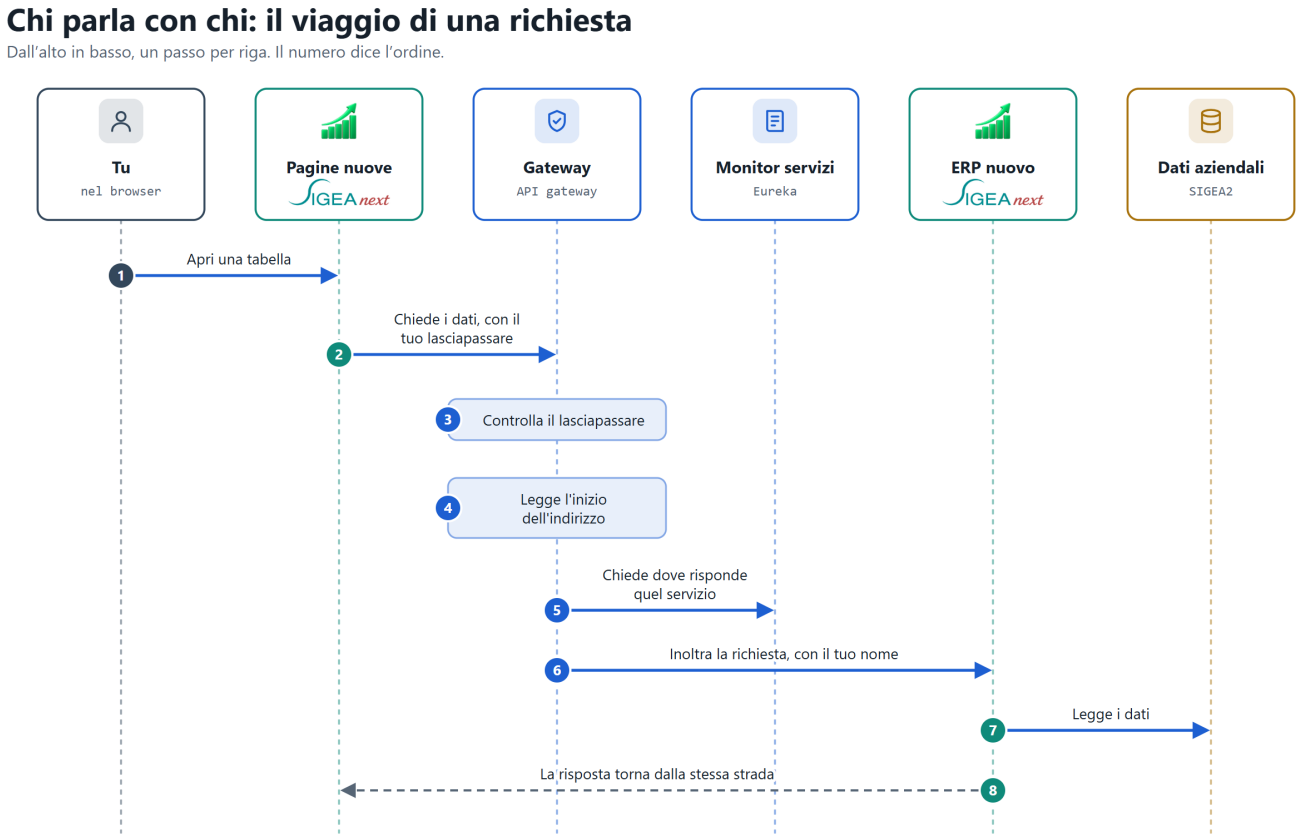
JC2 - Il viaggio di una richiesta

- JC2.1 - Dal browser al servizio giusto
- JC2.2 - Come si controlla chi sei
- JC2.3 - Le due strade: Gateway e Ponte col COBOL

JC2.1 - Dal browser al servizio giusto

In una frase. Come fa una richiesta partita dalla tua pagina ad arrivare proprio al servizio che la sa soddisfare, e a tornare con la risposta.

