🗒️ VADEMECUM PER LA DOCUMENTAZIONE DI UN PROGETTO SOFTWARE

1. 📌 Introduzione

✅ Cosa includere:

- **Descrizione del sistema**: dominio applicativo, problema risolto
- **Motivazioni delle tecnologie scelte**: non solo cosa, ma perché
- **Obiettivi e ambito**: cosa fa il sistema e cosa non fa
- **Sfide affrontate**: problemi tecnici rilevanti

✍️ Esempio:

> “LibraryAPI digitalizza la gestione di una biblioteca universitaria. Scelto FastAPI per performance asincrone e documentazione automatica. Non gestisce multe o notifiche email.”

---

2. 🏗️ Architettura del Progetto

✅ Struttura da descrivere:

- **Organizzazione modulare**: separazione per domini (es. `books.py`, `users.py`)
- **Principio di Responsabilità Singola**
- **Gestione delle dipendenze**: uso di `Depends()` con FastAPI

✍️ Consiglio:

Fornire uno schema a directory con spiegazione dei moduli e codice di esempio.

---

3. 🔁 Pattern Architetturali e Principi di Design

✅ Da includere:

- **Repository Pattern** per accesso ai dati
- **API-First Design** con annotazioni OpenAPI
- **Principi SOLID**: spiegare come sono stati rispettati (SRP, OCP, ecc.)

🧠 Suggerimento:

Usa esempi di codice per mostrare estensibilità senza modificare moduli esistenti.

---

4. 🗃️ Design del Database

✅ Aspetti chiave:

- **Schema relazionale** ben normalizzato
- **Vincoli di integrità** (CHECK, FOREIGN KEY)
- **Trigger per automazioni**
- **Indici per ottimizzazione query**

🧱 Esempio utile:

> “Trigger che decrementa `available_copies` dopo un prestito.”

5. 🔐 Sicurezza

✅ Coprire:

- **SQL Injection**: uso di prepared statements
- **Gestione password**: hash con `bcrypt` e salt
- **Autenticazione JWT**
- **Rate Limiting**
- **Validazione input con Pydantic**

🧰 Best Practice:

Includere snippet sia del codice sicuro che di quello vulnerabile (per confronto).

---

6. 📡 API Design

✅ Best practices:

- **Struttura RESTful** chiara e coerente
- **Gestione centralizzata degli errori**
- **Documentazione automatica con esempi**
- **Risposte coerenti e informative**

---

7. 🧪 Testing e Qualità del Codice

✅ Test da documentare:

- **Test di sicurezza**
- **Test di logica applicativa**
- **Test di integrazione**
- **Uso di `pytest`, `httpx`, mocking, ecc.**

🛠️ Consiglio:

Scrivi esempi di test con `assert`, spiegando cosa verificano.

---

8. 🚀 Deployment e Configurazione

✅ Punti da trattare:

- **Configurazioni via environment variables**
- **Inizializzazione automatica del DB**
- **Pratiche sicure per ambienti di produzione**
- **Backup e monitoraggio**

---

9. 📊 Valutazione Critica

✅ Contenuti suggeriti:

- **Obiettivi raggiunti**
- **Punti di forza dell'architettura**
- **Obiettivi non raggiunti (motivazioni)**
- **Possibili miglioramenti futuri**

---

10. 📚 Librerie e Risorse Tecniche

✅ Cosa elencare:

- Framework e tool usati (es. FastAPI, SQLite, bcrypt)
- Link a documentazioni ufficiali
- Tutorial consigliati
- Esempio di `requirements.txt`

📄 Template finale (struttura consigliata)

```
1. Introduzione
2. Architettura
3. Pattern e Design
4. Database
5. Sicurezza
6. API e Interfacce
7. Testing
8. Deployment
9. Conclusioni
10. Risorse Tecniche
```