plumeCMS



Manuale dei temi

Come si costruisce un tema da zero, file per file, con un esempio completo.

plumeCMS 2.17.0 · 2 novembre 2026

Indice

1. Cos'è un tema, in due minuti

2. Il percorso, dal foglio bianco al tema attivo

3. Com'è fatta la cartella di un tema

4. `tema.json`: la carta d'identità

5. `stile.json`: colori, caratteri, pulsanti, campi, sfondo

- 5.1 `colori`
- 5.2 `font`
- 5.3 `tipografia`
- 5.4 `pulsanti`
- 5.5 `immagini`
- 5.6 `campi`
- 5.7 `layout`
- 5.8 `sfondo`

6. `modelli.json`: header e footer

- 6.1 Il modo comodo
- 6.2 Il formato dell'albero
- 6.3 Il menu: per posizione, mai per numero
- 6.4 Le impostazioni che servono di più
- 6.5 I colori dentro gli alberi
- 6.6 Attivare di nuovo

7. `tema.css`: il CSS in più

- 7.1 Dove si carica e chi vince
- 7.2 Le variabili CSS
- 7.3 Le classi del sito
- 7.4 Consigli

8. I colori delle infografiche

9. Il PHP del tema: filtri, eventi e pagine ridisegnate

- 9.1 Le regole
- 9.2 Se il tema sbaglia
- 9.3 `funzioni.php`: filtri ed eventi
- 9.4 Le viste: ridisegnare una pagina
- 9.5 Caricare un tema con PHP

10. Provare il tema

- 10.1 Anteprima
- 10.2 La lista dei controlli
- 10.3 Una prova automatica

11. Impacchettare, firmare, distribuire

- 11.1 Lo zip
- 11.2 Caricare
- 11.3 Firmare (solo per i temi con PHP)
- 11.4 Versioni e compatibilità

12. Esempio completo: Lavagna

- `tema.json`
- `stile.json`
- `modelli.json`
- `tema.css`
- `funzioni.php`
- `viste/404.php`
- `anteprima.png`

Le scelte, spiegate

13. Errori comuni e come uscirne

14. Riferimento rapido

Questo manuale è per chi vuole disegnare il proprio tema: un aspetto nuovo per un sito fatto con plumeCMS, da usare sul proprio sito o da regalare e vendere ad altri. Serve saper scrivere un po' di CSS. Il PHP è facoltativo: la maggior parte dei temi non ne ha bisogno.

Alla fine trovi un tema d'esempio completo, **Lavagna**, provato su un plumeCMS vero: i suoi file sono anche nella cartella `esempi/tema-lavagna/` del repository.

1. Cos'è un tema, in due minuti

Un tema cambia **l'aspetto intero** del sito in un colpo solo:

Cambia	Non cambia
colori, caratteri, tipografia di titoli e testi	pagine, articoli, categorie, tag
pulsanti, campi dei moduli, immagini	menu di navigazione (le voci)
sfondo del sito, larghezza del contenuto	immagini caricate
header e footer	nome, sottotitolo, logo e favicon del sito
colori delle infografiche	il CSS personalizzato scritto da chi gestisce il sito
se vuole, il disegno di alcune pagine (con PHP)	impostazioni, utenti, newsletter

Chi gestisce il sito sceglie il tema dal pannello, **Editor > Temi**: lo guarda con **Anteprima** (lo vede solo lui), lo attiva con **Usa questo tema**, e poi ritocca ogni dettaglio da **Stile del sito** e da **Header e footer**. Il tema è il punto di partenza, non una gabbia.

Quando si attiva un tema succedono tre cose:

1. le sezioni dell'aspetto dello Stile del sito ripartono dai valori di partenza del CMS e prendono quelli del tema; lo stile di prima resta nelle **Versioni** (si torna indietro con un clic);
2. l'header e il footer del tema diventano modelli ("Header Nometema", "Footer Nometema") pubblicati e predefiniti; quelli di prima restano in Header e footer;
3. il CMS ricorda quale tema è attivo: da quel momento carica il suo CSS, i colori delle infografiche e il suo PHP.

Il PHP del tema non gira mai al momento dell'attivazione, e mai nel pannello.

2. Il percorso, dal foglio bianco al tema attivo

1. **Prepara un sito di prova.** Un plumeCMS 1.2.0 o successivo, installato dove puoi fare quello che vuoi (va bene anche una sottocartella, per esempio `https://tuosito.it/prova`). Non provare i temi sul sito vero.
2. **Parti da un aspetto vicino a quello che vuoi.** Nel pannello, Editor > Stile del sito: cambia colori e caratteri finché ti piace. Componi header e footer in Editor > Header e footer.
3. **Scaricalo come tema.** Editor > Temi, pulsante **Scarica l'aspetto di adesso come tema**: ottieni uno zip con `tema.json`, `stile.json`, `modelli.json` e, se c'è, `tema.css`, già scritti nel formato giusto. È il modo più veloce e sicuro per partire.
4. **Rifinisci a mano.** Estrai lo zip, cambia nome, descrizione e cartella (capitolo 4), aggiungi il tuo CSS (capitolo 7), i colori delle infografiche (capitolo 8) e, se serve, il PHP (capitolo 9).
5. **Prova.** Ricarica lo zip con **Sostituisci** spuntato, guarda con **Anteprima**, segui la lista del capitolo 10.
6. **Impacchetta e distribuisci** (capitolo 11).

Durante lo sviluppo, chi ha accesso ai file del server (FTP o File Manager di cPanel) può lavorare direttamente nella cartella `temi/<nome>/`: ogni modifica si vede ricaricando l'Anteprima, senza ricaricare lo zip.

3. Com'è fatta la cartella di un tema

Un tema è una cartella dentro `temi/`, nella cartella del CMS sul server:

```
temi/
  lavagna/
    tema.json       obbligatorio: nome, versione, autore, descrizione, nomi degli altri file,
colori delle infografiche
    stile.json      facoltativo: lo Stile del sito del tema
    modelli.json    facoltativo: header e footer
    tema.css        facoltativo: il CSS in più (il nome lo scegli tu in tema.json)
    anteprima.png   facoltativo: l'immagine della scheda nel pannello
    funzioni.php    facoltativo: filtri ed eventi in PHP
    viste/          facoltativo: pagine ridisegnate in PHP (404.php, articolo.php, ...)
```

Il nome della cartella è il nome tecnico del tema: lettere minuscole senza accenti, cifre e trattini, da 2 a 40 caratteri, comincia con una lettera (`lavagna`, `carta-da-zucchero`, `studio2`). Non si può cambiare dopo: è quello che il CMS ricorda come tema attivo.

I nomi dei file dentro la cartella: lettere, cifre, `_`, `-` e `.`, niente spazi, niente file che cominciano con un punto. I percorsi scritti in `tema.json` sono relativi alla cartella del tema e non possono risalire (`..`).

Dal browser nella cartella `temi/` si aprono solo CSS, immagini e caratteri. PHP, JSON, `.htaccess` e `.ini` rispondono 403: li legge solo il CMS. Lo garantisce un `temi/.htaccess` che il CMS scrive da solo.

4. `tema.json`: la carta d'identità

L'unico file obbligatorio. Se manca o non è un JSON valido, il tema non compare nel pannello.

```
{
  "nome": "Lavagna",
  "versione": "1.0",
  "autore": "Mario Rossi",
  "descrizione": "Verde ardesia, testo color gesso, giallo e menta.",
  "slug": "lavagna",
  "anteprima": "anteprima.png",
  "css": "tema.css",
  "stile": "stile.json",
  "modelli": "modelli.json",
  "infografiche": { }
}
```

Campo	Obbligatorio	Regole	Dove si vede
`nome`	consigliato	fino a 60 caratteri; se manca, il nome della cartella	scheda del pannello, nomi dei modelli "Header Nome", messaggi
`versione`	consigliato	fino a 20 caratteri, testo libero (`1.0`, `2.3.1`); se manca, `1.0`	scheda del pannello, nome dello zip scaricato
`autore`	no	fino a 80 caratteri	scheda del pannello
`descrizione`	consigliato	fino a 300 caratteri	scheda del pannello
`slug`	no	come il nome della cartella	serve solo se lo zip ha `tema.json` nella radice, senza cartella: dà il nome alla cartella
`anteprima`	no	png, jpg, jpeg o webp (niente SVG); consigliato 800 per 500 pixel	scheda del pannello; senza, il pannello disegna un'anteprima con i colori di `stile.json`
`css`	no	un file `.css` della cartella	caricato sul sito e nella tela dell'editor
`stile`	no	un file `.json`	applicato all'attivazione e in anteprima
`modelli`	no	un file `.json`	applicato all'attivazione e in anteprima
`infografiche`	no	oggetto, vedi capitolo 8	colori delle infografiche, pallini nella scheda

Se un file scritto in `tema.json` manca o non si legge, il tema compare lo stesso, con l'avviso in rosso sulla scheda ("Il CSS tema.css non c'è.", "Lo stile stile.json non si legge.").

