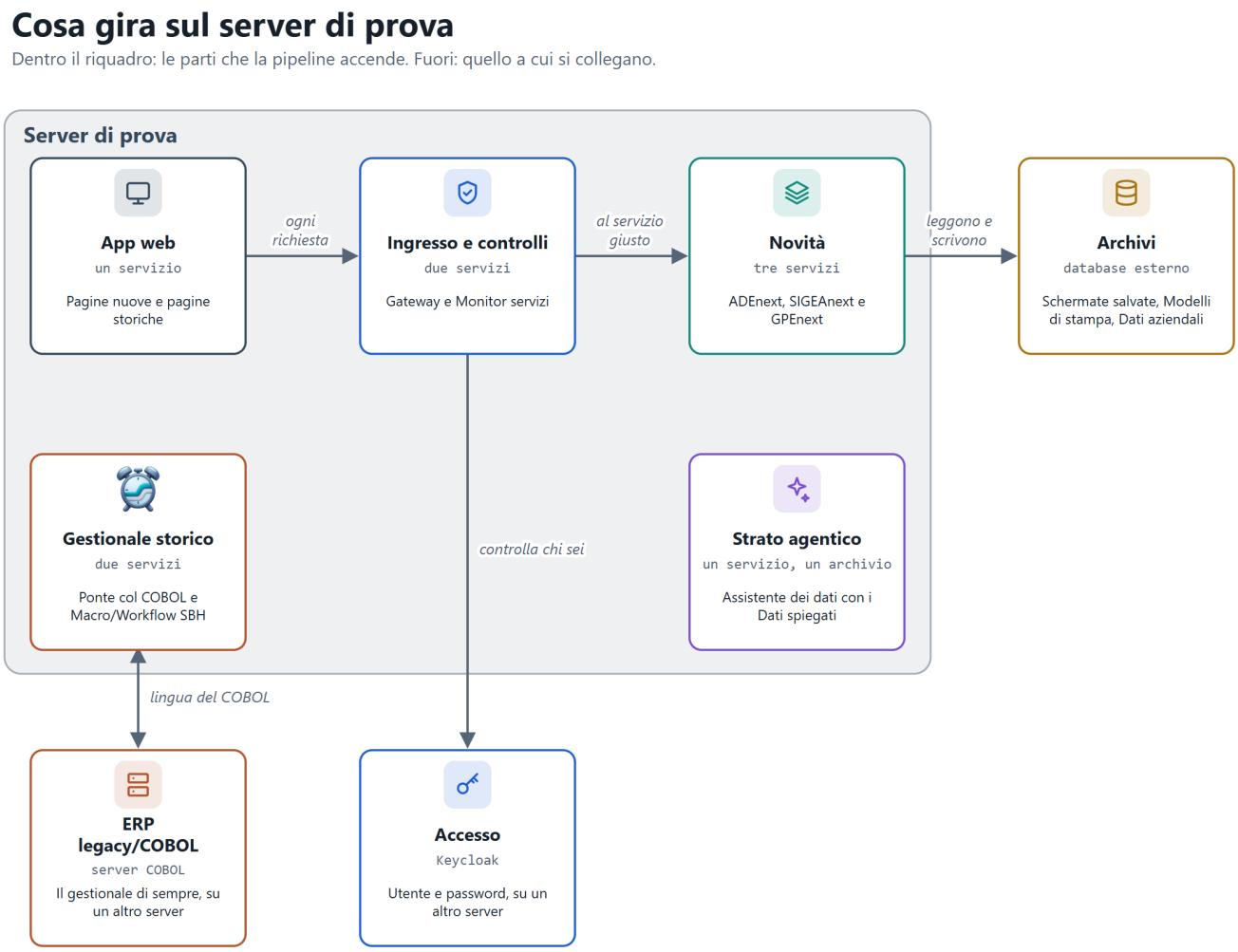
JM2 - Il server di prova

- JM2.1 - Cosa gira e dove
- JM2.2 - Il file delle impostazioni
- JM2.3 - Accendere un servizio nuovo

JM2.1 - Cosa gira e dove

In una frase. Sul server di prova girano l'app web e i servizi di ADSWnext, ognuno nel suo contenitore; archivi, Accesso e gestionale di sempre stanno fuori e ci si collegano.

Il server di prova è il posto dove si guarda il lavoro finito prima di portarlo altrove. Sapere cosa c'è sopra serve a due cose: capire cosa cambia con un aggiornamento e sapere dove guardare quando una pagina non risponde.



