REPORT TECNICO

Banca Dati Complessa "PSR Regionali" — Piani di Sviluppo Rurale 2014-2020

Autore e responsabile: Mario Cariello
Versione documento: 3.0 (estesa)
Data emissione: dicembre 2025
Stato: Documento tecnico ufficiale

---

1. Premessa e inquadramento normativo

1.1 Il contesto europeo: la PAC 2014-2020

La Politica Agricola Comune (PAC) dell'Unione Europea per il periodo 2014-2020 è stata disciplinata da un quadro normativo complesso, articolato su quattro regolamenti di base:

- **Regolamento (UE) n. 1303/2013** — disposizioni comuni sui Fondi Strutturali e di Investimento Europei (fondi SIE)
- **Regolamento (UE) n. 1305/2013** — sostegno allo sviluppo rurale da parte del FEASR
- **Regolamento (UE) n. 1306/2013** — finanziamento, gestione e monitoraggio della PAC
- **Regolamento (UE) n. 1307/2013** — pagamenti diretti agli agricoltori

Il **secondo pilastro** della PAC, dedicato allo sviluppo rurale, è stato finanziato dal **FEASR (Fondo Europeo Agricolo per lo Sviluppo Rurale)** e attuato attraverso i **Programmi di Sviluppo Rurale (PSR)** nazionali e regionali.

1.2 I PSR come strumento di programmazione

Ogni Regione italiana ha adottato un proprio PSR, articolato in:

- **Priorità dell'Unione** (6 priorità, dalla 1 alla 6)
- **Focus area** (aree di intervento specifiche)
- **Misure** (singoli interventi finanziati, dalla misura 1 alla misura 19)
- **Sottomisure e operazioni** (declinazioni operative delle misure)
- **Dotazioni finanziarie** (ripartizione del budget FEASR + cofinanziamento nazionale)

Ogni PSR è stato **negoziato con la Commissione Europea** e approvato con una **Decisione di esecuzione**. Successivamente, ogni modifica sostanziale (nuove misure, rimodulazioni finanziarie, correzioni) ha richiesto una **nuova versione** del programma, approvata con un'ulteriore Decisione.

1.3 La complessità del ciclo di vita dei PSR

Il ciclo di vita di un PSR 2014-2020 è stato particolarmente complesso:

1. **Fase di programmazione** (2013-2014): stesura del programma, analisi SWOT, partenariato
2. **Fase di negoziazione** (2014-2015): confronto con la Commissione, osservazioni, modifiche

3. **Fase di approvazione** (2015): Decisione di esecuzione della Commissione
4. **Fase di attuazione** (2015-2023): gestione delle misure, bandi, pagamenti
5. **Fase di modifica continua**: ogni variazione richiedeva una nuova versione approvata

La banca dati "PSR Regionali" traccia **tutte le versioni** pubblicate dalle Regioni, consentendo di ricostruire l'intero percorso evolutivo di ciascun programma.

---

2. Architettura generale del sistema

2.1 Modello dati

Il cuore del sistema è un **indice strutturato in JavaScript** (oggetto PSR_INDEX) organizzato per **codici regionali normalizzati**. La scelta di JavaScript come formato dati è motivata da:

- **Compatibilità universale** — nessuna dipendenza da database esterni
- **Velocità di accesso** — i dati sono caricati direttamente nel browser
- **Semplicità di aggiornamento** — basta modificare il file HTML
- **Portabilità** — il file può essere ospitato su qualsiasi server statico

2.2 Struttura dei record

Ogni record della banca dati contiene i seguenti campi:

Campo	Tipo	Descrizione
`version`	number/string	Numero di versione (es. 2.3, 14.1)
`date`	string (ISO 8601)	Data di pubblicazione (YYYY-MM-DD)
`url`	string	Collegamento diretto al PDF
`title`	string	Denominazione del piano
`period`	string	Periodo di programmazione (2014-2020)

2.3 Volume di dati

La banca dati complessa gestisce attualmente **oltre 300 record documentali**, distribuiti su 21 entità territoriali. Il numero di versioni per regione è il seguente:

Regione/PA	Codice	Numero versioni	Prima versione	Ultima versione
Abruzzo	ABR	14	2016-11-11	2025-11-11
Basilicata	BAS	17	2016-05-10	2025-11-07
Calabria	CAL	16	2017-05-29	2025-12-15
Campania	CAM	17	2017-02-27	2025-10-21
Emilia-Romagna	EMR	17	2015-12-21	2025-12-17
Friuli V.G.	FVG	18	2016-11-14	2025-11-24
Lazio	LAZ	19	2016-12-23	2025-12-11
Liguria	LIG	18	2017-02-13	2025-11-25
Lombardia	LOM	16	2016-06-09	2025-11-06
Marche	MAR	15	2017-02-16	2025-12-05
Molise	MOL	12	2016-08-23	2023-11-20
Piemonte	PIE	18	2017-02-27	2025-09-18

P.A. Bolzano	PABZ	14	2016-01-27	2024-02-28
P.A. Trento	PATN	15	2017-02-08	2025-10-20
Puglia	PUG	18	2017-01-27	2025-11-11
Sardegna	SAR	13	2016-12-12	2025-11-19
Sicilia	SIC	16	2016-12-22	2025-11-27
Toscana	TOS	19	2017-02-27	2025-11-28
Umbria	UMB	16	2017-01-04	2025-12-11
Valle d'Aosta	VDA	19	2017-02-21	2025-09-18
Veneto	VEN	18	2016-02-19	2025-12-11

Totale: oltre 300 record.

Questa **disomogeneità** (da 12 a 19 versioni per regione) riflette:

- **Diversi ritmi di attuazione** — alcune Regioni hanno modificato i programmi più frequentemente
- **Complessità amministrativa** — Regioni con più misure e più fondi hanno richiesto più aggiustamenti
- **Eventi contingenti** — emergenze (es. COVID-19) hanno richiesto modifiche straordinarie

---

3. Componenti tecnologiche

3.1 Front-end

3.1.1 HTML5

La struttura della pagina è costruita con **HTML5 semantico**:

- `<header>` — intestazione con titolo e sottotitolo
- `<main>` — contenuto principale
- `<div class="panel">` — contenitori per mappa, menu e risultati
- `<select>` — menu a tendina per la selezione della regione
- `<input type="search">` — campo di ricerca
- `<ul class="list">` — lista dei risultati
- `<footer>` — piè di pagina

Vengono utilizzati attributi ARIA per l'accessibilità:

- `aria-live="polite"` — annuncia automaticamente i cambiamenti della lista risultati agli screen reader
- `role` e `aria-label` — etichette esplicite per i controlli

3.1.2 CSS3

Il sistema di design è completamente custom:

- **Palette colori**: blu istituzionale `#1e3a8a`, blu intermedio `#2563eb`, grigi `#4b5563`, `#6b7280`, `#9ca3af`
- **Tipografia**: "Inter", system-ui, -apple-system, Segoe UI, Roboto, Arial
- **Layout**: flexbox e grid per il responsive design
- **Componenti**: badge, bottoni, card, tooltip, legenda

Il CSS include **transizioni fluide** (`transition: background 0.2s ease`) per migliorare l'esperienza utente.

3.1.3 JavaScript (ES6+)

La logica applicativa è scritta in **JavaScript moderno**:

- `const` e `let` per la dichiarazione delle variabili
- Arrow functions per le funzioni anonime
- Template literals per la generazione dell'HTML dinamico
- `async/await` per le operazioni asincrone (download ZIP)
- `Object.entries()` per l'iterazione sugli oggetti
- `Array.prototype.sort()`, `filter()`, `map()` per la manipolazione dei dati

3.2 Librerie esterne — Analisi approfondita

La banca dati complessa "PSR Regionali" si avvale di **quattro librerie JavaScript esterne**, ciascuna selezionata per la sua affidabilità, performance e diffusione nel panorama dello sviluppo web professionale. Di seguito un'analisi dettagliata di ciascuna.

3.2.1 JSZip (versione 3.10.1)

Fornitore: CDNJS (cdnjs.cloudflare.com) **Licenza:** MIT (permissiva, consente uso commerciale e modifiche) **Peso:** ~100 KB (minificato)

Funzione principale: generazione di archivi ZIP lato client, senza necessità di un server intermedio.