I testi non devono contenere emoji o altri caratteri a 4 byte: il database di molti hosting li rifiuta.

5. `stile.json`: colori, caratteri, pulsanti, campi, sfondo

`stile.json` ha lo stesso formato dello **Stile del sito** del pannello. Il modo più sicuro per scriverlo è farlo scrivere al CMS: imposta lo stile nel pannello e usa **Scarica l'aspetto di adesso come tema**.

Regole generali:

- Contano solo le **sezioni dell'aspetto**: `colori`, `font`, `tipografia`, `pulsanti`, `immagini`, `campi`, `layout`, `sfondo`. Le sezioni `sito` (nome, logo, favicon) e `css` (il CSS personalizzato di chi gestisce il sito) vengono ignorate: sono di chi gestisce il sito, non del tema.
- Ogni sezione **riparte dai valori di partenza** del CMS e prende i valori del tema: scrivi solo quello che vuoi cambiare. Un campo vuoto (`""`) vuol dire "come il foglio di stile del sito".
- Tutto passa dalla stessa pulizia del pannello: un valore non valido torna vuoto (o al valore di partenza, per i colori di sistema), senza errori.

5.1 `colori`

```
"colori": {
  "sistema": { "primario": "#f2c94c", "secondario": "#7fb7a4", "testo": "#f1efe6", "accento":
"#e4785b", "sfondo": "#1f2a26" },
  "personalizzati": [
    { "id": "sfondo-2", "nome": "Sfondo secondario", "valore": "#26332e" },
    { "id": "testo-2", "nome": "Testo attenuato", "valore": "#c9c6b8" },
    { "id": "testo-3", "nome": "Testo discreto", "valore": "#a8a597" },
    { "id": "bordo", "nome": "Bordi", "valore": "rgba(241,239,230,.16)" }
  ]
}
```

Colore	Cosa colora sul sito
`primario`	pulsante principale (`.btn-viola`) e la sua sfumatura, parte del gradiente dei titoli, bagliori, riquadri evidenziati, indicatori dei widget
`secondario`	pulsanti secondari (`.btn-ciano`, `.btn-filo`), occhielli, link nei riquadri, icone, l'altra parte del gradiente dei titoli
`testo`	il testo del sito e degli articoli
`accento`	piccoli segni (per esempio i punti verdi degli indirizzi nella home)
`sfondo`	lo sfondo della pagina; con uno sfondo chiaro i "vetri" del sito diventano grigi invece che bianchi, da soli

I quattro colori personalizzati con gli `id` **`sfondo-2`**, **`testo-2`**, **`testo-3`**, **`bordo`** hanno un significato speciale: il CMS li usa per lo sfondo secondario (riquadri, tendine, popup), il testo attenuato (date, sottotitoli, voci di menu), il testo discreto e i bordi. Mettili sempre, con questi `id`. Puoi aggiungere altri colori (fino a 40 in tutto): diventano colori globali che chi gestisce il sito può usare nell'editor.

Formati ammessi: `#rgb`, `#rrggbb`, `#rrggbbaa`, `rgb()`, `rgba()`, `hsl()`, `hsla()`, `transparent`. Gli `id` dei personalizzati: lettere minuscole, cifre e trattini, fino a 40.

Leggibilità. Il testo e lo sfondo devono avere un contrasto di almeno **4,5 a 1** (norme WCAG). Controlla anche `testo-2` sullo sfondo e il testo dei pulsanti (capitolo 7.4).

5.2 `font`

```
"font": {
  "sistema": {
    "primario": { "famiglia": "Trebuchet MS" },
    "secondario": { "famiglia": "Trebuchet MS" },
    "testo": { "famiglia": "Inter" },
    "accento": { "famiglia": "Inter" }
  }
}
```

`primario` è il carattere dei titoli, `testo` quello del testo. Le famiglie ammesse sono un elenco chiuso:

Gruppo	Famiglie	Note
In casa (serviti dal sito)	Inter, Space Grotesk	nessuna chiamata esterna
Sistema (già sul computer di chi legge)	Carattere del sistema, Arial, Helvetica, Verdana, Tahoma, Trebuchet MS, Georgia, Times New Roman, Garamond, Palatino, Courier New, Monospazio del sistema	nessuna chiamata esterna; l'aspetto varia un poco fra i computer
Google Fonts	Roboto, Open Sans, Lato, Montserrat, Poppins, Raleway, Nunito, Nunito Sans, Work Sans, DM Sans, Manrope, Rubik, Outfit, Plus Jakarta Sans, Source Sans 3, Oswald, Fira Sans, IBM Plex Sans, Josefin Sans, Quicksand, Mulish, Karla, Barlow, Archivo, Figtree, Lexend, Urbanist, Sora, Be Vietnam Pro; Playfair Display, Merriweather, Lora, PT Serif, Libre Baskerville, EB Garamond, Crimson Text, Cormorant Garamond, Fraunces, Bitter, Roboto Slab, IBM Plex Serif, Source Serif 4, Spectral, Literata, Noto Serif, DM Serif Display, Libre Caslon Text; Fira Code, JetBrains Mono, Roboto Mono, Source Code Pro, IBM Plex Mono, Space Mono	il sito li carica da Google: ogni visita fa una richiesta a Google. Per la privacy dei visitatori (GDPR) è meglio preferire i caratteri in casa o di sistema

Una famiglia fuori elenco viene scartata e torna quella di partenza. Ogni voce può avere anche `peso`, `dimensione`, `interlinea` eccetera, come la tipografia qui sotto.

5.3 `tipografia`

Ritocchi per elemento. Ogni voce ha `colore` e `tipografia`; `corpo` ha in più `spazio_paragrafi`.

```
"tipografia": {
  "h1": { "colore": "", "tipografia": { "peso": "800", "spaziatura_lettere": "-0.5px" } },
  "link": { "colore": "globale:secondario", "tipografia": { "decorazione": "underline" } }
}
```

Voce	Elementi del sito
`corpo`	`body`
`paragrafo`	`p`
`link`, `link_hover`	i link dentro il contenuto (non pulsanti, card, marchio)
`h1` … `h6`	i titoli, comprese le varianti del sito (`.hero h1`, `.card-titolo`…)

Campi di `tipografia`:

Campo	Valori
`famiglia`	una famiglia dell'elenco, oppure `globale:primario`, `globale:testo`…
`dimensione`, `interlinea`	per misura: `{"d": "48px", "t": "40px", "m": "32px"}` (computer, tablet, telefono); l'interlinea anche senza unità (`1.2`)
`peso`	`100` … `900`
`stile`	`normal`, `italic`
`trasformazione`	`none`, `uppercase`, `lowercase`, `capitalize`
`decorazione`	`none`, `underline`, `line-through`, `overline`
`spaziatura_lettere`, `spaziatura_parole`	una misura, anche negativa (`-0.5px`)

Misure: un numero con `px`, `em`, `rem`, `%`, `vw`, `vh` o `ch`; un numero senza unità vale pixel. Un colore può essere scritto come riferimento a un colore globale: `globale:primario`, `globale:testo-2` e così via.

5.4 `pulsanti`

Valgono per tutti i pulsanti del sito (`.btn`).