Dentro il server di prova i servizi per gruppo; fuori gli archivi, l'Accesso e l'ERP legacy/COBOL

Ogni servizio gira in un **contenitore**: una scatola chiusa con dentro il programma e tutto ciò che gli serve. Le scatole si accendono, si spengono e si sostituiscono una per una, senza toccare le altre.

Gruppo della mappa	Cosa gira sul server

App web	Una sola pagina con due parti: pagine nuove e pagine storiche
Ingresso e controlli	Gateway e Monitor servizi
Novità	ADEnext (con Analytics), SIGEAnext, GPEnext con il suo servizio di impaginazione
Gestionale storico	Ponte col COBOL e Macro/Workflow SBH
Strato agentico	Assistente dei dati e il suo archivio, i Dati spiegati

Cosa sta fuori. Gli archivi delle Schermate salvate, dei Modelli di stampa e dei Dati aziendali sono su un database esterno. L'Accesso è su un altro server. L'ERP legacy/COBOL è il gestionale di sempre, raggiunto dal Ponte col COBOL.

Chi si vede da fuori. Dal tuo PC raggiungi solo l'app web e il Gateway. Gli altri servizi rispondono solo dentro il server: in mezzo c'è sempre il Gateway.

Cosa resta dopo un aggiornamento. I contenitori si sostituiscono, i dati no. Quello che i servizi tengono su file, come le viste delle tabelle salvate dagli utenti, sta in spazi separati che sopravvivono al cambio del contenitore.

> **Nota.** GPEnext e il suo servizio di impaginazione sono entrati fra i servizi accesi dalla pipeline a novembre 2026. L'Assistente dei dati gira, ma la chat si accende con un interruttore a parte: vedi il libro JJ - Strato agentico.

> **Per chi sviluppa.**
> • Servizi e profili: `devops/deploy/docker-compose.deploy.yml`. Ogni servizio ha un profilo; quelli accesi li decide la pipeline (`PROFILI_CLASSICO`), non il file delle impostazioni.
> • Le porte dei servizi interni sono pubblicate solo sul server stesso (`PORTE_INTERNE_BIND`). Davanti all'app web e al Gateway va messo un proxy con HTTPS.
> • Dati su file: volumi `sbh_data`, `sigea_data`, `sbh_flow_data`, `gpe_output`, più quelli dell'Assistente dei dati. Spegnere lo stack con l'opzione che rimuove i volumi li cancella.
> • Nella pipeline ci sono anche lavori per prodotti che non fanno parte di questa raccolta.

JM2.2 - Il file delle impostazioni

In una frase. Tutto ciò che cambia da un'installazione all'altra, comprese le password, sta in un solo file sul server, fuori dal codice.

Il codice è uguale per tutti. Quello che cambia da un server all'altro sono gli indirizzi, le cartelle e le password. Se stessero nel codice, chiunque lo legga li vedrebbe, e ogni installazione vorrebbe una copia diversa del programma. Per questo stanno in un file a parte, che vive solo sul server.

Nel codice c'è il **modello** del file: l'elenco dei nomi, con le spiegazioni e senza i valori. Sul server c'è la **copia compilata**, con i valori veri. La pipeline e i servizi leggono quella.

Dentro trovi, per gruppi:

Cosa dice	Esempi di nomi
Dove sono gli archivi e con quale utente	`ADE_DB_URL`, `GPE_DB_URL`, `SIGEA_DB_URL`, con i loro `_USER` e `_PASSWORD`
Come si controlla chi entra	`AUTH_ENABLED`, `KEYCLOAK_BASE_URL`, `SSO_REDIRECT_ORIGINS`
Dove sta il gestionale di sempre	`SBH_HOST`, `SBH_PORT`, `SBH_ALLOWED_ORIGINS`
Le cartelle dei dati per le stampe	`GPE_DATA_DIR`, `GPE_FONT_DIR`, `GPE_CATALOGO_HOST_DIR`
L'Assistente dei dati	`SIGEAGENT_ENABLED` e le sue password
Le porte da cui si entra	`FRONTEND_PORT`, `GATEWAY_PORT`

Tre regole valgono sempre.