Dettagli tecnici:

- Consente di creare archivi compressi direttamente nel browser
- Supporta la compressione DEFLATE (il metodo standard per i file ZIP)
- Gestisce cartelle, file annidati e metadati
- API asincrona (`generateAsync()`) che non blocca il thread principale
- Compatibile con tutti i browser moderni (Chrome, Firefox, Safari, Edge)

Utilizzo nella banca dati:

- Creazione di un oggetto JSZip per ogni download
- Creazione di una cartella con nome PSR_{Regione} (spazi sostituiti con underscore)
- Aggiunta di ogni PDF scaricato come file binario (folder.file(fileName, blob))

- Aggiunta del file README.txt con l'elenco completo dei documenti
- Generazione dell'archivio con zip.generateAsync({ type: "blob" })
- Avvio del download tramite un link temporaneo (URL.createObjectURL)

Perché è stata scelta:

- **Nessuna dipendenza server** — la compressione avviene interamente nel browser, riducendo il carico sul server
- **Standard de facto** — è la libreria più utilizzata per la manipolazione ZIP in JavaScript
- **Manutenzione attiva** — aggiornamenti regolari e community ampia

3.2.2 SheetJS (xlsx.full.min.js, versione 0.18.5)

Fornitore: CDNJS (cdnjs.cloudflare.com) **Licenza:** Apache 2.0 (permissiva) **Peso:** ~800 KB (minificato)

Funzione principale: lettura, manipolazione e scrittura di fogli di calcolo in formato Excel (XLSX) direttamente nel browser.

Dettagli tecnici:

- Supporta i formati XLSX, XLS, CSV, ODS e altri
- Consente la creazione di fogli di lavoro da array di oggetti JavaScript
- Gestisce stili, formule, grafici e tabelle pivot
- API sincrona e asincrona
- Compatibile con browser e Node.js

Utilizzo nella banca dati:

- Conversione dei dati piatti (flattenPSRData()) in un foglio di lavoro
- Creazione di un nuovo workbook (XLSX.utils.book_new())
- Aggiunta del foglio al workbook (XLSX.utils.book_append_sheet())
- Scrittura del file Excel (XLSX.writeFile())
- Download automatico con nome PSR_OpenData_{data}.xlsx

Perché è stata scelta:

- **Standard industriale** — è la libreria di riferimento per Excel in JavaScript
- **Formati multipli** — consente di esportare anche in CSV e JSON con la stessa libreria
- **Affidabilità** — utilizzata da migliaia di applicazioni enterprise

3.2.3 Leaflet (versione 1.x)

Fornitore: caricata dal server istituzionale (`/downloads/leaflet.js`) **Licenza:** BSD-2-Clause (permissiva) **Peso:** ~145 KB (minificato)

Funzione principale: rendering di mappe interattive georeferenziate nel browser.

Dettagli tecnici:

- Basata su HTML5 Canvas e SVG
- Supporta GeoJSON, KML, GPX e altri formati geografici
- Gestisce layer, marker, poligoni, tooltip e popup
- Ottimizzata per performance su dispositivi mobili
- Plugin ecosystem molto ampio

Utilizzo nella banca dati:

- Inizializzazione della mappa con centro su Italia ([42.5, 12.5], zoom 5)
- Caricamento dei GeoJSON (regioni.geojson e province.geojson)
- Creazione di layer geografici con L.geoJSON()
- Applicazione di stili personalizzati (colori, spessori, opacità)
- Gestione degli eventi di click e hover
- Binding dei tooltip con layer.bindTooltip()
- Adattamento automatico della vista con map.fitBounds()

Perché è stata scelta:

- **Leggerezza** — molto più leggera di alternative come OpenLayers o Mapbox GL
- **Semplicità** — API intuitiva e ben documentata
- **Open source** — licenza BSD, nessun costo di licenza
- **Affidabilità** — utilizzata da NASA, GitHub, e migliaia di siti istituzionali

3.2.4 Leaflet CSS (versione 1.x)

Fornitore: caricata dal server istituzionale (`/downloads/leaflet.css`) **Licenza:** BSD-2-Clause (permissiva)

Funzione principale: stile e controlli della mappa Leaflet.

