**🔹 Všeobecné pravidlá pre logy**

1. **Štruktúrovanosť**
   * Logy by mali byť **strojovo čitateľné** (napr. JSON, XML, CSV), nie len voľný text.
   * Vhodné na parsing, filtrovanie, vizualizáciu.
2. **Konzistentný formát**
   * Rovnaké poradie a názvy polí (napr. timestamp, level, message, user\_id, session\_id, error\_code).
   * **Časová pečiatka** v štandarde ako ISO 8601 (napr. 2025-07-21T10:23:45Z).
3. **Jasné úrovne logovania**
   * DEBUG, INFO, WARNING, ERROR, CRITICAL
   * Vyberaj primeranú úroveň – nech nie sú všetky chyby len INFO alebo všetko ERROR.
4. **Bez citlivých údajov**
   * Neloguj heslá, celé rodné čísla, čísla kariet, adresy a pod.
   * Maskuj alebo anonymizuj citlivé dáta (\*\*\*\*, hash()).
5. **Obsahujúce kontext**
   * Kto akciu vykonal (user/session/token)
   * Kde sa stala (modul, služba)
   * Čo sa stalo (popis + vstupné hodnoty, ak sú bezpečné)
6. **Jednoznačnosť a čitateľnosť**
   * Každý záznam má byť pochopiteľný bez potreby čítať celý stacktrace.
   * Vyhýbaj sa vnútorne nezmyselným hláškam typu „unexpected error“ bez detailov.

**🔹 Pravidlá pre auditné záznamy**

1. **Nezmazateľnosť a nemennosť**
   * Auditný záznam by sa **nemaže**, ale **dopĺňa** (princíp nemenného záznamu).
   * Ak sa opravuje záznam, **pôvodný stav sa zachováva**.
2. **Zaznamenáva sa:**
   * **Kto** – používateľ, rola, systém
   * **Čo** – zmena (napr. stav: "schválené" → "zamietnuté")
   * **Kedy** – presná časová pečiatka
   * **Odkiaľ** – IP adresa, prehliadač, API klient
3. **Čitateľnosť a auditovateľnosť**
   * Auditný záznam má byť ľahko dohľadateľný v prípade kontroly (GDPR, bezpečnostný incident).
   * Často je viazaný na konkrétny **objekt** alebo **entitu** (napr. objednávka č. 1234).
4. **Neslúži na ladenie**
   * Audit je právna/zodpovednostná stopa, nie debug nástroj.

**📌 Zhrnutie**

| **Prvok** | **Logy** | **Auditné záznamy** |
| --- | --- | --- |
| Účel | Debug, monitoring, incidenty | Zodpovednosť, sledovanie zmien |
| Zmeniteľné? | Áno (v princípe) | Nie (nemenné) |
| Granularita | Technická / detailná | Biznisovo významná |
| Frekvencia | Vysoká | Nízka až stredná |
| Formát | Strojovo čitateľný výhodou | Formálny, konzistentný |

**Stacktrace** (alebo „výpis zásobníka volaní“) je technický výpis, ktorý ukazuje **reťazec volaní funkcií**, ktoré viedli k chybe alebo výnimke v programe.

**🔧 Čo to presne znamená?**

Keď program zlyhá (napr. spadne na chybe), operačný systém alebo runtime (napr. Python, Java, .NET) zobrazí **stacktrace** – zoznam funkcií, ktoré sa volali predtým, než chyba nastala.

**🧠 Prečo je stacktrace užitočný?**

Pomáha vývojárovi alebo testerovi zistiť:

* **kde presne** v kóde sa chyba stala (riadok, súbor, trieda),
* **čo sa volalo predtým** – teda ako sa program dostal do chybového stavu,
* **aký typ chyby** nastal (napr. NullPointerException, IndexError, TypeError...).

**📌 Príklad – jednoduchý stacktrace v Pythone:**

Python

```

KopírovaťUpraviť

Traceback (most recent call last):

File "main.py", line 10, in <module>

divide(4, 0)

File "main.py", line 6, in divide

return a / b

ZeroDivisionError: division by zero

```

**📋 Vysvetlenie:**

* Funkcia divide bola volaná na riadku 10 súboru main.py
* V tele tejto funkcie (riadok 6) sa vykonalo a / b, kde b bolo 0
* Program zlyhal na chybe ZeroDivisionError: division by zero

**🛠️ Stacktrace môžeš nájsť:**

* v **konzolovom výstupe** (ak testuješ lokálne),
* v **logoch** (produkčných alebo testovacích),
* vo **výstupoch testovacích nástrojov** (napr. Selenium, pytest, JUnit),
* v **chybných reportoch** alebo bug trackeri (napr. Jira).