Campo	Valori
`tipografia`	come sopra
`ombra_testo`	un'ombra CSS (`0 1px 2px rgba(0,0,0,.3)`)
`normale`, `hover`	ciascuno con `colore`, `sfondo`, `bordo_colore`, `ombra`
`bordo_tipo`	`none`, `solid`, `dashed`, `dotted`, `double`
`bordo_larghezza`, `raggio`, `padding`	da una a quattro misure separate da spazio (`4px`, `12px 22px`)

Attenzione: `normale.sfondo` impone **lo stesso sfondo a tutti i pulsanti**, principale e secondario. Se vuoi tenere la differenza fra i due, lascialo vuoto e usa i colori `primario` e `secondario`.

5.5 `immagini`

Per le immagini dentro il contenuto (`main img`): `raggio`, `bordo_tipo`, `bordo_larghezza`, `bordo_colore`, `opacita` (da 0 a 1), `ombra`, `filtri` (`sfocatura` 0-50, `luminosita` 0-200, `contrasto` 0-200, `saturazione` 0-200, `tonalita` 0-360), `hover` (con `opacita`, `bordo_colore`, `ombra`, `filtri`), `transizione` (millisecondi, fino a 5000).

5.6 `campi`

I campi dei moduli (contatti, newsletter, ricerca):

Parte	Campi
`etichetta`	`colore`, `tipografia`
`campo`	`tipografia`, `colore`, `sfondo`, `bordo_colore`, `bordo_larghezza`, `raggio`, `padding`
`fuoco` (campo selezionato)	`colore`, `sfondo`, `bordo_colore`, `contorno`

5.7 `layout`

Campo	Valori	Partenza
`larghezza`	larghezza massima del contenuto in pixel, 600-2400	1180
`spazio_widget`	spazio sotto ogni widget, 0-200	24
`spazio_colonne`	spazio fra le colonne, 0-200	32
`tablet`	sotto questa larghezza vale la misura "tablet", 600-1600	1024
`telefono`	sotto questa larghezza vale la misura "telefono", 320-1000 (sempre minore di `tablet`)	767

5.8 `sfondo`

Campo	Valori
`colore`	colore dello sfondo (vince su `colori.sistema.sfondo` solo per lo sfondo della pagina)
`immagine`	indirizzo che comincia con `/` o `https://`
`posizione`	`center center`, `center top`, `left top` … (9 posizioni)
`ripetizione`	`no-repeat`, `repeat`, `repeat-x`, `repeat-y`
`dimensione`	`cover`, `contain`, `auto`
`allegato`	`scroll`, `fixed`
`velo`	un colore trasparente sopra l'immagine (`rgba(0,0,0,.4)`)
`sfumature_sito`	`true` o `false`: le sfumature e le stelle dell'aspetto di partenza. Per un tema chiaro o sobrio mettilo a `false`

Un'immagine di sfondo deve stare su un indirizzo pubblico: le immagini del tema stanno in `temi/<nome>/`, quindi l'indirizzo è `/temi/<nome>/sfondo.jpg` (se il CMS è in una cartella, il CMS aggiunge la cartella da solo ai link della pagina, ma non dentro il CSS: in quel caso usa `tema.css` con un percorso relativo, capitolo 7).

6. `modelli.json`: header e footer

```
{
  "header": { "versione": 1, "elementi": [ ... ] },
  "footer": { "versione": 1, "elementi": [ ... ] }
}
```

Header e footer sono pagine dell'editor visuale in miniatura: **alberi** di sezioni, colonne e widget. Puoi dare solo l'header, solo il footer, o nessuno dei due (restano quelli del sito).

6.1 Il modo comodo

Componi header e footer nell'editor del pannello (Editor > Header e footer, Modifica), rendili predefiniti, poi **Scarica l'aspetto di adesso come tema**: `modelli.json` esce già pronto, con il menu indicato per posizione.

6.2 Il formato dell'albero

```
{
  "versione": 1,
  "elementi": [
    {
      "id": "lavh01",
      "tipo": "sezione",
      "impostazioni": { "larghezza": "contenuta", "spazio_sopra": { "d": 0, "m": 0 },
"spazio_sotto": { "d": 0, "m": 0 } },
      "figli": [
        {
          "id": "lavh02",
          "tipo": "colonna",
          "impostazioni": {},
          "figli": [
            { "id": "lavh03", "tipo": "logo_sito", "impostazioni": { "margine_sotto": { "d": 0 } }
},
            { "id": "lavh04", "tipo": "menu_nav", "impostazioni": { "menu": "posizione:principale",
"spingi": true } }
          ]
        }
      ]
    }
  ]
}
```

- `elementi` contiene **sezioni**; ogni sezione contiene **colonne** (fino a 6); ogni colonna contiene **widget** (fino a 100). Al massimo 200 sezioni.
- `id`: da 4 a 12 caratteri, lettere minuscole e cifre, diversi fra loro. Un `id` sbagliato o ripetuto viene sostituito da uno nuovo.
- `impostazioni`: scrivi solo quelle che cambiano; le altre prendono il valore di partenza del widget. Un widget di tipo sconosciuto viene tolto; un'impostazione non valida torna al valore di partenza.
- Le misure che cambiano con lo schermo si scrivono `{"d": computer, "t": tablet, "m": telefono}`; basta `d`.
- In un header le colonne si dispongono in riga: logo a sinistra, menu a destra con `"spingi": true`.

6.3 Il menu: per posizione, mai per numero

Ogni menu di un sito ha un numero che cambia da un'installazione all'altra. In un tema il menu si indica **per posizione**:

```
{ "tipo": "menu_nav", "impostazioni": { "menu": "posizione:principale" } }
```

All'attivazione (e in anteprima) il CMS lo traduce nel numero del menu che, su quel sito, ha la posizione indicata (`principale` è il menu creato dall'installatore). Se nessun menu ha quella posizione, il widget resta senza menu.

6.4 Le impostazioni che servono di più

Sezione (`sezione`):

Impostazione	Valori
`larghezza`	`contenuta`, `stretta`, `piena`, `personalizzata` (con `larghezza_px`), `articolo`
`spazio_sopra`, `spazio_sotto`	pixel per misura; partenza `{"d":72,"m":48}`, in un header di solito `0`
`spazio_colonne`	pixel
`allineamento_verticale`	`stretch`, `flex-start`, `center`, `flex-end`
`sfondo`	`nessuno`, `vetro`, `gradiente`, `colore` (con `colore_sfondo`), `immagine` (con `immagine_sfondo` e `velo` 0-95), `avanzato`
`colore_testo`, `colore_titoli`, `colore_link`	colori, anche `globale:testo-2`
`sticky`	`alto` o `basso`: la sezione resta attaccata al bordo

Colonna (`colonna`): `larghezza` (percentuale per misura), `allineamento_verticale`, `allineamento_orizzontale`, `riquadro`, `spazio_interno`, `colore_testo`, `colore_titoli`, `colore_link`.

Widget del tema (gruppo "Tema" nella tavolozza dell'editor):

Widget	Cosa mostra	Impostazioni principali
`logo_sito`	nome del sito, sottotitolo, logo	`modo` (`testo`, `immagine`, `entrambi`), `maiuscolo`, `nome_colore`, `larghezza_immagine`
`nome_sito`	il nome del sito	
`descrizione_sito`	la frase del sito	
`menu_nav`	un menu	`menu` (`posizione:<nome>`), `disposizione` (`orizzontale`, `verticale`, `tendina`), `spingi`, `collasso` (sotto questa larghezza diventa il menu del telefono, partenza 960), `hamburger_testo`, `indicatore` (`linea`, `sfondo`, `punto`), `indicatore_colore`, `voce_colore`, `attiva_colore`
`ricerca`	la casella di ricerca (va a `/?s=`)	
`breadcrumb`	le briciole di pane	

Widget	Cosa mostra	Impostazioni principali
`copyright`	"© anno sito" (`{anno}`, `{sito}`)	
`titolo_contenuto`, `testo_contenuto`, `immagine_evidenza`, `info_contenuto`	pezzi del contenuto corrente	

Nome, sottotitolo, frase e logo, se lasciati vuoti nel widget, vengono dallo Stile del sito di chi usa il tema: **lasciali vuoti**, così il tema va bene per ogni sito.

