**Bezpečnosť na strane klienta (Frontend)**

Bezpečnostná architektúra klienta (webovej aj mobilnej aplikácie) je navrhnutá tak, aby spĺňala najnovšie odporúčania a štandardy (napr. OWASP Top 10), s dôrazom na ochranu pred bežnými útokmi, bezpečné uchovávanie tokenov a správne spracovanie vstupov.

**1. Autentifikácia a tokeny**

* Autentifikácia používateľov prebieha na základe **JWT tokenov (JSON Web Token)**, ktoré sú vydané backendom po úspešnom prihlásení.
* Token je uložený do **HttpOnly, Secure cookies**:
  + **HttpOnly** zabezpečuje, že JavaScript k tokenu nemá prístup (ochrana pred XSS).
  + **Secure** zabezpečuje prenos len cez HTTPS spojenie.
* Tento prístup eliminuje potrebu pracovať s localStorage alebo sessionStorage, ktoré sú zraniteľné voči XSS útokom.

**2. Ochrana pred Cross-Site Scripting (XSS)**

* Používame **Content Security Policy (CSP)** s prísnou konfiguráciou:
  + *default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; img-src \*; object-src 'none';*
  + Zamedzuje načítanie neautorizovaných skriptov z externých zdrojov.
* Všetky vstupy od používateľa sú **validované a escapované** pred zobrazením:
  + Validácia sa vykonáva cez typovacie schémy (napr. knižnica [zod](https://github.com/colinhacks/zod)) alebo manuálne.
  + Výstupy do DOM-u sú vždy escapované.

**3. Ochrana pred Cross-Site Request Forgery (CSRF)**

* Hoci použitie HttpOnly cookies prirodzene zvyšuje bezpečnosť, CSRF ochranu dopĺňame pomocou tzv. **double-submit tokenu**:
  + Server vydáva **X-CSRF-TOKEN** pri autentifikácii (napr. v cookie alebo response headeri).
  + Frontend (napr. cez **RTK Query** middleware) tento token číta a pridáva ako hlavičku do každej mutujúcej požiadavky (POST/PUT/DELETE).
  + Backend porovnáva hlavičku s očakávanou hodnotou v session/cookie a odmieta neautorizované požiadavky.

**4. Správa stavu a požiadaviek**

* Používame **Redux Toolkit (RTK)** na správu globálneho stavu používateľa (napr. autentifikačný stav, role).
* Na komunikáciu s API používame **RTK Query**:
  + Automaticky spravuje caching, request lifecycle (pending/success/error), a refetching.
  + Umožňuje centrálne pridávať hlavičky (napr. Authorization, X-CSRF-TOKEN) cez baseQuery.

**5. Komunikácia cez HTTPS a HSTS**

* Všetka komunikácia prebieha výhradne cez **HTTPS**.
* **Strict-Transport-Security (HSTS)** hlavička núti prehliadače komunikovať výhradne šifrovane:

*Strict-Transport-Security: max-age=63072000; includeSubDomains; preload*

**6. CORS politika (Cross-Origin Resource Sharing)**

* Na backendovej strane (Spring Boot) je nakonfigurovaná prísna CORS politika:
  + **Povoľujeme len konkrétne frontend domény** (napr. https://app.ubytovanie.sk), žiadne \*.
  + Povoľujeme len bezpečné metódy: GET, POST, PUT, DELETE.
  + Explicitne povoľujeme len potrebné hlavičky (Authorization, X-CSRF-TOKEN, Content-Type).

**7. Session bezpečnosť**

* Každý klientský request je kontrolovaný pomocou tokenu, nie identifikátora session.
* Pri strate platného JWT tokenu alebo CSRF tokenu sa používateľ automaticky odhlasuje.
* K zraniteľným akciám sa nedá dostať bez opätovného overenia identity.