}` (copié directement)
  - **Format simple** : 2692x3430 ou 2692,3430
  - **Juste les chiffres** : 2692 3430

### Workflow ultra-rapide :

- 1  Saisir DOI et hash des pages
- 2  Clic sur "Ouvrir URLs info.json" → Tous les onglets s'ouvrent

- 3  Dans chaque onglet, **Ctrl+A puis Ctrl+C** (copier tout le JSON)
- 4  Coller dans le champ "Coller les dimensions ici"
- 5  Clic "Appliquer" → Les dimensions se remplissent automatiquement

### Avantages :

-  **Plus de problèmes CORS**
-  **Plus rapide** que la saisie manuelle
-  **Plus intelligent** : accepte plein de formats
-  **Plus fiable** : utilise l'API officielle dans le navigateur

Je suis très satisfait car le manifest iiif produit est parfaitement accepté par Transkribus.

Voici le code complet :

```
import React, { useState, useRef } from 'react';
import { Download, Upload, Plus, Minus, FileText, ExternalLink, CheckCircle,
AlertCircle } from 'lucide-react';
import * as Papa from 'papaparse';

const IIIFManifestGenerator = () => {
  const [csvData, setCsvData] = useState([]);
  const [selectedLetter, setSelectedLetter] = useState("");
  const [manifestData, setManifestData] = useState({
    doi: "",
    letterId: "",
    title: "",
    creator: "",
    dateIssued: "",
    attribution: 'Nicolas Moron / NAKALA',
    license: 'http://creativecommons.org/licenses/by-nc/4.0/',
    pages: [
      { hash: "", width: 2692, height: 3430, loading: false, validated: false, error: null,
dimensionsInput: "" }
    ]
  });
  const [generatedManifest, setGeneratedManifest] = useState("");
  const [isGenerating, setIsGenerating] = useState(false);
  const fileInputRef = useRef(null);

  // Fonction pour générer l'URL de l'API et l'ouvrir dans un nouvel onglet
  const openInfoUrl = (doi, hash) => {
    const infoUrl = `https://api.nakala.fr/iiif/${doi}/${hash}/info.json`;
    window.open(infoUrl, '_blank');
  };

  // Fonction pour valider et parser les dimensions saisies manuellement
  const parseDimensions = (input, pageIndex) => {
```

```

if (!input.trim()) return;

try {
  // Essayer de parser comme JSON complet
  const jsonData = JSON.parse(input);
  if (jsonData.width && jsonData.height) {
    updatePageDimensions(pageIndex, jsonData.width, jsonData.height);
    return;
  }
} catch (e) {
  // Si ce n'est pas du JSON, essayer d'autres formats
}

// Essayer de parser comme "width x height" ou "width,height"
const dimensionRegex = /(\d+)[s,xx](\d+)/;
const match = input.match(dimensionRegex);
if (match) {
  const width = parseInt(match[1]);
  const height = parseInt(match[2]);
  updatePageDimensions(pageIndex, width, height);
  return;
}

// Essayer de parser comme deux nombres séparés
const numbers = input.match(/\d+/g);
if (numbers && numbers.length >= 2) {
  const width = parseInt(numbers[0]);
  const height = parseInt(numbers[1]);
  updatePageDimensions(pageIndex, width, height);
}
};

const updatePageDimensions = (pageIndex, width, height) => {
  const newPages = [...manifestData.pages];
  newPages[pageIndex] = {
    ...newPages[pageIndex],
    width: width,
    height: height,
    validated: true,
    error: null,
    dimensionsInput: "
  };
  setManifestData({ ...manifestData, pages: newPages });
};

// Validation du DOI en temps réel
const validateDoi = (doi) => {
  const doiRegex = /^10\.\d{4,}\.\d{,}.*$/;
  return doiRegex.test(doi);
};