Dettagli tecnici:

- Definisce lo stile dei controlli (zoom, attribuzione, scale)
- Gestisce il layout dei tooltip e dei popup
- Include i marker e le icone di default
- Compatibile con il sistema di design custom della banca dati

Utilizzo nella banca dati:

- Stile dei controlli di zoom (posizione, dimensione, colore)
- Stile dei tooltip (sfondo, bordo, font)

- Stile dei popup (se utilizzati)
- Integrazione con il CSS custom (#map.leaflet-container { font-family: "Inter", system-ui, sans-serif; })

Perché è stata scelta:

- **Coerenza visiva** — garantisce che la mappa si integri perfettamente con il design della pagina
- **Performance** — CSS leggero e ottimizzato
- **Compatibilità** — funziona con tutte le versioni di Leaflet

3.3 Dati geografici

La mappa di calore si basa su **GeoJSON** caricati dal server:

- `regioni.geojson` — confini amministrativi delle Regioni
- `province.geojson` — confini delle Province (usati per le Province Autonome di Trento e Bolzano)

Fonte dei dati geografici: repository open-source **openpolis/geojson-italy** (licenza CC-BY 4.0), basato sui limiti amministrativi ufficiali **ISTAT**.

---

4. Funzionalità implementate

4.1 Mappa di calore interattiva cliccabile

4.1.1 Logica di colorazione

La mappa rappresenta **cromaticamente** il numero di versioni PSR pubblicate per ciascuna Regione/Provincia Autonoma. La funzione `getColor(n)` definisce una **scala cromatica progressiva**:

- Numero versioni: 1 — Colore: Azzurro chiaro — Codice esadecimale: `#dbe4ff`
- Numero versioni: 2 — Colore: Azzurro medio — Codice esadecimale: `#bac8ff`
- Numero versioni: 3 — Colore: Azzurro intenso — Codice esadecimale: `#91a7ff`
- Numero versioni: 4 — Colore: Blu medio — Codice esadecimale: `#5c7cfa`
- Numero versioni: 5 — Colore: Blu intenso — Codice esadecimale: `#3b5bdb`
- Numero versioni: 6+ — Colore: Blu scuro — Codice esadecimale: `#1e3a8a`

La scala è stata scelta per garantire:

- **Accessibilità** — contrasto sufficiente anche per utenti con daltonismo
- **Intuitività** — più versioni = colore più scuro
- **Coerenza** — stessa palette della banca dati CSR

4.1.2 Mappatura ISTAT

La funzione `getIstatCode(feature)` estrae il codice ISTAT da ogni feature del GeoJSON, cercando in ordine di priorità:

- prov_istat_code
- prov_istat_code_num
- prov_uts_code
- reg_istat_code
- reg_istat_code_num
- cod_reg
- cod_prov

Il codice viene normalizzato a 3 cifre con `String(raw).padStart(3, "0")`.

La tabella `ISTAT_TO_CODES` mappa i codici ISTAT alle entità della banca dati:

- Codice ISTAT: 001 — Codice banca dati: PIE — Entità: Piemonte
- Codice ISTAT: 002 — Codice banca dati: VDA — Entità: Valle d'Aosta
- Codice ISTAT: 003 — Codice banca dati: LOM — Entità: Lombardia
- Codice ISTAT: 004 — Codice banca dati: — — Entità: (Trentino-Alto Adige, escluso)
- Codice ISTAT: 005 — Codice banca dati: VEN — Entità: Veneto
- Codice ISTAT: 006 — Codice banca dati: FVG — Entità: Friuli Venezia Giulia
- Codice ISTAT: 007 — Codice banca dati: LIG — Entità: Liguria
- Codice ISTAT: 008 — Codice banca dati: EMR — Entità: Emilia-Romagna
- Codice ISTAT: 009 — Codice banca dati: TOS — Entità: Toscana
- Codice ISTAT: 010 — Codice banca dati: UMB — Entità: Umbria
- Codice ISTAT: 011 — Codice banca dati: MAR — Entità: Marche
- Codice ISTAT: 012 — Codice banca dati: LAZ — Entità: Lazio
- Codice ISTAT: 013 — Codice banca dati: ABR — Entità: Abruzzo
- Codice ISTAT: 014 — Codice banca dati: MOL — Entità: Molise
- Codice ISTAT: 015 — Codice banca dati: CAM — Entità: Campania
- Codice ISTAT: 016 — Codice banca dati: PUG — Entità: Puglia
- Codice ISTAT: 017 — Codice banca dati: BAS — Entità: Basilicata
- Codice ISTAT: 018 — Codice banca dati: CAL — Entità: Calabria
- Codice ISTAT: 019 — Codice banca dati: SIC — Entità: Sicilia
- Codice ISTAT: 020 — Codice banca dati: SAR — Entità: Sardegna
- Codice ISTAT: 021 — Codice banca dati: PABZ — Entità: P.A. Bolzano

