Uitleg in het Nederlands

Hotelbeds Cache API Integratie – Node.js Developer (Uitleg in het Nederlands)

Dit document beschrijft de vereisten en aanpak voor het bouwen van een snelle en schaalbare integratie met de Hotelbeds Cache API in Node.js.

1) Doel en kernpunten:
- Ontwikkelen van een performant systeem voor hotelzoekopdrachten en detailpagina's.
- Gebruik van MySQL voor gestructureerde opslag en Redis voor caching.
- Scheiding van API's: goedkoopste prijs, matrix details, en statische hoteldata.
- Toepassing van business rules: kortingen, promoties, stop sales, annuleringsvoorwaarden, restricties.
- Toekomstbestendig ontwerp: uitbreidbaar met externe leveranciers, cloud■ready.

2) Taken van de ontwikkelaar:
- Database-integratie: MySQL schema, sync jobs (elk uur), verwerken van cache data (ZIP → DB).
- Businesslogica toepassen: inventory, prijzen, promoties, annuleringen, min/max nachten, check-in/out restricties.
- Node.js API's bouwen:
* Result Page API (goedkoopste prijzen per hotel).
* Detail Page API (matrix met tariefdetails, regels).
* Hotel Static Data API (beschrijvingen, voorzieningen, foto's).
- Filters ondersteunen: hotelnaam, duur, board type, categorie, zone, afstand tot strand/centrum, kindvriendelijkheid, promoties, enz.
- Documentatie via Postman + Swagger.

3) Frontend compatibiliteit:
- Integratie met homepage, resultatenpagina en matrixpagina.
- Oplossen van bestaande integratieproblemen.

4) Tech stack:
- Backend: Node.js (Express/NestJS).
- Database: MySQL.
- Cache: Redis.
- Documentatie: Postman, Swagger/OpenAPI.

5) Deliverables:
- MySQL schema + scripts.
- Sync jobs (elk uur, flexibel interval 60 → 15/30 min).
- Node.js API's (search, matrix, static).
- Documentatie en broncode.
- Geteste frontend integratie.

6) Aanbevelingen:
- Logging en monitoring (Winston, Prometheus, Grafana).
- Tests (unit/integration).
- Cloud deployment (Docker, Kubernetes).
- API Gateway (Kong/Nginx) voor security en throttling.
- Precompute goedkoopste prijs p.p. per hotel.

7) Prijslogica per categorie:
- Citytrips: minimaal 2 nachten (3 dagen), output: "Vanaf €XXX p.p. | 3 dagen / 2 nachten".
- Andere reizen: minimaal 5 nachten, zelfde logica.

- Altijd promoties meerekenen (gratis nachten, kortingen).
- Zoekmachine: voor opgegeven datums exacte prijzen tonen, met fallback bij onbeschikbaarheid.

Conclusie: Het document geeft een complete handleiding voor de architectuur, businesslogica, ontwikkeltaken en technische vereisten voor een schaalbare Hotelbeds Cache API integratie in Node.js.

Explanation in English

Hotelbeds Cache API Integration – Node.js Developer (Explanation in English)

This document outlines the requirements and approach to building a fast and scalable integration with the Hotelbeds Cache API using Node.js.

1) Objective and key highlights:
- Build a high-performance system for hotel search and detail results.
- Use MySQL for structured storage and Redis for caching.
- Separate APIs: cheapest price, matrix details, and static hotel data.
- Apply business logic: discounts, promotions, stop sales, cancellation rules, restrictions.
- Future-proof design: extendable to external suppliers, cloud-ready.

2) Developer responsibilities:
- Database integration: MySQL schema, sync jobs (hourly), process cache data (ZIP → DB).
- Apply business rules: inventory, pricing, promotions, cancellations, min/max nights, check-in/out rules.
- Develop Node.js APIs:
* Result Page API (cheapest prices per hotel).
* Detail Page API (matrix with detailed breakdown and rules).
* Hotel Static Data API (descriptions, amenities, photos).
- Support filters: hotel name, duration, board type, category, zone, distance to beach/city, child facilities, promotions, etc.
- Provide documentation with Postman + Swagger.

3) Frontend compatibility:
- Ensure integration with homepage, results page, and matrix page.
- Fix existing issues.

4) Tech stack:
- Backend: Node.js (Express/NestJS).
- Database: MySQL.
- Cache: Redis.
- Documentation: Postman, Swagger/OpenAPI.

5) Deliverables:
- MySQL schema + scripts.
- Sync jobs (hourly, configurable intervals 60 → 15/30 min).
- Node.js APIs (search, matrix, static).
- Documentation and source code.
- Tested frontend integration.

6) Recommendations:
- Implement logging & monitoring (Winston, Prometheus, Grafana).

- Automated unit/integration tests.
- Cloud deployment (Docker, Kubernetes).
- API Gateway (Kong/Nginx) for caching, throttling, and security.
- Precompute cheapest price per hotel.

7) Pricing logic per category:
- City trips: minimum 2 nights (3 days), output: “From €XXX p.p. | 3 days / 2 nights”.
- Other trips: minimum 5 nights, same logic applies.
- Always include promotions (free nights, discounts).
- Search engine: show exact prices for given dates, with fallback if unavailable.

Conclusion: The document serves as a comprehensive guideline for architecture, business logic, developer tasks, and technical requirements for a scalable Hotelbeds Cache API integration in Node.js.