# Оптимизирана архитектура за Google Places API при 1000 потребителя дневно

Целта е да поддържаме пълна функционалност на Vegan Score Agent – включително 20 обекта дневно на потребител – при прогнозен API разход под $300/месец. Предложеният модел използва кеширане, контрол на обновяването и локално съхранение.

## 1. Архитектура на решението

Потребител → UI → Backend → Проверка в кеш → Ако е нова зона → API call → Кеш → Отговор.

• Кешът е базиран на зона (гео-хеш, напр. 2km²).

• Данните от Google Places (обекти, детайли, снимки) се пазят локално и се обновяват само ако са стари над 7 или 30 дни.

• Функциите се изпълняват чрез Firebase Functions или API Gateway + Cloud Function в GCP.

## 2. Механизми за оптимизация

|  |  |  |
| --- | --- | --- |
| Мярка | Описание | Редукция на API |
| Кеш на зона | Всяка зона (гео-хеш) се кешира до 30 дни | -90% от Place Search |
| Локален snapshot на обекти | Place Details се пазят локално | -95% от Details |
| Снимки/ревюта при първо зареждане | Запазват се и се обновяват периодично | -70% от снимки |
| Месечен лимит в GCP | Hard cap на $300 | Предотвратява изненада |

## 3. Симулиран месечен разход (при кеширане)

• Place Search: 3,000 заявки – $51

• Place Details: 10,000 заявки – $170

• Photos/Reviews: 5,000 заявки – $35

→ Общо: $256/месец

## 4. Предимства на решението

• Разходите се намаляват над 50 пъти (от $12,810 → $256)

• UX не се влошава – кешираните обекти се показват бързо

• Мащабируемо за до 10,000 потребителя при същия модел

• Позволява прогнозируемост и контрол чрез GCP лимити

## 5. Следващи стъпки

1. Изграждане на структурата на кеш по зони (гео-хеш)

2. Firebase или GCP база със запис за: зона, данни, дата на последно обновяване

3. Функция за периодично обновяване (напр. cron, PubSub)

4. Мониторинг и контрол на заявките чрез Cloud Billing