- Codice ISTAT: 022 — Codice banca dati: PATN — Entità: P.A. Trento

4.1.3 Gestione delle Province Autonome

Il Trentino-Alto Adige (codice ISTAT 004) viene **escluso** come regione, e le sue due Province Autonome vengono rappresentate **separatamente**:

- **P.A. di Bolzano** (ISTAT 021) — dal file `province.geojson`
- **P.A. di Trento** (ISTAT 022) — dal file `province.geojson`

Questa scelta è motivata dalla **autonomia amministrativa** delle due Province, che gestiscono autonomamente i propri PSR.

4.1.4 Interazioni disponibili

Hover:

- Tooltip sticky con nome entità, numero versioni, data ultima modifica
- Opacità ridotta al passaggio del mouse

Click:

- Selezione della regione
- Sincronizzazione automatica col menu a tendina (`sel.value = codes[0]`)
- Evento `change` sul menu per aggiornare la lista risultati
- Evidenziazione del confine selezionato (bordo blu `#1e3a8a`, spessore 2.5px)

Legenda dinamica:

- Generata automaticamente in base alla scala cromatica
- Ogni voce include un quadratino colorato (`legend-swatch`) e l'etichetta testuale

4.2 Motore di ricerca

Il campo di ricerca consente filtri testuali su:

- **Anno** — es. "2020" → trova tutte le versioni pubblicate nel 2020
- **Numero di versione** — es. "7.1" → trova la versione 7.1
- **Parola chiave nel titolo** — es. "Sicilia" → trova i PSR della Sicilia
- **Data completa** — es. "2021-10-14" → trova la versione pubblicata in quella data

La ricerca è **case-insensitive** (tutto convertito in minuscolo) e aggiorna la lista in tempo reale ad ogni input.

4.3 Ordinamento cronologico

Le versioni sono presentate in **ordine decrescente di data** (dalla più recente alla più vecchia):

```
const items = (PSR_INDEX[code] || []).slice().sort((a, b) => new Date(b.date) - new Date(a.date));
```

Questo garantisce un accesso immediato all'ultima versione approvata.

4.4 Download ZIP con README — Analisi approfondita

La funzione `downloadAllPdfs()` è una delle **funzionalità più complesse e utili** della banca dati. Consente all'utente di scaricare **tutte le versioni PSR di una Regione in un unico archivio ZIP**, corredato da un file README che documenta l'intero contenuto.

4.4.1 Flusso operativo dettagliato

Fase 1 — Verifica dei dati:

```
const files = PSR_INDEX[regionCode];
if (!files ||!files.length) return alert("Nessun documento disponibile per questa Regione.");
```

- Controlla che la Regione selezionata abbia almeno un documento
- In caso negativo, mostra un alert all'utente

Fase 2 — Creazione dell'archivio ZIP:

```
const zip = new JSZip();
const folder = zip.folder(`PSR_${regionName.replace(/\s+/g, "_")}`);
```

- Istanzia un nuovo oggetto JSZip
- Crea una cartella interna con nome `PSR_{Regione}` (spazi → underscore)

Fase 3 — Inizializzazione del README:

```
let readme = `Elenco versioni PSR ${regionName}\n----------------------------------\n`;
```

- Il README inizia con un'intestazione che identifica la Regione

Fase 4 — Aggiornamento dello stato del pulsante:

```
const btn = document.querySelector(".btn-zip");
btn.textContent = "Download in corso...";
```

- Il pulsante diventa "Download in corso..." per informare l'utente

Fase 5 — Download e aggiunta dei PDF:

```
for (const file of files) {
 const fixedUrl = ensureHttps(file.url);
 try {
 const response = await fetch(fixedUrl);
 const blob = await response.blob();
 const fileName = `v${fmtVersion(file.version)}_${regionName}.pdf`;
 folder.file(fileName, blob);
 readme += `Versione ${fmtVersion(file.version)} - Data ${fmtDate(file.date)}\n${fixedUrl}\n\n`;
 } catch (err) {
 readme += `⚠ Errore download: ${fixedUrl}\n`;
 }
}
```

