Uitleg in het Nederlands

Hotelbeds Cache API — Beschikbaarheid, Synchronisatie-intervallen & Fallback-logica

Dit document legt uit hoe de **beschikbaarheidslaag** van de Hotelbeds Cache API ontworpen en geïmplementeerd moet worden.

1) Beschikbaarheidslaag Ontwerp:
- Houd per hotel/datum/bezetting een canoniek beschikbaarheidsrecord bij.
- Velden: status (beschikbaar, stop-sell, uitverkocht), allotment, policies (CTA, CTD, minLOS, maxLOS), rate_keys.
- Gebruik Redis keyschema's om beschikbaarheidsobjecten op te slaan en op te vragen.

2) Synchronisatie-intervallen:
- Hoogseizoen (vakanties, juli–augustus, kerst): beschikbaarheid elke 1–2 minuten, prijzen elke 2–5 minuten.
- Midden seizoen: beschikbaarheid elke 5 minuten, prijzen elke 10 minuten.
- Laagseizoen: beschikbaarheid elke 10–15 minuten, prijzen elke 15–30 minuten.
- Dynamische TTL's op basis van seizoenskalender, met overrides indien plotselinge vraag stijgt.

3) Fallback-logica:
- Als een gebruiker zoekt naar een datum zonder beschikbaarheid, zoek de volgende beschikbare datum.
- Toon een bericht: "Niet beschikbaar op de gekozen datum. Eerstvolgende beschikbaarheid vanaf {datum} voor {prijs}."
- Implementatie: controleer achtereenvolgens de volgende dagen tot max horizon (bijv. 30 dagen).

4) Prestaties & Betrouwbaarheid:
- Stale-While-Revalidate: toon direct cache en ververs op de achtergrond.
- Fouttolerantie: bij upstream-fouten cached data blijven tonen.
- Circuit breaker: bij herhaaldelijke fouten naar cache-only modus.
- Telemetrie: monitor beschikbaarheidsfouten, fallback-gebruik en responstijden.

5) QA Checklist:
- Controleer sync-intervallen per seizoen.
- Simuleer storingen om failover te testen.
- Controleer correcte stopcondities in fallback-logica.
- Zorg voor duidelijke UI-indicatie bij fallback-datums.

Explanation in English

Hotelbeds Cache API — Availability, Sync Intervals & Fallback Logic

This document explains how to design and implement the **availability layer** of the Hotelbeds Cache API.

1) Availability Layer Design:
- Maintain a canonical availability record per hotel/date/occupancy.
- Fields: status (available, stop-sell, sold-out), allotment, policies (CTA, CTD, minLOS, maxLOS), rate_keys.
- Use Redis key schemas to store and query availability objects.

2) Sync Intervals:
- High season (holidays, July–August, Christmas): availability every 1–2 minutes, prices every 2–5 minutes.
- Shoulder season: availability every 5 minutes, prices every 10 minutes.
- Low season: availability every 10–15 minutes, prices every 15–30 minutes.
- Apply dynamic TTLs based on a seasonality calendar, with overrides if sudden demand spikes.

3) Fallback Logic:
- If a user searches for a date with no availability, find the next available date.
- Show message: “Not available on your chosen date. Next available from {date} at {price}.”
- Implementation: check sequentially the following days until max horizon (e.g., 30 days).

4) Performance & Reliability:
- Stale-While-Revalidate: serve cache instantly, refresh in background.
- Error tolerance: if upstream fails, continue serving cached data.
- Circuit breaker: switch to cache-only mode after repeated failures.
- Telemetry: monitor availability misses, fallback usage, response times.

5) QA Checklist:
- Verify sync intervals per season.
- Simulate outages to test failover.
- Ensure fallback logic stops at max horizon.
- Confirm UI clearly distinguishes fallback dates.