Ogni volta che apri una tabella, premi **Salva** o chiedi una stampa, la pagina manda una richiesta. Le richieste sono tante e i servizi sono diversi. Il Gateway le smista tutte con una regola sola, semplice da controllare: guarda come comincia l'indirizzo.



Il viaggio di una richiesta: dalla pagina al Gateway, al servizio, e ritorno

Il viaggio, in ordine.

La pagina chiede. La richiesta parte dal browser e porta con sé il tuo lasciapassare, cioè la prova che sei entrato.

Il Gateway controlla. Se il lasciapassare manca o non è valido, il viaggio finisce qui. Come lo controlla è in Come si controlla chi sei.

Il Gateway legge l'inizio dell'indirizzo. Ogni servizio ha il suo:

L'indirizzo comincia con	La richiesta va a	Per fare cosa
`/api/ade`	ADEnext	Schermate disegnate, cruscotti di Analytics
`/api/sigea`	SIGEAnext	Dati del gestionale
`/api/gpe`	GPEnext	Stampe e modelli
`/api/sigeagent`	Assistente dei dati	Domande sui dati (solo dove è acceso)

Chiede al Monitor servizi dove sta quel servizio. Il Gateway conosce il nome del servizio, non il suo indirizzo. L'indirizzo lo prende dall'elenco.

Inoltra. Toglie dall'indirizzo il pezzo iniziale, che serviva solo a lui, scrive sulla richiesta il tuo nome utente e la consegna.

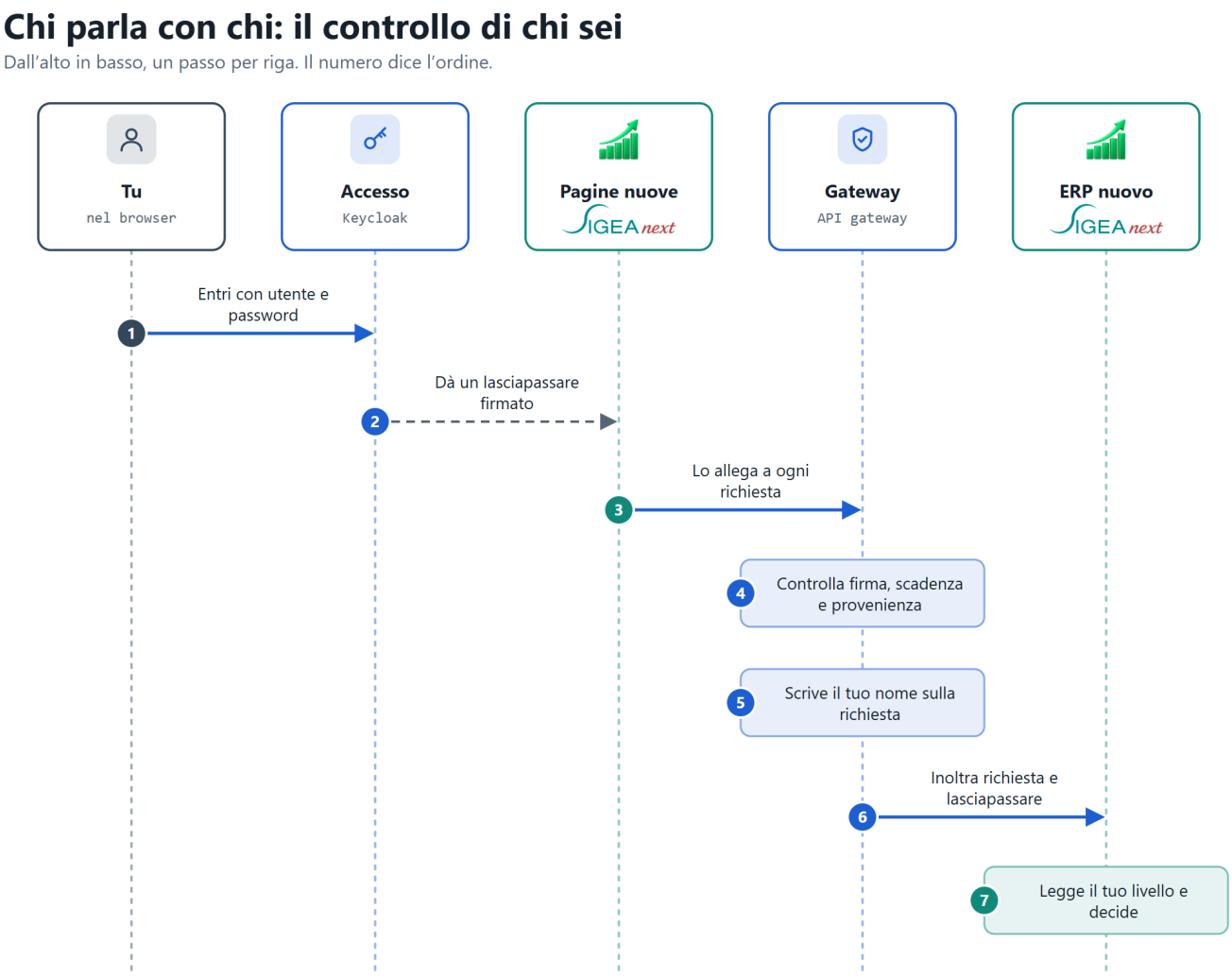
La risposta torna dalla stessa strada. Il servizio risponde al Gateway, il Gateway alla pagina.

