Дигитален идентитет со користење на Verifiable credentials

Идеи-што би избиладале

- „мини-екосистем“ со **Issuer–Holder–Verifier** и VC/VP токени потпишани со криптографски клучеви. W3C VC Data Model опишува токму ваков 3-страничен модел (issuer, holder, verifier), каде holder може да креира verifiable presentation и да ја сподели со verifier.
- Целта е да направиме прототип што издава Verifiable Credential (пр. „Student ID“, „Enrollment“, „Employee“) и потоа го верификува, а не само да складира JSON. VC Data Model дефинира VC како сет на „tamper-evident claims“ + метаподатоци што криптографски докажуваат кој го издал.
- Функционалности:

1. Issuer сервис: генерира клуч, издава VC (како JWT/JWS), и објавува јавен клуч (JWKS endpoint).
2. Holder “wallet”: прима VC, го чува локално (JSON/SQLite), и (опционално) прави VP за да одговори на „challenge“ од verifier.
3. Verifier сервис: прима VC/VP и проверува потпис, валидност, тип, nonce/challenge, и (опционално) статус/ревокација.

Архитектура (Flask)

1. Python/Flask со „Securing VCs using JOSE“ (JWS/JWT), бидејќи W3C има Recommendation што специфицира како да се обезбедат VC/VP со JOSE/SD-JWT/COSE. Во таа спецификација има и важна безбедносна насока: ако JWT има `"alg": "none"` (без интегритет), имплементациите мора да ги игнорираат тие токени.

Предлог структура (едно repo, 3 Flask апликации или 1 апликација со 3 blueprint-и):

- `issuer/` (порт 5001): `/issue`, `/.well-known/jwks.json`
- `holder/` (порт 5002): `/store`, `/present`
- `verifier/` (порт 5003): `/challenge`, `/verify`