- Itera su tutte le versioni della Regione
- Normalizza l'URL con `ensureHttps()`
- Scarica il PDF con `fetch()` e lo converte in blob
- Aggiunge il file alla cartella ZIP con nome `v{versione}_{Regione}.pdf`
- Aggiunge la riga corrispondente al README
- In caso di errore, registra l'errore nel README senza interrompere il processo

Fase 6 — Aggiunta del README all'archivio:

```
folder.file("README.txt", readme);
```

Fase 7 — Generazione dell'archivio ZIP:

```
const content = await zip.generateAsync({ type: "blob" });
```

- Genera l'archivio compresso in formato blob

Fase 8 — Avvio del download:

```
const link = document.createElement("a");
link.href = URL.createObjectURL(content);
link.download = `PSR_${regionName}.zip`;
link.click();
```

- Crea un link temporaneo
- Imposta l'URL del blob
- Imposta il nome del file (`PSR_{Regione}.zip`)
- Simula il click per avviare il download

Fase 9 — Ripristino dello stato del pulsante:

```
btn.textContent = "📦 Scarica tutte le versioni in ZIP";
```

4.4.2 Esempio di README generato

```
Elenco versioni PSR Abruzzo
-----------------------------------
Versione 15.1 - Data 11/11/2025
https://www.reterurale.it/downloads/PSR_Abruzzo_15_1.pdf

Versione 14.1 - Data 26/03/2025
https://www.reterurale.it/downloads/PSR_Abruzzo_14_1.pdf

Versione 13.0 - Data 26/11/2024
https://www.reterurale.it/downloads/PSR_Abruzzo_13_0.pdf

...
```

4.4.3 Vantaggi per l'utente

- **Risparmio di tempo** — scarica tutte le versioni in un unico click
- **Organizzazione** — file rinominati in modo coerente (`v15.1_Abruzzo.pdf`)
- **Documentazione** — README con elenco completo e link
- **Affidabilità** — gestione degli errori senza interrompere il processo
- **Portabilità** — archivio ZIP compatibile con tutti i sistemi operativi

4.5 Copia link negli appunti — Analisi approfondita

La funzione `copyAllLinks()` consente di **copiare l'elenco completo dei link** delle versioni PSR di una Regione in un unico testo formattato, pronto per essere incollato in documenti, email o fogli di calcolo.

4.5.1 Flusso operativo dettagliato

Fase 1 — Verifica dei dati:

```
const files = PSR_INDEX[regionCode];
if (!files ||!files.length) return alert("Nessun documento disponibile per questa Regione.");
```

Fase 2 — Generazione del testo formattato:

```
let text = `Elenco link PSR ${regionName}\n----------------------------------\n`;
files.forEach(f => {
 const fixedUrl = ensureHttps(f.url);
 text += `Versione ${fmtVersion(f.version)} (${fmtDate(f.date)}) → ${fixedUrl}\n`;
});
```

- Intestazione con nome della Regione
- Per ogni versione: numero, data, URL normalizzato

Fase 3 — Copia nella clipboard:

```
navigator.clipboard.writeText(text)
.then(() => alert("☑ Elenco link copiato negli appunti"))
.catch(() => alert("⚠ Errore nella copia dei link"));
```

- Utilizza l'API `navigator.clipboard` (standard moderno)
- Gestisce sia il successo che l'errore

4.5.2 Esempio di testo copiato

```
Elenco link PSR Abruzzo
----------------------------------
Versione 15.1 (11/11/2025) → https://www.reterurale.it/downloads/PSR_Abruzzo_15_1.pdf
Versione 14.1 (26/03/2025) → https://www.reterurale.it/downloads/PSR_Abruzzo_14_1.pdf
Versione 13.0 (26/11/2024) → https://www.reterurale.it/downloads/PSR_Abruzzo_13_0.pdf
Versione 12.2 (07/03/2024) → https://www.reterurale.it/downloads/PSR_Abruzzo_12_2.pdf
```

```
Versione 11.1 (01/08/2023) → https://www.reterurale.it/downloads/PSR_Abruzzo_11_1.pdf
Versione 10.1 (16/09/2022) → https://www.reterurale.it/downloads/PSR_Abruzzo_10_1.pdf
Versione 9.1 (14/10/2021) → https://www.reterurale.it/downloads/PSR_Abruzzo_9_1.pdf
Versione 8.0 (24/12/2020) → https://www.reterurale.it/downloads/PSR_Abruzzo_8_0.pdf
Versione 7.1 (29/10/2020) → https://www.reterurale.it/downloads/PSR_Abruzzo_7_1.pdf
Versione 6.0 (30/01/2020) → https://www.reterurale.it/downloads/PSR_Abruzzo_6_0.pdf
Versione 5.1 (18/12/2018) → https://www.reterurale.it/downloads/PSR_Abruzzo_5_1.pdf
Versione 4.1 (01/03/2018) → https://www.reterurale.it/downloads/PSR_Abruzzo_4_1.pdf
Versione 3.3 (05/12/2017) → https://www.reterurale.it/downloads/PSR_Abruzzo_3_3.pdf
Versione 2.3 (11/11/2016) → https://www.reterurale.it/downloads/PSR_Abruzzo_2_3.pdf
```