```

```

// Validation du hash en temps réel
const validateHash = (hash) => {
  const hashRegex = /^[a-f0-9]{40}$/i;
  return hashRegex.test(hash);
};

const handleFileUpload = (event) => {
  const file = event.target.files[0];
  if (file) {
    Papa.parse(file, {
      complete: (results) => {
        setCsvData(results.data);
      },
      header: true,
      skipEmptyLines: true
    });
  }
};

const handleLetterSelect = (letterItem) => {
  const letterData = csvData.find(item =>
    item['Linked in item'] === letterItem
  );

  if (letterData) {
    const titleMatch = letterData['http://nakala.fr/terms#title']?.match(/^(SPA_\d+)/);
    const letterId = titleMatch ? titleMatch[1] : "";

    setManifestData({
      ...manifestData,
      letterId: letterId,
      title: letterData['http://nakala.fr/terms#title'] || "",
      creator: letterData['http://nakala.fr/terms#creator'] || "",
      dateIssued: letterData['http://nakala.fr/terms#created'] || ""
    });
    setSelectedLetter(letterItem);
  }
};

const addPage = () => {
  setManifestData({
    ...manifestData,
    pages: [...manifestData.pages, {
      hash: "",
      width: 2692,
      height: 3430,
      loading: false,
      validated: false,
      error: null,
    }
  ]
});

```

```

        dimensionsInput: "
    }}
  });
};

```

```

const removePage = (index) => {
  if (manifestData.pages.length > 1) {
    const newPages = manifestData.pages.filter((_, i) => i !== index);
    setManifestData({ ...manifestData, pages: newPages });
  }
};

```

```

const updatePage = (index, field, value) => {
  const newPages = [...manifestData.pages];
  newPages[index][field] = value;

  // Réinitialiser la validation si le hash change
  if (field === 'hash') {
    newPages[index].validated = false;
    newPages[index].error = null;
  }
}

```

```

  setManifestData({ ...manifestData, pages: newPages });
};

```

```

// Récupérer automatiquement les dimensions de toutes les pages
const openAllInfoUrls = () => {
  if (!manifestData.doi) {
    alert('Veuillez d\'abord saisir le DOI');
    return;
  }
}

```

```

manifestData.pages.forEach((page, index) => {
  if (page.hash && validateHash(page.hash)) {
    setTimeout(() => {
      openInfoUrl(manifestData.doi, page.hash);
    }, index * 500); // Délai pour éviter d'ouvrir tous les onglets en même temps
  }
});

```

```

  alert(`${manifestData.pages.filter(p => p.hash && validateHash(p.hash)).length}
onglets vont s'ouvrir. Copiez-collez les dimensions dans les champs ci-dessous.`);
};

```

```

const generateManifest = () => {
  setIsGenerating(true);
}

```

```

setTimeout(() => {
  const manifest = {
    "@context": "http://iiif.io/api/presentation/2/context.json",
  }
}

```

```

"@id": `https://nakala.fr/iiif/${manifestData.doi}/manifest`,
"@type": "sc:Manifest",
"label": manifestData.letterId,
"attribution": manifestData.attribution,
"license": manifestData.license,
"logo": "https://nakala.fr/logo/",
"metadata": [
  {
    "label": "Id",
    "value": manifestData.letterId
  },
  {
    "label": "Title",
    "value": manifestData.title
  },
  {
    "label": "Creator",
    "value": manifestData.creator
  },
  {
    "label": "Date Issued",
    "value": manifestData.dateIssued
  },
  {
    "label": "Type",
    "value": "manuscript"
  }
],
"sequences": [
  {
    "@id": `https://nakala.fr/iiif/${manifestData.doi}/sequence/normal`,
    "@type": "sc:Sequence",
    "label": manifestData.letterId,
    "viewingHint": "paged",
    "startCanvas": `https://nakala.fr/iiif/${manifestData.doi}/canvas/page1`,
    "canvases": manifestData.pages.map((page, index) => ({
      "@id": `https://nakala.fr/iiif/${manifestData.doi}/canvas/page${index + 1}`,
      "@type": "sc:Canvas",
      "label": `Page ${index + 1}`,
      "height": page.height,
      "width": page.width,
      "thumbnail": {
        "@id": `https://api.nakala.fr/iiif/${manifestData.doi}/${page.hash}/full/,150/0/default.jpg`,
        "@type": "dctypes:Image"
      },
      "images": [
        {
          "@id": `https://nakala.fr/iiif/${manifestData.doi}/annotation/page${index +
1}`,