Nell'header e nel footer si possono usare tutti gli altri widget dell'editor (testo, pulsante, icone, social…). L'elenco completo dei widget e delle loro impostazioni sta in `editor-pagine.md`.

6.5 I colori dentro gli alberi

Nei colori dei widget, oltre a `#rrggbb` e `rgba()`, si possono scrivere:

- `globale:<id>`: un colore dello Stile del sito (`globale:primario`, `globale:testo-2`, `globale:bordo`). Segue lo stile di chi usa il tema: è la scelta migliore.
- `ig:<nome>`: un colore della tavolozza delle infografiche (`ig:ciano-2`, `ig:rosso`, `ig:sfondo`), capitolo 8.

6.6 Attivare di nuovo

Riattivando lo stesso tema, il CMS riusa i modelli creati la volta prima (non li duplica) e li rende di nuovo predefiniti. Se hai aggiornato `modelli.json`, per avere i modelli nuovi chi gestisce il sito deve cestinare "Header Nometema" e "Footer Nometema" e premere di nuovo **Usa questo tema**.

7. `tema.css`: il CSS in più

Il nome del file lo scegli in `tema.json` (`"css": "tema.css"`).

7.1 Dove si carica e chi vince

Nell'`<head>` di ogni pagina del sito, in quest'ordine:

1. `sito/assets/sito.css` e `sito/assets/tema.css` del CMS (l'aspetto di partenza);
2. `<style id="tema-ig">` con i colori delle infografiche del tema (capitolo 8);
3. **il tuo `tema.css`** (`<link id="tema-css">`);
4. `<style id="g-css">` dello Stile del sito: variabili e regole che chi gestisce il sito ha scelto, e il suo CSS personalizzato;
5. il CSS delle sezioni fatte con l'editor.

Quindi, a parità di selettore, **lo Stile del sito e il CSS personalizzato vincono sul tuo**: è voluto, perché chi gestisce il sito deve poter ritoccare il tema. Il tuo CSS vince sul CSS di partenza del CMS.

`tema.css` si carica anche nella **tela dell'editor** del pannello, così chi compone le pagine le vede con il tema.

7.2 Le variabili CSS

Il CSS del sito è costruito su variabili. Prima di scrivere regole nuove, guarda se basta cambiare una variabile.

Variabili che vengono dallo Stile del sito: si cambiano da `stile.json`, non da `tema.css` (lo Stile del sito viene dopo e le riscrive).

Variabile	Viene da
`--g-colore-primario`, `--g-colore-secondario`, `--g-colore-testo`, `--g-colore-accento`, `--g-colore-sfondo`, `--g-colore-<id>` dei personalizzati	`colori`
`--g-font-primario`, `--g-font-testo`, … (e `-peso`, `-dimensione`)	`font`
`--g-larghezza`, `--g-spazio-widget`, `--g-spazio-colonne`	`layout`
`--viola` (primario), `--ciano` (secondario), `--testo`, `--bg` (sfondo), `--bg-2`, `--testo-2`, `--testo-3`, `--bordo`	i colori, con i nomi storici di `sito.css`
`--viola-rgb`, `--ciano-rgb`, `--bg-rgb`	i tre numeri dei colori, per le trasparenze: `rgba(var(--viola-rgb),.2)`
`--f-titolo`, `--f-testo`, `--max`	caratteri e larghezza

Variabili ricavate, che il tuo `tema.css` può cambiare (sono in `:where(:root)`, che non pesa: un tuo `:root{...}` le scavalca sempre):

Variabile	Cosa fa	Partenza
`--primario-chiaro`, `--primario-scuro`	i due capi del gradiente del pulsante principale	ricavati dal primario
`--secondario-chiaro`, `--secondario-scuro`	i due capi del gradiente del pulsante secondario	ricavati dal secondario

Variabile	Cosa fa	Partenza
`--su-secondario`	il testo sopra il colore secondario	bianco o quasi nero, secondo la luminosità
`--testo-corpo`	il testo degli articoli	il colore del testo
`--accento`	piccoli segni	il colore d'accento
`--inchiostro`, `--inchiostro-rgb`	il colore dei "vetri" (riquadri appena più chiari o più scuri dello sfondo)	bianco su sfondo scuro, nero su sfondo chiaro
`--vetro`, `--vetro-2`	lo sfondo dei riquadri e delle card	inchiostro al 4,5% e all'8%
`--successo`, `--successo-chiaro`, `--errore`, `--errore-chiaro`, `--errore-testo` (e `-rgb`)	messaggi dei moduli, liste con spunta	verde e rosso
`--r`	raggio degli angoli di card e riquadri	18px
`--grad`	gradiente dei titoli (`linear-gradient(100deg,var(--ciano),var(--viola))`)	
`--velo-1`, `--velo-2`, `--velo-3`	i colori del velo sulle foto di Apertura (tre) e Presentazione (i primi due)	`#1aa7c4`, `#5b21b6`, `#2e1065`
`--velo-o`	l'intensità del velo, da 0 a 1	1
`--fx-1`, `--fx-2`	i colori degli effetti animati delle foto (cornice, alone, scintille)	`--ciano` e `--viola`

Le variabili del velo e degli effetti non sono in `:where(:root)`: il tema le scrive in `:root{...}` e valgono su tutte le foto; chi usa il sito le cambia poi widget per widget dalla scheda Stile, sezione Foto, che vince sul tema.

7.3 Le classi del sito

Il sito usa classi italiane e stabili. Le principali:

Zona	Classi
Struttura	`.wrap` (contenitore centrato), `main#contenuto`, `.salta` (link "Vai al contenuto")
Header e footer	`.top` e `.piede` (quelli scritti nel codice), `.ed-testata.ed-modello` e `.ed-piede.ed-modello` (quelli fatti con l'editor, cioè quasi sempre), `.marchio`, `.nav`, `.mn-*` (menu)
Pulsanti	`.btn`, `.btn-viola` (principale), `.btn-ciano` (secondario), `.btn-filo` (contorno)
Testate di sezione	`.occhiello`, `.sez-titolo`, `.sez-intro`, `.hero`, `.lead`
Elenchi di articoli	`.griglia`, `.card`, `.card-titolo`, `.card-meta`, `.filtri`, `.paginazione`
Articolo	`.art`, `.art-testa`, `.art-meta`, `.toc` (sommario), `.corpo` (il testo), `.in-breve`, `.cta-box`
Riquadro in fondo agli articoli (1.6.0)	`aside.fine-articolo#fine-articolo` (Articoli > Aggiungi sempre alla fine): sfondo `var(--bg-2)`, bordo `var(--bordo)`, angoli `var(--r)`, ombra. Se il tema ridefinisce queste variabili il riquadro le segue da solo; per cambiargli l'ombra o lo spazio interno si scrive una regola su `.fine-articolo`
Pagine dell'editor	`.ed-pagina`, `.ed-sezione`, `.ed-contenitore`, `.ed-riga`, `.ed-colonna`, `.ed-widget`, `.ed-w-<tipo>` (un widget di un tipo), `.ed-<id>` (un elemento preciso)

Zona	Classi
Moduli	`.modulo`, `.modulo-campi`, `.cerca`
Widget extra	`.ew-*` (vedi `sito/assets/widget.css`)
Infografiche	`.igx`, `.igx-*` (capitolo 8)
Popup	`.popup`, `.popup-finestra`

Per vedere le classi di una pagina: tasto destro, **Ispeziona**, nel browser.

7.4 Consigli

- **Colori in `stile.json`, forme in `tema.css`.** Se metti i colori nel CSS, chi gestisce il sito non li ritrova nello Stile del sito e non li può cambiare.
- **Il testo sopra il colore primario è bianco.** Se il tuo primario è chiaro (giallo, verde acqua), scrivi in `tema.css` un colore scuro per `.btn-viola`. Lo fa l'esempio Lavagna.
- **Immagini del tema nel CSS con percorsi relativi**: `background:url(img/carta.jpg)` dentro `tema.css` cerca il file in `temi/<nome>/img/`, ovunque sia installato il CMS.
- **Niente `@import` di CSS esterni** e niente richieste ad altri siti: il CMS è fatto per non chiamare nessuno.
- **Provalo su un sito con contenuti veri**, non solo sulla home d'esempio.