4.5.3 Vantaggi per l'utente

- **Velocità** — copia tutti i link in un unico click
- **Formato uniforme** — testo ordinato e leggibile
- **Compatibilità** — può essere incollato in qualsiasi applicazione
- **Tracciabilità** — ogni link è associato alla versione e alla data

4.6 Esportazione Open Data — La banca dati è completamente Open Data

La banca dati complessa "PSR Regionali" è **completamente open data**, nel pieno rispetto dei principi di trasparenza, riusabilità e interoperabilità promossi dalla normativa europea e italiana (direttiva PSI, CAD — Codice dell'Amministrazione Digitale).

4.6.1 Principi open data applicati

- Accessibilità totale — tutti i dati sono liberamente accessibili senza autenticazione, registrazione o pagamento
- Formati aperti — i dati sono disponibili in formati standard e non proprietari (JSON, CSV, XLSX)
- Riusabilità — i dati possono essere utilizzati, modificati e ridistribuiti da chiunque, anche per scopi commerciali
- Granularità — i dati sono disponibili a livello di singolo record (versione PSR per Regione)
- Aggiornamento — i dati vengono aggiornati regolarmente man mano che le Regioni pubblicano nuove versioni

4.6.2 Pannello "Scarica database PSR (Open Data)"

Il pannello dedicato consente di esportare l'intero archivio in **tre formati aperti**:

JSON (JavaScript Object Notation):

- Formato standard per lo scambio di dati strutturati
- Leggibile da qualsiasi linguaggio di programmazione
- Struttura gerarchica che preserva le relazioni tra i dati
- File di esempio: `PSR_OpenData_2026-09-01.json`

CSV (Comma Separated Values):

- Formato tabellare universale
- Compatibile con Excel, LibreOffice, Google Sheets
- Importabile in qualsiasi database relazionale
- File di esempio: `PSR_OpenData_2026-09-01.csv`

XLSX (Excel Open XML):

- Formato standard per fogli di calcolo
- Include intestazioni di colonna e formattazione
- Compatibile con Microsoft Excel e alternative open source
- File di esempio: `PSR_OpenData_2026-09-01.xlsx`

4.6.3 Struttura dei dati esportati

Ogni file esportato include i seguenti campi per ogni record:

- Campo: `codice` — Descrizione: Codice regionale normalizzato (es. ABR)
- Campo: `regione` — Descrizione: Nome completo della Regione/PA
- Campo: `versione` — Descrizione: Numero di versione (es. 15.1)
- Campo: `data` — Descrizione: Data di pubblicazione (formato italiano DD/MM/YYYY)
- Campo: `titolo` — Descrizione: Denominazione del piano (es. PSR Abruzzo)
- Campo: `periodo` — Descrizione: Periodo di programmazione (2014-2020)
- Campo: `url` — Descrizione: Collegamento diretto al PDF

4.6.4 Vantaggi dell'approccio open data

- **Trasparenza amministrativa** — i cittadini possono verificare l'uso dei fondi europei
- **Ricerca scientifica** — i ricercatori possono analizzare i dati con strumenti statistici
- **Innovazione** — gli sviluppatori possono creare applicazioni basate sui dati
- **Interoperabilità** — i dati possono essere integrati con altre banche dati (es. CSR)
- **Riuso** — i dati possono essere combinati con altre fonti per analisi avanzate