> **Per chi sviluppa.**
> • Le rotte sono in `backend/middleware/it_gsba_api_gateway/src/main/java/it/gsba/api_gateway/config/GatewayConfig.java`. Ognuna ha `stripPrefix(2)` e il filtro `AuthenticationFilter`.
> • La destinazione è `lb://NOME-DEL-SERVIZIO`: il nome con cui il servizio si è iscritto al Monitor servizi.
> • L'Assistente dei dati fa eccezione: ha un indirizzo fisso e la sua rotta esiste solo con `SIGEAGENT_ENABLED=true`.
> • Davanti al Gateway c'è il server web dell'app (`frontend/docker/nginx.conf`), che gli passa tutto ciò che comincia per `/api/`. Le richieste lunghe possono durare fino a dieci minuti.

JC2.2 - Come si controlla chi sei

In una frase. Il Gateway non ti chiede la password: controlla il lasciapassare che hai ricevuto entrando, e passa ai servizi il tuo nome.

Utente e password li scrivi una volta sola, nella pagina dell'Accesso. In cambio ricevi un **lasciapassare**: un piccolo documento firmato che dice chi sei e che livello hai, e che scade dopo poco. Da quel momento la pagina lo allega a ogni richiesta. La password non viaggia più.



Il controllo di chi sei: l'Accesso firma il lasciapassare, il Gateway lo verifica, il servizio legge il livello

Per ogni richiesta il Gateway controlla quattro cose:

- che la **firma** sia quella dell'Accesso, quindi che il lasciapassare non sia stato fabbricato o ritoccato;
- che non sia **scaduto**;
- che venga dall'**Accesso di questa installazione** e non da un altro;
- che l'abbia chiesto un **programma ammesso**, cioè l'app web o uno dei servizi.

Se una sola non torna, la richiesta si ferma con l'errore 401 e nessun servizio la vede.

Se tutto torna, il Gateway scrive sulla richiesta il tuo **nome utente** e la inoltra insieme al lasciapassare. Prima però cancella qualunque nome fosse già scritto: così nessuno può presentarsi ai servizi con il nome di un altro.

Il **livello** (lettore, base, avanzato, staff) sta dentro il lasciapassare. Lo legge il servizio che riceve la richiesta, e decide lui se quella cosa la puoi fare. Se non puoi, risponde con l'errore 403. I livelli sono spiegati in Lettore, base, avanzato, staff (libro «JB - Accesso e permessi», pagina «JB3.1 - Lettore, base, avanzato, staff»).

> **Nota.** Poche cose passano senza lasciapassare, perché chi le chiede non può averlo: il controllo «sei acceso?» che il server fa ai servizi, le immagini dentro una stampa, la procedura di prima installazione.