8. I colori delle infografiche

Le infografiche dell'editor (widget `ig_*`: passi, linea del tempo, checklist, tabella…) usano una tavolozza di colori con un nome: rosso, oro, verde, ciano, viola, scuro, "corpo crema", "testata scura". **I colori con un nome li decide il tema**; i colori scritti a mano su un'infografica restano. Senza tema la tavolozza è "Carta", l'aspetto originale.

Nel `tema.json`:

```
"infografiche": {
  "sfondo": "#26332e",
  "bianco": "#2c3a35",
  "bordo": "#3a4a44",
  "riga": "#33423c",
  "titolo": "#f1efe6",
  "testo": "#d6d3c6",
  "muto": "#a8a597",
  "situazione": "#2f3d38",
  "su_testata": "#ffffff",
  "su_scuro": "#d6d3c6",
  "voci": { "rosso": "#d9694c", "oro": "#e0b63c", "verde": "#5f9c87", "ciano": "#4f97ad", "viola":
"#8f78b8", "scuro": "#121a17" },
  "variabili": { "mattone": "#b5523a" }
}
```

Campo	Cosa colora	Carta
`sfondo`	il corpo della card	`#fef9f0` (crema)
`bianco`	i riquadri dentro la card	`#ffffff`
`bordo`	il bordo della card	`#e8ddd0`
`riga`	le righe fra le voci	`#f0e6d8`
`titolo`	i titoli delle voci	`#2d1b0e`
`testo`	il testo	`#6b4c35`
`muto`	firma e note	`#9c7b65`
`situazione`	la riga "situazione" del prima e dopo	`#f5f0ec`
`su_testata`	il testo sopra la testata colorata	`#ffffff`
`su_scuro`	il testo dentro i riquadri scuri	`#c4b5a5`
`voci`	i sei colori principali	rosso `#e8553e`, oro `#e8a830`, verde `#6b8f71`, ciano `#4a7c82`, viola `#7b5ea7`, scuro `#2d1b0e`

Tutto è facoltativo. Dai sei colori di `voci` il CMS ricava il resto della tavolozza: la seconda tinta di ogni colore (più scura; per lo scuro un po' più chiara), le tinte chiare degli occhielli, arancio, giallo, oro scuro, verde scuro, ciano scuro, rosso scuro, mattone, lilla, marrone. Se una tinta ricavata non ti piace, scrivila a mano in `variabili`, senza il prefisso `--ig-`. Le variabili disponibili:

`rosso`, `rosso-2`, `rosso-scuro`, `rosso-scuro-2`, `arancio`, `arancio-2`, `oro`, `oro-2`, `oro-scuro`, `oro-scuro-2`, `giallo`, `giallo-2`, `verde`, `verde-2`, `verde-scuro`, `verde-scuro-2`, `ciano`, `ciano-2`, `ciano-scuro`, `ciano-scuro-2`, `ciano-profondo`, `viola`, `viola-2`, `scuro`, `scuro-2`, `marrone`, `marrone-2`, `bianco`, `bianco-2`, `sfondo`, `situazione`, `bordo`, `riga`, `titolo`, `testo`, `muto`, `su-testata`, `su-scuro`, `rosso-chiaro`, `verde-chiaro`, `ciano-chiaro`, `viola-chiaro`, `oro-chiaro`, `oro-scuro-chiaro`, `lilla`, `mattone`, `mattone-2`, `mattone-chiaro`.

Si accettano solo colori validi: un valore che non è un colore (per esempio `url(...)`) viene scartato.

La stessa tavolozza colora anche la CTA (sfumatura `ciano-2` e `ciano-profondo`, pulsante `rosso`), il riquadro del test e la testata della Scheda. Nella pagina Temi del pannello ogni scheda mostra i sei pallini di `voci`.

Tema scuro? Le infografiche hanno il loro corpo: su un sito scuro conviene dare `sfondo`, `bianco`, `titolo` e `testo` scuri e chiari come nell'esempio, altrimenti restano card color crema sul fondo scuro (che può anche essere una scelta).

Le quindici infografiche disegnate (imbuto, piramide, ciambella, lancetta…) usano la stessa tavolozza: le forme prendono i colori di `voci` e delle tinte ricavate, le scritte `titolo` e `testo`, le linee `bordo` e `riga`, gli omini spenti `muto`. Il testo sopra un colore pieno (dentro un gradino dell'imbuto, una tappa del ciclo) e il titolo della loro testata li sceglie il CMS, bianco o nero secondo il colore: non serve scriverli nel tema. Se il tuo `tema.css` cambia il colore di una classe (`.igx .igx-c-oro{--c:var(--ig-oro-scuro)}`), anche quel testo si adatta. Il foglio di queste quindici è `sito/assets/infografiche-svg.css`, caricato solo nelle pagine che ne hanno una; le sue classi cominciano con `igs-`.

9. Il PHP del tema: filtri, eventi e pagine ridisegnate

Il PHP serve quando il CSS non basta: cambiare un pezzo di testo calcolato dal CMS, aggiungere una riga in un file, ridisegnare una pagina. **Prima di usarlo, chiediti se basta il CSS**: un tema senza PHP si carica senza firma, non può rompere niente e piace di più a chi gestisce i siti.

9.1 Le regole

- Il PHP del tema gira **solo sul sito pubblico**. Nel pannello non gira mai: un tema rotto non chiude fuori nessuno.
- Gira dentro il CMS: hai a disposizione le sue funzioni (`h()`, `cms_impostazione()`, le viste del sito...).
- **Non stampare niente** (`echo`, HTML fuori da `<?php`): ritorna i valori. **Non chiamare `exit`**.
- **Non dichiarare funzioni con nomi generici**: due funzioni con lo stesso nome sono un errore fatale. Se ti serve una funzione, usa una funzione anonima o un prefisso con il nome del tema (`lavagna_...`) e `if (!function_exists(...))`.
- I file PHP del tema non si aprono dal browser (403): non scrivere file pensati per essere chiamati da fuori.

9.2 Se il tema sbaglia

Errore	Cosa succede
un'eccezione o un errore in una vista, in un filtro, in `funzioni.php` (anche un errore di sintassi)	il CMS lo prende: la richiesta continua con l'aspetto di base, l'errore va nel log (`logs/php-errori.log`, riga `temi: <nome>: ...`), il tema si segna come **guasto**
un errore fatale che PHP non lascia prendere (per esempio una funzione dichiarata due volte)	quella sola richiesta risponde con un errore 500; alla fine della richiesta il CMS riconosce il file del tema e lo segna come guasto

Da quando è guasto, il sito usa l'aspetto di base. Nel pannello, pagina Temi, compare un avviso rosso con l'errore e i pulsanti **Riprova ad attivarlo** e **Resta sull'aspetto di base**. In anteprima un errore non segna il tema come guasto: puoi sbagliare tranquillo mentre lo provi.

9.3 `funzioni.php`: filtri ed eventi

```
<?php
return [
    'filtri' => [
        'llms_intestazione' => function ($righe) {
            if (!is_array($righe)) return $righe;
            $righe[] = '> Aspetto: tema Lavagna di plumeCMS.';
            return $righe;
        },
    ],
    'eventi' => [],
];
```

Il file **ritorna un array** con `filtri` ed `eventi`, come il manifesto di un modulo. Un **filtro** riceve un valore (più eventuali argomenti) e ritorna il valore, cambiato o no. Un **evento** viene chiamato e non ritorna niente. Il tema passa **per ultimo**, dopo i moduli: se un modulo e il tema cambiano la stessa cosa, vince il tema.

I filtri e gli eventi disponibili:

Il PHP del tema gira solo sul sito pubblico, quindi contano solo i filtri e gli eventi chiamati mentre il sito risponde a una pagina. Quelli del pannello (per esempio `dashboard_numeri`, `redattore_pagine`) per un tema non partono mai.