---

5. Gestione degli errori e robustezza

5.1 Normalizzazione URL

La funzione `ensureHttps()`:

```
function ensureHttps(url) {
 if (!url) return "";
 const u = url.trim();
 return u.match(/^https?:\/\//i)? u: "https://" + u.replace(/^\/+/, "");
```

```
}
```

- Aggiunge `https://` se mancante
- Rimuove gli slash iniziali
- Gestisce URL vuoti

5.2 Gestione fetch fallita

Durante il download ZIP, gli errori di rete vengono gestiti con `try/catch`:

```
try {
 const response = await fetch(fixedUrl);
 const blob = await response.blob();
 folder.file(fileName, blob);
} catch (err) {
 readme += `⚠ Errore download: ${fixedUrl}\n`;
}
```

5.3 Stati di caricamento

- **Mappa**: "Caricamento confini geografici..." durante il fetch dei GeoJSON
- **ZIP**: "Download in corso..." durante la generazione dell'archivio

5.4 Messaggi di errore

- **Leaflet non caricato**: "Errore: Leaflet non caricato. Verifica che /downloads/leaflet.js sia raggiungibile."
- **GeoJSON non raggiungibili**: "Errore nel caricamento dei confini. Verifica la connessione internet."
- **Nessun documento**: "Nessun documento disponibile per questa Regione."
- **Nessun risultato ricerca**: "Nessun documento trovato per la selezione corrente."

---

6. Comunicazione con la pagina madre

La banca dati è progettata per essere **integrata in un portale più ampio** tramite `iframe`.
Implementa un sistema di **resize dinamico**:

```
function postHeight() {
 const height = getDocHeight();
 parent.postMessage({ type: 'dbpsr:resize', height }, '*');
}
```

Trigger di invio:

- Evento `load` della finestra
- Ritardi di 200ms e 800ms dopo il load
- Evento `resize` della finestra

- `MutationObserver` sul body (per cambiamenti DOM)
- `ResizeObserver` sul body (per variazioni dimensionali)
- `setInterval` di sicurezza ogni 1.5 secondi

Questo garantisce che il contenitore esterno si adatti sempre alle dimensioni reali del contenuto, evitando scrollbar interne o spazi vuoti.

---

7. Complessità e valore dell'opera

7.1 Fasi di realizzazione

1. **Ricerca documentale approfondita**
   - Individuazione di ogni versione PSR pubblicata dalle 21 Regioni/PA nel periodo 2014-2020
   - Verifica incrociata delle date e dei numeri di versione
   - Controllo della validità degli URL
2. **Normalizzazione dei dati**
   - Gestione di formati eterogenei (versioni con decimali, denominazioni regionali diverse)
   - Uniformazione delle date in formato ISO 8601
   - Normalizzazione degli URL
3. **Integrazione geografica**
   - Mappatura tra codici ISTAT e entità amministrative
   - Gestione delle Province Autonome come entità separate
   - Esclusione del Trentino-Alto Adige come regione unica
4. **Sviluppo front-end complesso**
   - Interazione tra mappa Leaflet, menu a tendina e lista risultati in tempo reale
   - Gestione degli eventi di click, hover e change
   - Rendering dinamico delle liste
5. **Ottimizzazione per l'integrazione**
   - Sistema di comunicazione cross-frame per il corretto dimensionamento nel portale
   - Gestione degli eventi di resize e mutation

7.2 Valore per gli utenti

- **Ricercatori** — analisi dell'evoluzione delle politiche agricole regionali
- **Amministratori pubblici** — confronto tra Regioni, benchmark delle pratiche
- **Tecnici del settore** — accesso rapido alle versioni aggiornate dei programmi
- **Cittadini** — trasparenza sull'uso dei fondi europei

8. Conclusioni

La banca dati complessa "PSR Regionali" rappresenta uno **strumento di consultazione e analisi di elevato valore** per ricercatori, amministratori pubblici, tecnici del settore agricolo e cittadini interessati alle politiche di sviluppo rurale. La sua complessità risiede non solo nella quantità di dati gestiti, ma nella **strutturazione intelligente** che consente di navigare l'evoluzione amministrativa di un intero ciclo di programmazione europea attraverso un'interfaccia intuitiva e potente. La natura **completamente open data** della banca dati ne amplifica il valore, rendendola una risorsa pubblica riutilizzabile da chiunque.