```

```

        "@type": "oa:Annotation",
        "motivation": "sc:painting",
        "on": `https://nakala.fr/iiif/${manifestData.doi}/canvas/page${index + 1}`,
        "resource": {
            "@id": `https://api.nakala.fr/iiif/${manifestData.doi}/${page.hash}/full/full/0/default.jpg`,
            "@type": "dctypes:Image",
            "format": "image/jpeg",
            "height": page.height,
            "width": page.width,
            "service": {
                "@context": "http://iiif.io/api/image/2/context.json",
                "@id": `https://api.nakala.fr/iiif/${manifestData.doi}/${page.hash}`,
                "profile": "http://iiif.io/api/image/2/level2.json"
            }
        }
    }
}
]
}))
}
]
};

```

```

const manifestJson = JSON.stringify(manifest, null, 2);
setGeneratedManifest(manifestJson);
setIsGenerating(false);
}, 500);
};

```

```

const downloadManifest = () => {
    const blob = new Blob([generatedManifest], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `manifest_${manifestData.letterId || 'document'}.json`;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);
};

```

```

// Vérifier si le formulaire est valide
const isFormValid = () => {
    return manifestData.doi &&
        validateDoi(manifestData.doi) &&
        manifestData.letterId &&
        manifestData.pages.every(p => p.hash && validateHash(p.hash));
};

```

```

return (

```

```

<div className="max-w-6xl mx-auto p-6 bg-white">
  <h1 className="text-3xl font-bold text-gray-800 mb-8">
    Générateur de Manifests IIIF Nakala
  </h1>

  {/ * CSV Upload Section */}
  <div className="mb-8 p-6 border-2 border-dashed border-gray-300 rounded-
lg">
    <div className="text-center">
      <Upload className="mx-auto h-12 w-12 text-gray-400 mb-4" />
      <div className="flex flex-col items-center gap-4">
        <input
          type="file"
          accept=".csv"
          onChange={handleFileUpload}
          ref={fileInputRef}
          className="hidden"
        />
        <button
          onClick={() => fileInputRef.current?.click()}
          className="bg-blue-500 hover:bg-blue-600 text-white px-6 py-2 rounded-lg
transition-colors"
        >
          Charger le CSV myNKL
        </button>
        {csvData.length > 0 && (
          <p className="text-green-600 font-semibold">
            ✓ {csvData.length} lettres chargées
          </p>
        )}
      </div>
    </div>
  </div>

  <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
    {/ * Form Section */}
    <div className="space-y-6">
      {/ * Letter Selection */}
      {csvData.length > 0 && (
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Sélectionner une lettre
          </label>
          <select
            value={selectedLetter}
            onChange={(e) => handleLetterSelect(e.target.value)}
            className="w-full p-3 border border-gray-300 rounded-md focus:ring-2
focus:ring-blue-500"
          >
            <option value="">--- Choisir une lettre ---</option>

```



```

        {csvData.map((item, index) => (
          <option key={index} value={item['Linked in item']}>
            {item['http://nakala.fr/terms#title']}
          </option>
        ))}
      </select>
    </div>
  )}

  { /* Basic Information */ }
  <div className="space-y-4">
    <h3 className="text-lg font-semibold text-gray-800">Informations de base</h3>

    <div>
      <label className="block text-sm font-medium text-gray-700 mb-1">
        DOI Nakala *
      </label>
      <div className="relative">
        <input
          type="text"
          value={manifestData.doi}
          onChange={(e) => setManifestData({...manifestData, doi: e.target.value})}
          placeholder="10.34847/nkl.xxxxxxxx"
          className={`w-full p-3 border rounded-md focus:ring-2 focus:ring-
blue-500 ${
            manifestData.doi && !validateDoi(manifestData.doi)
              ? 'border-red-300 bg-red-50'
              : manifestData.doi && validateDoi(manifestData.doi)
                ? 'border-green-300 bg-green-50'
                : 'border-gray-300'
            }}
        />
        {manifestData.doi && (
          <div className="absolute right-3 top-3">
            {validateDoi(manifestData.doi) ? (
              <CheckCircle className="h-5 w-5 text-green-500" />
            ) : (
              <AlertCircle className="h-5 w-5 text-red-500" />
            )}
          </div>
        )}
      </div>
    </div>
    {manifestData.doi && !validateDoi(manifestData.doi) && (
      <p className="text-red-600 text-sm mt-1">Format DOI invalide</p>
    )}
  </div>

  <div className="grid grid-cols-2 gap-4">
    <div>