Nome	Tipo	Riceve	Deve ritornare	Quando
`vista_home`	filtro	`null`	l'HTML della home (stringa) oppure `null` per lasciarla al CMS	home scritta nel codice (pagina iniziale non fatta con l'editor)
`testata_fissa`	filtro	`null`, `$menu`, `$menuMobile` (HTML delle voci)	`[$testata, $piede]` oppure `null`	header e footer scritti nel codice, usati solo se non esiste nessun modello di Header e footer
`sito_menu`	filtro	elenco di voci `[['Home', '/'], ['Blog', '/blog/']]`	l'elenco	il menu dell'header scritto nel codice (stesso caso di sopra)
`blog_filtri`	filtro	elenco di slug di categoria (vuoto)	l'elenco	i filtri per categoria in testa al blog
`categorie_noindex`	filtro	elenco di slug	l'elenco	categorie i cui archivi e articoli non vanno indicizzati
`sito_contenuti`	filtro	tutti i contenuti pubblici, e se si stanno leggendo per i redirect	l'elenco	contenuti in più da pubblicare, nello stesso formato
`sito_html_contenuto`	filtro	`null`, il contenuto `$c`	l'HTML del contenuto oppure `null`	il testo di un contenuto non fatto con l'editor
`sito_mappa`	filtro	la mappa degli indirizzi, i contenuti attivi	la mappa	aggiungere o cambiare indirizzi del sito
`servizi_predefiniti`	filtro	elenco (vuoto)	l'elenco	le pagine "servizio" per i dati strutturati
`jsonld_about`	filtro	elenco (vuoto)	l'elenco	i temi ("about") del JSON-LD della home
`seo_predefiniti`	filtro	la configurazione SEO di partenza	la configurazione	valori di partenza della pagina SEO
`llms_intestazione`	filtro	le righe di testa di `llms.txt`, e se tutte le guide hanno il riassunto	le righe	il file `/llms.txt`
`llms_servizi`	filtro	elenco di percorsi (vuoto)	l'elenco	la sezione "Servizi" di `/llms.txt`
`sito_query`	evento	la query string come array, il percorso	niente	ogni richiesta con una query string, prima di tutto il resto
`sito_genere`	evento	la voce della mappa, i metadati	niente (chi risponde chiude la richiesta)	un genere di pagina che il CMS non conosce

Il filtro `design_predefiniti` esiste anche sul sito, ma non usarlo da un tema: i colori del tema stanno in `stile.json`, e cambiare i predefiniti solo sul sito darebbe un pannello che mostra valori diversi da quelli veri.

I filtri e gli eventi sono gli stessi dei moduli del sito: un tema con PHP ne usa un sottoinsieme, elencato qui sotto.

9.4 Le viste: ridisegnare una pagina

Un file in `viste/` ridisegna una pagina del sito e **ritorna l'HTML come stringa**. Se ritorna qualcosa che non è una stringa, il CMS usa la sua vista.

File	Cosa ridisegna	Variabili che riceve
`viste/documento.php`	la pagina intera, da `<!DOCTYPE html>` a `</html>`	`$testa` (il contenuto dell'`<head>`, già con CSS e metadati), `$testata` (header), `$corpo` (il contenuto), `$piede` (footer), `$popup`, `$meta` (titolo, descrizione, robots, canonical, og)
`viste/home.php`	la home scritta nel codice, quando la pagina iniziale non è fatta con l'editor	nessuna
`viste/articolo.php`	un articolo non fatto con l'editor	`$c` (il contenuto: titolo, percorso, data, autore, categorie…), `$html` (il testo già pronto)
`viste/pagina.php`	una pagina non fatta con l'editor	`$c`, `$html`
`viste/blog.php`	la pagina del blog	`$v`: `articoli`, `pagina`, `totale`
`viste/archivio.php`	una categoria o un tag	`$v`: `tassonomia`, `nome`, `descrizione`, `articoli`, `base`, `pagina`, `totale`
`viste/cerca.php`	la ricerca	`$q` (la parola cercata), `$trovati` (elenco di `['c' => contenuto, 'e' => estratto]`)
`viste/404.php`	la pagina non trovata (il codice resta 404)	nessuna

Le pagine fatte con l'editor visuale non passano dalle viste: il loro aspetto lo danno stile, CSS e widget.

Dalla 1.6.0, se l'amministratore ha acceso il riquadro in fondo agli articoli, il CMS lo mette prima dell'ultimo `</article>` dell'HTML che ritorna `viste/articolo.php`. Una vista dell'articolo che non scrive `<article>` lo riceve subito dopo, largo quanto la colonna di lettura: non serve fare niente, ma con `<article>` il riquadro sta dove deve.

Funzioni del CMS utili nelle viste:

Funzione	Cosa fa
`h($testo)`	rende sicuro un testo da mettere nell'HTML. **Usala su ogni testo** che viene dal database o dall'indirizzo
`cms_impostazione('sito_nome')`	un'impostazione del sito (`sito_nome`, `sito_sottotitolo`, `sito_descrizione`, `blog_titolo`…)
`sito_ultimi(3)`	gli ultimi articoli pubblicati
`vista_griglia($articoli)`	la griglia di card degli articoli
`vista_scheda($articolo)`	una card

Funzione	Cosa fa
`vista_paginazione($base, $pagina, $totale)`	i numeri delle pagine
`vista_testa_sezione($occhiello, $titolo, $intro)`	occhiello, titolo e introduzione con le classi del sito. Attenzione: non rende sicuri i testi, passali già con `h()`

I link scritti come `/blog/` vanno bene anche con il CMS in una cartella: il CMS aggiunge la cartella da solo a tutti i link della pagina.

Un esempio (la 404 di Lavagna):

```
<?php
$ultimi = function_exists('sito_ultimi') ? sito_ultimi(3) : [];
$html = '<div class="wrap" style="padding:96px 0 72px">'
    . vista_testa_sezione('Errore 404', 'Qui la lavagna è pulita.', 'La pagina che cercavi non
c&#039;è, o ha cambiato indirizzo.')
    . '<p><a class="btn btn-viola" href="/">Torna alla home</a></p>';
if ($ultimi) {
    $html .= '<h2 class="sez-titolo" style="margin-top:56px;font-size:28px">Intanto, gli ultimi
articoli</h2>'
        . '<div style="margin-top:24px">' . vista_griglia($ultimi) . '</div>';
}
return $html . '</div>';
```

9.5 Caricare un tema con PHP

Un tema con anche un solo file `.php` si carica dal pannello solo in due modi:

1. **firmato** con la chiave del sito su cui va caricato (capitolo 11.3);
2. con la **finestra di un'ora** aperta da chi gestisce il sito: Impostazioni > Sicurezza, scheda Impostazioni, riquadro della finestra, password, **Apri la finestra per un'ora**.

È una protezione per chi gestisce il sito: il PHP di un tema gira sul suo server con i suoi dati.

10. Provare il tema

10.1 Anteprima

Pannello, Editor > Temi, **Anteprima** sul tuo tema: si apre il sito con il tema, solo per te (l'indirizzo è `/?preview=1&tema=<nome>`, e funziona solo con l'accesso al pannello). Puoi navigare: header, footer, stile, CSS, infografiche e viste sono quelli del tema su ogni pagina che apri da lì, aggiungendo `?preview=1&tema=<nome>` all'indirizzo. Per vedere il sito senza nessun tema: `?preview=1&tema=base`.

10.2 La lista dei controlli

Su computer **e** telefono (nel browser: Ispeziona, icona del telefono):

- [ ] home, un articolo lungo, la pagina del blog, una categoria, la ricerca, la 404, la pagina dei contatti;
- [ ] il testo si legge: contrasto almeno 4,5 a 1 fra testo e sfondo (anche il testo attenuato, le date, i pulsanti); strumenti gratuiti: "WebAIM Contrast Checker";
- [ ] il menu del telefono si apre e si chiude;
- [ ] i campi dei moduli si vedono e si capisce quale è selezionato;
- [ ] una pagina con le infografiche: colori in armonia, testo leggibile anche nei riquadri scuri;
- [ ] la tela dell'editor nel pannello: le pagine si vedono con il tema;
- [ ] Stile del sito: cambia il colore primario e controlla che il tema lo segua (se non lo segue, hai messo un colore nel CSS invece che in `stile.json`);
- [ ] il CMS installato in una cartella (`/prova`): immagini e CSS del tema si caricano;
- [ ] con PHP: il log `logs/php-errori.log` è pulito dopo aver girato il sito.