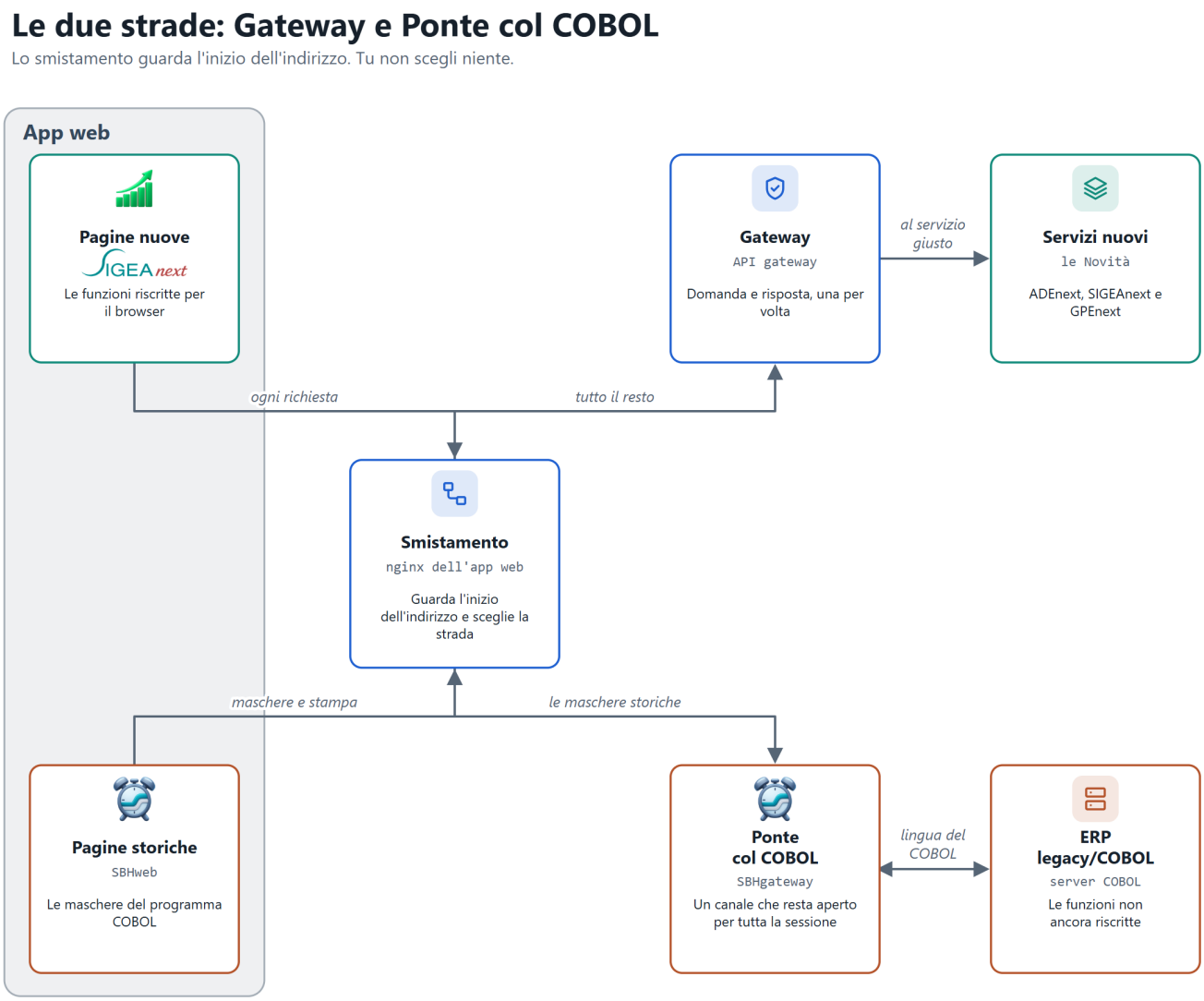
> **Per chi sviluppa.**
> • Il controllo è in `SecurityConfig` e `SsoConfig` del Gateway: firma verificata con le chiavi pubbliche dell'Accesso (JWKS), poi scadenza, emittente e programma che ha chiesto il lasciapassare (`SSO_CLIENT_AMMESSI`).
> • `AuthenticationFilter` scrive `X-User-Id` con il nome utente del lasciapassare e toglie sempre quello arrivato dal client.
> • L'interruttore è `AUTH_ENABLED`. Spento serve solo in sviluppo: il Gateway lascia passare tutto e scrive un utente di comodo. Se è acceso ma mancano gli indirizzi dell'Accesso, il Gateway rifiuta ogni lasciapassare.
> • Anche i servizi verificano il lasciapassare per conto loro: una richiesta che non passa dal Gateway non entra.