```

```

        <label className="block text-sm font-medium text-gray-700 mb-1">
            ID Lettre *
        </label>
        <input
            type="text"
            value={manifestData.letterId}
            onChange={(e) => setManifestData({...manifestData, letterId:
e.target.value})}
            placeholder="SPA_2469"
            className="w-full p-3 border border-gray-300 rounded-md focus:ring-2
focus:ring-blue-500"
        />
    </div>
    <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">
            Date
        </label>
        <input
            type="date"
            value={manifestData.dateIssued}
            onChange={(e) => setManifestData({...manifestData, dateIssued:
e.target.value})}
            className="w-full p-3 border border-gray-300 rounded-md focus:ring-2
focus:ring-blue-500"
        />
    </div>
    </div>

    <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">
            Titre
        </label>
        <input
            type="text"
            value={manifestData.title}
            onChange={(e) => setManifestData({...manifestData, title: e.target.value})}
            className="w-full p-3 border border-gray-300 rounded-md focus:ring-2
focus:ring-blue-500"
        />
    </div>

    <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">
            Créateur
        </label>
        <input
            type="text"
            value={manifestData.creator}
            onChange={(e) => setManifestData({...manifestData, creator:
e.target.value})}

```

```

        className="w-full p-3 border border-gray-300 rounded-md focus:ring-2
focus:ring-blue-500"
      />
    </div>
  </div>

  { /* Pages Section */ }
  <div className="space-y-4">
    <div className="flex items-center justify-between">
      <h3 className="text-lg font-semibold text-gray-800">Pages</h3>
      <div className="flex gap-2">
        <button
          onClick={openAllInfoUrls}
          disabled={!manifestData.doi || !validateDoi(manifestData.doi)}
          className="bg-purple-500 hover:bg-purple-600 disabled:bg-gray-300
text-white px-4 py-2 rounded-md transition-colors text-sm"
        >
          <ExternalLink className="h-4 w-4 inline mr-1" />
          Ouvrir URLs info.json
        </button>
        <button
          onClick={addPage}
          className="bg-green-500 hover:bg-green-600 text-white p-2 rounded-
md transition-colors"
        >
          <Plus className="h-4 w-4" />
        </button>
      </div>
    </div>

    {manifestData.pages.map((page, index) => (
      <div key={index} className="p-4 bg-gray-50 rounded-md">
        <div className="flex items-center justify-between mb-3">
          <h4 className="font-medium flex items-center gap-2">
            Page {index + 1}
            {page.validated && <CheckCircle className="h-4 w-4 text-green-500" /
>}
          </h4>
          {manifestData.pages.length > 1 && (
            <button
              onClick={() => removePage(index)}
              className="text-red-500 hover:text-red-700 p-1"
            >
              <Minus className="h-4 w-4" />
            </button>
          )}
        </div>

        <div className="space-y-3">
          <div>

```

```

<label className="block text-sm font-medium text-gray-700 mb-1">
  Hash de l'image *
</label>
<div className="relative">
  <input
    type="text"
    value={page.hash}
    onChange={(e) => updatePage(index, 'hash', e.target.value)}
    placeholder="e69e975a11ecbdbf8ca3c64556c9f626ea183393"
    className={`w-full p-2 border rounded-md focus:ring-2 focus:ring-
blue-500 ${
      page.hash && !validateHash(page.hash)
        ? 'border-red-300 bg-red-50'
        : page.hash && validateHash(page.hash)
        ? 'border-green-300 bg-green-50'
        : 'border-gray-300'
      }}
  />
  <div className="absolute right-2 top-2 flex gap-1">
    {page.hash && validateHash(page.hash) && manifestData.doi &&
validateDoi(manifestData.doi) && (
      <button
        onClick={() => openInfoUrl(manifestData.doi, page.hash)}
        className="text-blue-500 hover:text-blue-700 p-1"
        title="Ouvrir l'URL info.json"
      >
        <ExternalLink className="h-4 w-4" />
      </button>
    )}
    {page.hash && (
      validateHash(page.hash) ? (
        <CheckCircle className="h-4 w-4 text-green-500" />
      ) : (
        <AlertCircle className="h-4 w-4 text-red-500" />
      )
    )}
  </div>
</div>
{page.error && (
  <p className="text-red-600 text-sm mt-1">{page.error}</p>
)}
{page.hash && !validateHash(page.hash) && (
  <p className="text-red-600 text-sm mt-1">Format hash invalide (40
caractères hexadécimaux)</p>
)}
</div>

{/* Nouveau champ pour saisie assistée des dimensions */}
<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">