10.3 Una prova automatica

Il repository di plumeCMS ha una prova che attiva un tema e controlla pagine, menu e contrasto del testo:

```
node prove/prova-temi-partenza.js https://tuosito.it/prova <cartella del CMS sul disco> <nome-del-tema>
```

Serve un sito di prova sul tuo computer (vedi `prove/LEGGIMI.md`).

11. Impacchettare, firmare, distribuire

11.1 Lo zip

Lo zip contiene **una cartella con il nome del tema** e dentro i file:

```
lavagna.zip
  lavagna/tema.json
  lavagna/stile.json
  lavagna/modelli.json
  lavagna/tema.css
  lavagna/anteprima.png
  lavagna/funzioni.php
  lavagna/viste/404.php
```

Va bene anche lo zip con `tema.json` nella radice, senza cartella: allora il nome della cartella viene da `slug` in `tema.json` (o dal nome). Il pulsante **Scarica** del pannello produce sempre lo zip giusto.

Regole che il pannello controlla prima di scrivere qualsiasi cosa:

Regola	Limite
tipi di file ammessi	json, css, png, jpg, jpeg, webp, gif, avif, woff, woff2, ttf, otf, txt, md, php
vietati	SVG (possono contenere script), file nascosti (`.htaccess`, `.user.ini`…), `.phtml` e simili, qualunque altro tipo
nomi	lettere, cifre, `_`, `-`, `.`; niente spazi, niente `..`, niente percorsi assoluti
numero di file	400
dimensione	10 MB per file, 40 MB in tutto (estratti)
compressione	un file molto grande compresso in pochissimo spazio viene rifiutato (zip bomba)
contenuto	un solo tema per zip, con un `tema.json` valido

Le cartelle `__MACOSX` e i file `.DS_Store` del Mac vengono saltati da soli. Il server ha anche un suo limite per i file caricati (spesso 2 o 8 MB, si cambia da cPanel): tieni le immagini leggere.

11.2 Caricare

Pannello, Editor > Temi, in fondo **Carica un tema**: scegli lo zip, **Carica**. Il tema arriva in `temi/` ma **non si attiva da solo**. Per caricare una versione nuova dello stesso tema spunta **Sostituisci il tema con lo stesso nome**: la versione di prima resta sul server in `aggiornamenti/temi-precedenti/`. Se il tema era attivo, per applicare stile, header e footer nuovi si preme di nuovo **Usa questo tema**.

Ogni caricamento resta nel registro della pagina Sicurezza.

11.3 Firmare (solo per i temi con PHP)

La firma dimostra che lo zip è stato preparato da chi ha la chiave del sito e non è stato cambiato dopo. Si firma con lo strumento del repository di plumeCMS, sul computer di chi ha la chiave:

```
PACCHETTO_CHIAVE='la chiave' python3 tools/pacchetto.py --rifirma lavagna.zip
```

Scrive `lavagna-firmato.zip`, con dentro un `manifest.json` che porta la firma (non viene installato). La chiave è il valore di `CMS_FIRMA_PACCHETTI` nel file `includes/config.local.php` del sito (se non c'è, `CMS_SEGRETO`): la conosce solo chi gestisce quel sito. **Ogni sito ha la sua chiave**: uno zip firmato per un sito non passa su un altro. Un tema distribuito a molti siti, quindi, o è senza PHP, o si carica con la finestra di un'ora aperta.

Se nella pagina Sicurezza è spuntato *I pacchetti devono essere firmati*, anche i temi senza PHP vanno firmati.

11.4 Versioni e compatibilità

- Cambia `versione` in `tema.json` a ogni zip che distribuisci.
- I temi funzionano con plumeCMS **1.2.0** e successivi.
- Un tema non deve contenere niente del sito per cui è nato: nome, logo e testi li mette chi lo usa.

12. Esempio completo: Lavagna

Un tema scuro, verde ardesia con il testo color gesso, pulsanti gialli e dettagli menta. Usa tutto quello che si può usare: stile, header e footer, CSS, colori delle infografiche, un filtro PHP e una pagina 404 ridisegnata. È stato caricato, attivato e provato su plumeCMS 1.2.0 (pagine, menu, contrasto del testo, 404, filtro). I file sono anche in `esempi/tema-lavagna/` del repository.

`tema.json`

```
{
  "nome": "Lavagna",
  "versione": "1.0",
  "autore": "plumeCMS, esempio del manuale",
  "descrizione": "Verde ardesia, testo color gesso, giallo e menta. Il tema d'esempio del manuale
dei temi.",
  "anteprima": "anteprima.png",
  "css": "tema.css",
  "stile": "stile.json",
  "modelli": "modelli.json",
  "infografiche": {
    "sfondo": "#26332e",
    "bianco": "#2c3a35",
    "bordo": "#3a4a44",
    "riga": "#33423c",
    "titolo": "#f1efe6",
    "testo": "#d6d3c6",
    "muto": "#a8a597",
    "situazione": "#2f3d38",
    "su_testata": "#ffffff",
    "su_scuro": "#d6d3c6",
    "voci": {
      "rosso": "#d9694c",
      "oro": "#e0b63c",
      "verde": "#5f9c87",
      "ciano": "#4f97ad",
      "viola": "#8f78b8",
      "scuro": "#121a17"
    }
  }
}
```

`stile.json`

```
{
  "colori": {
    "sistema": {
      "primario": "#f2c94c",
      "secondario": "#7fb7a4",
      "testo": "#f1efe6",
      "accento": "#e4785b",
      "sfondo": "#1f2a26"
    },
    "personalizzati": [
      { "id": "sfondo-2", "nome": "Sfondo secondario", "valore": "#26332e" },
      { "id": "testo-2", "nome": "Testo attenuato", "valore": "#c9c6b8" },
      { "id": "testo-3", "nome": "Testo discreto", "valore": "#a8a597" },
      { "id": "bordo", "nome": "Bordi", "valore": "rgba(241,239,230,.16)" }
    ]
  },
  "font": {
    "sistema": {
      "primario": { "famiglia": "Trebuchet MS" },
      "secondario": { "famiglia": "Trebuchet MS" },
      "testo": { "famiglia": "Inter" },
      "accento": { "famiglia": "Inter" }
    }
  },
  "pulsanti": {
    "raggio": "4px"
  },
  "sfondo": {
    "sfumature_sito": false
  }
}
```

`modelli.json`

```
{
  "header": {
    "versione": 1,
    "elementi": [
      {
        "id": "lavh01",
        "tipo": "sezione",
        "impostazioni": { "larghezza": "contenuta", "spazio_sopra": { "d": 0, "m": 0 },
"spazio_sotto": { "d": 0, "m": 0 }, "spazio_colonne": { "d": 28 } },
        "figli": [
          {
            "id": "lavh02",
            "tipo": "colonna",
            "impostazioni": {},
            "figli": [
              { "id": "lavh03", "tipo": "logo_sito", "impostazioni": { "margine_sotto": { "d": 0 }
} },
              { "id": "lavh04", "tipo": "menu_nav", "impostazioni": { "menu":
"posizione:principale", "disposizione": "orizzontale", "spingi": true, "collasso": 960,
"hamburger_testo": "Menu", "indicatore": "linea", "indicatore_colore": "globale:primario",
"margine_sotto": { "d": 0 } } }
            ]
          }
        ]
      }
    ]
  },
  "footer": {
    "versione": 1,
    "elementi": [
      {
        "id": "lavf01",
        "tipo": "sezione",
        "impostazioni": { "larghezza": "contenuta", "colore_testo": "globale:testo-2",
"spazio_sopra": { "d": 56, "m": 56 }, "spazio_sotto": { "d": 40, "m": 40 } },
        "figli": [
          {
            "id": "lavf02",
            "tipo": "colonna",
            "impostazioni": {},
            "figli": [
              { "id": "lavf03", "tipo": "logo_sito", "impostazioni": { "margine_sotto": { "d": 8 }
} },
              { "id": "lavf04", "tipo": "descrizione_sito", "impostazioni": { "margine_sotto": {
"d": 16 } } },
              { "id": "lavf05", "tipo": "copyright", "impostazioni": { "margine_sotto": { "d": 0 }
} }
            ]
          }
        ]
      }
    ]
  }
}
```

