**Tecnologie + Pattern usati**

**📌 Design Pattern che useremo:**

* ✅ **Strategy** → per gestire azioni diverse in base al tipo di utente (es. autorizzazioni o azioni specifiche)
* ✅ **Factory Method** → per creare istanze dei vari tipi di utente o venditore

--------------------------------------------------------------------------------------------------------------

Vuoi ora che ti configuro anche **Spring Security per validare le richieste con JWT** alle API?

Ottima domanda! 💡 Te lo spiego subito in modo semplice e pratico 👇

**🔐 Quando ti ho detto:**

**"Vuoi che ti configuro anche Spring Security per validare le richieste con JWT alle API?"**

Intendevo dire:

**Attivare una sicurezza completa sulle API REST** della tua applicazione, in modo che **solo gli utenti autenticati tramite token JWT** possano accedere ad alcune rotte o funzionalità.

**📦 Facciamo un esempio:**

**🔴 Attualmente (senza sicurezza)**

Chiunque può:

* accedere a /marketplace, /mappa, ecc.
* fare chiamate REST (tipo GET /prodotti) **senza autenticarsi**

**✅ Con Spring Security + JWT abilitati**

* L'utente fa login → riceve un **token JWT**
* Ogni richiesta alle API deve contenere quel token nella **header Authorization**
* Se il token **è valido** → l'utente può accedere
* Se il token **è mancante o non valido** → riceve errore 401 Unauthorized

**🧱 Esempio concreto**

**Richiesta API (protetta):**

sql

CopiaModifica

GET /api/prodotti

Authorization: Bearer <TOKEN\_JWT>

👉 Senza token valido = accesso negato

**🔐 Perché serve?**

È utile per:

* Proteggere **funzionalità sensibili** (es. caricare prodotti, modificare dati)
* Distinguere i **permessi in base al ruolo** (PRODUTTORE, CURATORE, GESTORE, ecc.)
* Offrire un accesso sicuro da **frontend moderni (React, Angular)** o **app mobile**

**🎯 In breve:**

Attivare Spring Security con JWT significa **bloccare l'accesso alle API finché l'utente non è autenticato tramite token.**

Se ti interessa, posso configurarti:

* Il **filtro JWT** per intercettare le richieste
* La **classe SecurityConfig**
* Le **regole di accesso per ruolo**

Fammi sapere se vuoi procedere con questo, oppure se vuoi dedicarti prima ad altre parti della piattaforma (es. marketplace, prodotti, ecc.).

--------------------------------------------------------------------------------------------------------------

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGtnD4bh6hoNHHMsfkUUg4Km2Cd+N0r8YuoYjZCbh6p/ [torquati79@yahoo.it](mailto:torquati79@yahoo.it)

--------------------------------------------------------------------------------------------------------------

Ottima domanda! 👇

Ora che hai **due bottoni di login**:

**🔘 1. Login**

→ Fa il login **senza token JWT**, quindi semplice validazione su username, password e ruolo  
→ Va bene se non hai ancora attivato Spring Security e vuoi solo "navigare" o testare funzionalità basic

**🔘 2. Login con Token**

→ Fa il login **con JWT**  
→ Genera un token e ti porta su una pagina "Benvenuto {nome}, {ruolo}"  
→ È pronto per un'integrazione futura con API sicure

**✳️ Quale usare?**

**Se vuoi:**

✅ Semplicità, testare form base, navigare l'app senza autenticazione avanzata → **usa Login**

🔐 Iniziare ad avvicinarti alla sicurezza vera → **usa Login con Token**

**📌 Consiglio pratico**

Per ora ti consiglio:

💡 **Mantieni entrambi i login attivi**, ma usa:

* Login per lo sviluppo di pagine, flussi base, CRUD
* Login con Token per quando inizierai a proteggere le API o fare richieste AJAX con toke

-------------------------------------------------------------------------------------------------------------

* Ricorda di inserire questo nel campo JDBC URL della console:
* bash
* CopiaModifica
* jdbc:h2:file:./data/filiera-db

-------------------------------------------------------------------------------------------------------------

| **Pattern** | **Dove è stato usato** | **Perché conta?** |
| --- | --- | --- |
| Factory Method | UtenteAutenticatoForm.toUtente() | Istanzia oggetti diversi in base al ruolo |
| Template Method | UtenteAutenticato, Venditore (abstract) | Definisce una struttura riusabile per i sottotipi |