- **Nel modello un valore vuoto è un segreto.** Va chiesto a chi gestisce il server e scritto solo nella copia sul server. Nel modello non si scrive mai un valore vero.
- **Alcune righe sono obbligatorie.** `AUTH_ENABLED` va scritta, con `true` o `false`: se manca, ogni comando si ferma e dice cosa manca. Spegnere il controllo di chi entra deve essere una scelta scritta, non una dimenticanza.
- **Una riga cambiata vale dal deploy successivo.** Il servizio che la legge viene riacceso con il valore nuovo. Gli altri restano accesi.

> **Attenzione.** Questo file non entra mai nel codice, in una chat o in un documento. Se serve mostrarlo a qualcuno, si mostrano i nomi, non i valori.

> **Per chi sviluppa.**
> • Il modello è `devops/deploy/env.simil-prod.example`. Il registro di tutte le variabili, con il significato e il valore predefinito, è `docs/devops/ENVIRONMENT.md`: ha anche l'elenco dei

controlli da fare prima di un deploy.
• Quali servizi si accendono non sta in questo file: lo decide la pipeline. Una variabile d'ambiente del lavoro vince sul file.
• Una variabile nuova si aggiunge insieme: al modello, al registro e agli avvisi (vedi Avvisi per chi aggiorna).

JM2.3 - Accendere un servizio nuovo

In una frase. Un servizio nuovo si aggiunge con poche righe e senza toccare gli altri; prima del deploy, però, sul server deve esserci già quello che gli serve per partire.

Aggiungere un servizio non vuol dire rifare l'installazione. Ogni servizio è una scatola a sé, con un'etichetta che dice se va acceso. L'esempio più recente è GPEnext, il servizio delle stampe: a novembre 2026 è entrato fra quelli che la pipeline costruisce e accende.

1. **Descrivi il servizio nello stack.** Un blocco che dice da quale cartella del codice si costruisce, quali impostazioni legge e con quale etichetta si accende.
2. **Dillo alla pipeline.** L'etichetta va aggiunta all'elenco dei servizi da costruire e a quello dei servizi da accendere. Da quel momento il servizio ha anche la sua impronta: si ricostruisce solo quando cambia.
3. **Prepara il server, prima del deploy.** È il passo che si dimentica. Per GPEnext servono tre cose:
   - l'archivio dei Modelli di stampa e il suo utente, creati una volta sul database. Le tabelle le crea il servizio da solo al primo avvio;
   - nel file delle impostazioni le tre righe dell'archivio: `GPE_DB_URL`, `GPE_DB_USER`, `GPE_DB_PASSWORD`;
   - le cartelle dei dati GPE sul server, leggibili dal servizio: dati, caratteri di stampa, catalogo dei modelli di base.
4. **Dai una rotta al Gateway.** Senza, le pagine non sanno come raggiungerlo: vedi il libro JC - Gateway e Monitor servizi, pagina «JC4.1 - Aggiungere una rotta».
5. **Fai il deploy e controlla.** Il servizio nuovo deve comparire acceso e in salute: vedi Controllare che sia andato tutto bene.

> **Attenzione.** Se le righe dell'archivio mancano, GPEnext non parte e continua a riprovare a vuoto. Il resto del gestionale lavora come prima: non vanno solo le stampe.

> **Nota.** Per importare i modelli del gestionale storico non basta che GPEnext sia acceso. Nella cartella dei dati GPE devono esserci i vecchi modelli, nella sottocartella `local/model`. Se manca, la finestra di importazione lo dice e non elenca niente: vedi il libro JK - GPEnext, pagina «JK6.1 - Importare i modelli legacy».

Per spegnere un servizio non basta togliere l'etichetta: il deploy non ferma i servizi usciti dall'elenco. Vanno fermati a mano, una volta. I loro dati restano.

>

Per chi sviluppa.

• Blocco del servizio: `devops/deploy/docker-compose.deploy.yml`, con `profiles:`. Pipeline: il profilo in `PROFILI_CLASSICO` e `PROFILI_TUTTI`, e una riga `costruisci` nel lavoro `build:images` di `.gitlab-ci.yml`.

• GPEnext: profilo `gpe-next`, che accende anche il servizio di impaginazione. Cartelle: `GPE_DATA_DIR`, `GPE_FONT_DIR`, `GPE_CATALOGO_HOST_DIR`. Passi completi in `devops/deploy/README.md`, paragrafo 3.7.

• L'Assistente dei dati ha i suoi passi nel paragrafo 3.6: senza le sue password il deploy si ferma prima di toccare qualunque servizio, e dice quale manca.

• Compose e pipeline sono file protetti: li modifica una persona, non un automatismo.