> **Per approfondire.**
> • Integrazione Autenticazione JWT Angular + API Gateway: `docs/guide/sviluppo/GUIDA-AUTENTICAZIONE-JWT.md` (descrive una versione precedente, in cui si entrava dal Gateway: oggi si entra dall'Accesso)
> • Il disegno di oggi: `docs/devops/auth/AUTENTICAZIONE.md`

JC2.3 - Le due strade: Gateway e Ponte col COBOL

In una frase. Le pagine nuove passano dal Gateway, le maschere storiche dal Ponte col COBOL: la strada la sceglie l'app, non tu.

L'app web è una sola, ma dentro ha due parti. Le **pagine nuove** parlano con i servizi nuovi: fanno una domanda, ricevono una risposta, fine. Le **pagine storiche** di SBHweb parlano con il programma COBOL, che ragiona in un altro modo: tiene aperta una conversazione per tutta la sessione, maschera dopo maschera. Sono due modi di lavorare diversi e servono due strade.



Le due strade: lo smistamento manda le maschere storiche al Ponte e tutto il resto al Gateway

La strada del Gateway. La prendono tutte le richieste delle pagine nuove. Ognuna viene controllata e consegnata al servizio giusto, come racconta Dal browser al servizio giusto.

La strada del Ponte col COBOL. La prendono le maschere storiche. Fra la pagina e il Ponte si apre un canale che resta aperto per ore, anche quando non scrivi niente. Il Ponte traduce fra il browser e il programma COBOL. Come funziona è in Come funziona il Ponte col COBOL (libro «JH - SBHweb», pagina «JH1.2 - Come funziona il Ponte col COBOL»).

Chi sceglie la strada. Il server web dell'app, guardando l'inizio dell'indirizzo. Le richieste delle maschere storiche vanno al Ponte. Tutte le altre vanno al Gateway.

L'incrocio: la stampa in PDF. Quando da una maschera storica scegli l'uscita `GPE_NEXT_PDF`, il programma COBOL prepara solo i dati. Arrivano alla pagina dal Ponte. Poi la pagina li manda a GPEnext, e lì prende la strada del Gateway, con il tuo lasciapassare come ogni altra richiesta. È il motivo per cui le stampe nuove e quelle storiche usano gli stessi modelli.

> **Cosa ci guadagni.**
> • Le funzioni storiche si usano nel browser senza aspettare che siano riscritte.
> • Quando una funzione viene riscritta cambia strada da sola: tu apri la stessa app.
> • Le due strade non dipendono l'una dall'altra: le maschere storiche non passano dal Gateway, le pagine nuove non passano dal Ponte.

> **Per chi sviluppa.**
> • Lo smistamento è in `frontend/docker/nginx.conf`: `/api/sbh/` va al Ponte (`backend/middleware/it_gsba_sbh_gateway`, porta 5086) con un WebSocket; il resto di `/api/` va al Gateway.
> • Il Ponte non ha una rotta nel Gateway e non è iscritto al Monitor servizi.
> • La stampa da SBHweb chiama il Gateway su `/api/gpe/print/`.
> • Le chiamate interne al Ponte (`/api/sbh/interno/`) dal browser non esistono: il server web risponde 404.