tema.css

```
/* Lavagna: i colori stanno in stile.json; qui solo i ritocchi che lo Stile del sito non copre. */

/* I pulsanti gialli vogliono il testo scuro (il CMS lo mette bianco sul colore primario). */
:root{--su-secondario:#1f2a26}
.btn-viola,.ed-video button span{color:#1f2a26}
.btn-viola,.btn-ciano{box-shadow:none}

/* Una riga di gesso sotto l'header, al posto del vetro. */
.top,.ed-testata{background:#1f2a26;border-bottom:2px solid rgba(241,239,230,.22)}

/* Titoli un po' più stretti, occhielli color menta. */
h1,h2,.sez-titolo{letter-spacing:-.01em}
.occhiello{color:var(--ciano)}

/* Le card come fogli appoggiati sulla lavagna. */
.card,.servizio,.ew-post{background:#26332e}
```

funzioni.php

```
<?php
/**
 * Lavagna: il PHP del tema. Ritorna filtri ed eventi, come un modulo.
 * Gira solo sul sito pubblico, mai nel pannello. Un errore qui spegne il tema, non il sito.
 */
return [
    'filtri' => [
        // llms.txt: una riga in più nell'intestazione, per dire che il sito usa questo tema.
        'llms_intestazione' => function ($righe) {
            if (!is_array($righe)) return $righe;
            $righe[] = '> Aspetto: tema Lavagna di plumeCMS.';
            return $righe;
        },
    ],
    'eventi' => [],
];
```

`viste/404.php`

```
<?php
/**
 * Lavagna: la pagina "non trovata". Una vista ritorna l'HTML del contenuto della pagina come
stringa
 * (header, footer e <head> li mette il CMS). Il codice di risposta resta 404.
 * Funzioni del CMS usate: h() per scrivere testo sicuro, sito_ultimi() e vista_griglia() per gli
ultimi articoli,
 * vista_testa_sezione() per l'occhiello, il titolo e l'introduzione con le classi del sito.
 */
$ultimi = function_exists('sito_ultimi') ? sito_ultimi(3) : [];
$html = '<div class="wrap" style="padding:96px 0 72px">'
    . vista_testa_sezione('Errore 404', 'Qui la lavagna è pulita.', 'La pagina che cercavi non
c&#039;è, o ha cambiato indirizzo.')
    . '<p><a class="btn btn-viola" href="/">Torna alla home</a></p>';
if ($ultimi) {
    $html .= '<h2 class="sez-titolo" style="margin-top:56px;font-size:28px">Intanto, gli ultimi
articoli</h2>'
        . '<div style="margin-top:24px">' . vista_griglia($ultimi) . '</div>';
}
return $html . '</div>';
```

`anteprima.png`

Una foto della home con il tema, 800 per 500 pixel. Il modo più semplice: apri l'Anteprima, fai uno screenshot della parte alta della pagina, riducilo a 800 per 500.

Le scelte, spiegate

- **Il giallo come primario** vuole il testo scuro sui pulsanti: `tema.css` mette `.btn-viola{color:#1f2a26}` e `--su-secondario` scuro.
- **I quattro colori personalizzati** (`sfondo-2`, `testo-2`, `testo-3`, `bordo`) ci sono tutti: altrimenti riquadri, date e bordi resterebbero quelli dell'aspetto di partenza.
- **`sfumature_sito: false`**: niente stelle e sfumature viola su una lavagna.
- **Il footer** usa `globale:testo-2` per il testo: segue lo stile anche se chi usa il tema cambia i colori.
- **Il menu** è `posizione:principale`: funziona su qualsiasi sito.
- **Le infografiche** hanno corpo e riquadri scuri e testo chiaro, intonati alla lavagna.
- **Il PHP** è il minimo: un filtro innocuo e una 404 più utile. Con questi due file il tema si carica solo firmato o con la finestra aperta; togliendoli, si carica ovunque.

13. Errori comuni e come uscirne

Sintomo	Causa	Soluzione
Il tema non compare nel pannello	manca `tema.json`, o non è JSON valido, o la cartella ha un nome non ammesso	controlla `tema.json` con un validatore JSON; nome della cartella in minuscolo, senza spazi
"Tipo di file non ammesso nel tema: logo.svg"	SVG nello zip	converti in PNG o WebP
"Nome di file non ammesso nel tema"	un file nascosto o con spazi	togli i file che cominciano con il punto, rinomina senza spazi
"Nello zip non trovo tema.json"	`tema.json` è in una sottocartella più profonda, o ci sono due temi	una sola cartella con dentro `tema.json`
"Esiste già un tema nella cartella temi/..."	stesso nome di un tema presente	spunta Sostituisci
"contiene codice PHP"	tema con PHP non firmato, finestra chiusa	firma lo zip o apri la finestra (capitolo 9.5)
"La firma del tema non corrisponde"	zip cambiato dopo la firma, o firmato con la chiave di un altro sito	rifirma con la chiave giusta
I colori del tema non cambiano quando cambio lo Stile del sito	colori scritti in `tema.css`	spostali in `stile.json`
Il tuo CSS non ha effetto	lo Stile del sito o il CSS personalizzato vincono (vengono dopo)	usa un selettore più preciso, o la variabile giusta
Testo degli articoli poco leggibile	colori di testo e sfondo troppo vicini	contrasto almeno 4,5 a 1
Pulsante principale illeggibile	primario chiaro con testo bianco	`.btn-viola{color:<scuro>}` in `tema.css`
L'header non ha il menu	menu indicato per numero, o nessun menu con quella posizione	`"menu": "posizione:principale"`
Header e footer nuovi non arrivano dopo l'aggiornamento del tema	il CMS riusa i modelli creati la prima volta	cestina "Header Nome" e "Footer Nome", poi **Usa questo tema**
Avviso rosso "si è rotto"	errore nel PHP del tema	leggi l'errore nell'avviso e in `logs/php-errori.log`, correggi, ricarica con Sostituisci, Riprova ad attivarlo
Pagina bianca o errore 500 una volta	errore fatale nel PHP del tema	come sopra; dalla richiesta dopo il sito torna all'aspetto di base da solo
Le immagini del CSS non si vedono con il CMS in una cartella	percorso assoluto `/temi/...` nel CSS	percorso relativo: `url(img/foto.jpg)`

14. Riferimento rapido

```
temi/<nome>/                  nome: [a-z][a-z0-9-]{1,39}
  tema.json                   nome, versione, autore, descrizione, slug, anteprima, css, stile,
modelli, infografiche
  stile.json                  colori, font, tipografia, pulsanti, immagini, campi, layout, sfondo
(sito e css ignorati)
  modelli.json                {"header": albero, "footer": albero}   menu: "posizione:principale"
  tema.css                    dopo sito.css, prima dello Stile del sito; :root vince su
:where(:root)
  funzioni.php                return ['filtri' => [...], 'eventi' => [...]];   solo sito pubblico
  viste/<vista>.php           return '<html...>';   documento home articolo pagina blog archivio
cerca 404

Colori nei widget:            #rrggbb  rgba()  globale:<id>  ig:<nome>
Anteprima:                    /?preview=1&tema=<nome>      senza tema: /?preview=1&tema=base
Zip:                          una cartella; json css png jpg webp gif avif woff woff2 ttf otf txt md
php
Firma (PHP):                  PACCHETTO_CHIAVE=... python3 tools/pacchetto.py --rifirma tema.zip
Contrasto minimo:             4,5 a 1
```

Il motore dei temi sta in `includes/temi.php`: è commentato riga per riga, per chi vuole guardarci dentro.