```

```

        Coller les dimensions ici (format libre)
    </label>
    <div className="flex gap-2">
        <input
            type="text"
            value={page.dimensionsInput || ''}
            onChange={(e) => updatePage(index, 'dimensionsInput',
e.target.value)}
            placeholder='Ex: {"width":2692,"height":3430} ou 2692x3430 ou
2692,3430'
            className="flex-1 p-2 border border-gray-300 rounded-md
focus:ring-2 focus:ring-blue-500 text-sm"
        />
        <button
            onClick={() => parseDimensions(page.dimensionsInput, index)}
            disabled={!page.dimensionsInput}
            className="bg-blue-500 hover:bg-blue-600 disabled:bg-gray-300
text-white px-3 py-2 rounded-md text-sm"
        >
            Appliquer
        </button>
    </div>
    <p className="text-xs text-gray-500 mt-1">
        Copiez tout ou partie du JSON depuis l'onglet info.json ouvert, ou
saisissez juste "largeur x hauteur"
    </p>
</div>

<div className="grid grid-cols-2 gap-3">
    <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">
            Largeur
        </label>
        <input
            type="number"
            value={page.width}
            onChange={(e) => updatePage(index, 'width',
parseInt(e.target.value))}
            className="w-full p-2 border border-gray-300 rounded-md
focus:ring-2 focus:ring-blue-500"
        />
    </div>
    <div>
        <label className="block text-sm font-medium text-gray-700 mb-1">
            Hauteur
        </label>
        <input
            type="number"
            value={page.height}
            onChange={(e) => updatePage(index, 'height',

```

```

parseInt(e.target.value)))
      className="w-full p-2 border border-gray-300 rounded-md
focus:ring-2 focus:ring-blue-500"
    />
  </div>
</div>
</div>
</div>
))}
</div>

```

```

{/* Generate Button */}
<button
  onClick={generateManifest}
  disabled={!isFormValid() || isGenerating}
  className="w-full bg-blue-600 hover:bg-blue-700 disabled:bg-gray-300 text-
white font-semibold py-3 px-6 rounded-lg transition-colors"
>
  {isGenerating ? (
    <div className="inline-block mr-2 h-5 w-5 border-2 border-white border-t-
transparent rounded-full animate-spin"></div>
    Génération en cours...
  </>
  ) : (
    <FileText className="inline-block mr-2 h-5 w-5" />
    Générer le Manifest IIIF
  </>
  )}
</button>
</div>

```

```

{/* Preview Section */}
<div className="space-y-4">
  <h3 className="text-lg font-semibold text-gray-800">Aperçu du Manifest</
h3>

```

```

{generatedManifest ? (
  <div className="space-y-4">
    <div className="bg-gray-100 p-4 rounded-md">
      <pre className="text-sm overflow-x-auto max-h-96 text-gray-800">
        {generatedManifest}
      </pre>
    </div>

    <button
      onClick={downloadManifest}
      className="w-full bg-green-600 hover:bg-green-700 text-white font-
semibold py-3 px-6 rounded-lg transition-colors"

```

```
>
  <Download className="inline-block mr-2 h-5 w-5" />
  Télécharger le Manifest
</button>
</div>
) : (
  <div className="bg-gray-50 p-8 rounded-md text-center text-gray-500">
    Le manifest généré apparaîtra ici...
  </div>
  })
</div>
</div>
</div>
);
};
```