Tokeny, limity i koszty — ściąga dla HyperHuman Club

Prosty, zwięzły materiał dla osób używających Claude, ChatGPT/Codex, Gemini/Antigravity i agentów AI.

Najważniejsza idea

W pracy z AI masz naraz **trzy różne budżety**:

1. **Okno kontekstowe** modelu — ile „widzi” w jednym zadaniu.
2. **Limit użycia / subskrypcji** — ile możesz zrobić w aplikacji.
3. **Koszt API / rate limit** — ile kosztują tokeny i ile zapytań/min.

Mylenie tych trzech rzeczy to najczęstsza przyczyna przepalonej subskrypcji i zaskoczenia kosztami.

---

1. Co to jest token

Token to kawałek tekstu przetwarzany przez model. Nie jest to jedno słowo. Tokenem może być:

- całe krótkie słowo,
- fragment dłuższego słowa,
- znak interpunkcyjny,
- liczba,
- spacja,
- fragment kodu.

W polskim jedno słowo to często kilka tokenów. Do orientacji: dłuższy tekst = więcej tokenów. Dokładną liczbę dla konkretnego modelu najlepiej sprawdzić jego oficjalnym tokenizerem.

---

2. Input, output, historia rozmowy

Input tokens

Wszystko, co model dostaje: twoja wiadomość, historia, instrukcje systemowe, pliki, PDF-y, schematy narzędzi, wyniki wcześniejszych komend.

Output tokens

Odpowiedź modelu. Generowanie danych kosztuje więcej mocy niż ich czytanie.

Historia rozmowy

Wielu agentach przy każdej kolejnej wiadomości model dostaje całą dotychczasową historię rozmowy. Dlatego długa rozmowa z czasem robi się coraz cięższa — to **efekt kuli śnieżnej** (snowball effect / pętla resubmisji kontekstu).

---

3. Okno kontekstowe

Okno kontekstowe to pamięć robocza modelu: instrukcje, historia, pliki, schematy narzędzi, odpowiedź, którą model dopiero wygeneruje.

Przykładowo (orientacyjnie, lipiec 2026):

- Claude Sonnet 5: duże okno kontekstowe (sprawdź aktualne w docs).
- ChatGPT / GPT-5.x: w zależności od wariantu modelu.
- Gemini 3.x Pro: największe okna, rzędu miliona tokenów.

Konkretne liczby zmieniają się z każdą wersją modelu, dlatego w dokumencie zostawiam tylko mechanikę — aktualne limity sprawdź w oficjalnej dokumentacji.

Gdy okno się zapełnia, narzędzie zwykle:

- skraca historię,
- tworzy streszczenie,
- robi compact,
- albo zaczyna gubić najstarsze szczegóły.

4. Compact, summarize, /clear, /rewind

W Claude Code, Codex i Antigravity masz do dyspozycji kilka komend. Nie są to synonimy:

Komenda	Kiedy	Co robi
`/clear` (lub `/new`)	zmiana tematu, koniec etapu, inny moduł kodu	czyści historię, nowe zadanie startuje z pustym kontekstem
`/compact`	długa praca w jednym wątku, okno zbliża się do 60%	zastępuje historię gęstym streszczeniem
`/rewind` (lub `/undo`)	agent się zgubił, zaczął pisać zły kod	cofa stan agenta do wybranego checkpointu i wycina błędne kroki z historii
`/checkpoint`	alias `/rewind` w Claude Code (dokładnie to samo menu)	j.w.

`/rewind` ma więcej niż jedno zastosowanie

`/rewind` to nie tylko „cofnięcie po błędzie kodu”. W Claude Code otwiera interaktywne menu z opcjami:

- **Summarize from here** / **Summarize up to here** — wybrany fragment historii zostaje zastąpiony AI-generated podsumowaniem w oknie (nie zapisuje automatycznie niczego na dysk).
- **Cofanie zmian w plikach** — razem ze stanem rozmowy wraca poprzednia wersja plików.
- **Skrót: dwa razy Esc** przy pustym polu tekstu działa tak samo.

Workflow: 10 tasków, doszedłeś do 5, chcesz czyści restart z podsumowaniem

1. Powiedz agentowi zwykłym promptem:

> Zrób krótkie podsumowanie każdego ukończonego taska (1–10) i zapisz jako `STATE.md` w folderze projektu. Dla każdego taska: co było zrobione, jakie pliki, jaki wynik.

2. Zrób commit `STATE.md` w gicie, jeśli zależy na historii stanu.

3. `/rewind` do checkpointa sprzed taska 6 (lub `/clear` jeśli chcesz totalnie czysto).
4. Przy starcie nowej sesji wklej lub załaduj `STATE.md` jako początkowy kontekst.

Dzięki temu agent od razu wie, że taski 1–5 są zrobione, a 6–10 do zrobienia. Zero powtórzeń.

Wracanie z offtopicu

Zdarza się, że agent poszedł w złą stronę albo Ty poszedłeś. Masz trzy wyjścia:

- **agent generuje błędy, kod nie działa** → `/rewind` do ostatniego dobrego checkpointu, przeformułuj prompt, jedziesz dalej.
- **rozmowa zbacza w dygresje, ale kod jest ok** → `/compact` — streszcza wszystko i zostawia istotny stan.
- **rozmowa poszła w zupełnie inny temat** → `/clear` (twardy reset) albo zacznij nową sesję z plikiem stanu, w którym zostawiłeś istotne fakty.

Różnice między ekosystemami

- **Claude Code:** `/rewind` i `/checkpoint` to synonimy. Wszystkie trzy powyższe ścieżki działają tak samo dla Sonnet 5, Opus 4.8 i Haiku 4.5.
- **Antigravity:** `/rewind` to główna komenda, oficjalny alias to `/undo`.
- **Codex CLI:** nie ma natywnego odpowiednika. `Esc` ucina transkrypt rozmowy, ale nie cofa zmian w plikach — tu trzeba polegać na `git`.

Limit czasu życia cache (TTL) — lipiec 2026

Dostawca	Domyślny TTL	Rozszerzony TTL	Jak włączyć
Anthropic (Claude)	5 minut	1 godzina	`ENABLE_PROMPT_CACHING_1H=1`, ale podwaja koszt zapisu do bufora
OpenAI (ChatGPT / API)	5–10 minut	do 24 godzin	extended caching w organizacjach z wyłączonym ZDR
Google (Gemini)	60 minut	brak górnego limitu	explicit caching z parametrem `ttl`, płatność za przechowywanie

Po przerwie dłuższej niż TTL następne pytanie liczone jest od zera. Dotyczy to zwłaszcza pracy po obiedzie, na drugi dzień, albo po dowolnej dłuższej przerwie.

Czekanie aż okno „samo się zapełni” jest błędem. Proaktywnie streszczaj przy ok. 60% i czyść po zakończonym etapie.

---

5. Dlaczego limity znikają szybciej niż myślisz

- **Długa rozmowa** — każda kolejna wiadomość rozjeżdża kontekst, bo cała historia leci ponownie jako input.
- **Duże pliki / foldery** — 100k tokenów wklejone raz, a potem 20 pytań = 2M tokenów przetworzonych.
- **Retry / pętla błędów** — agent sam się poprawia i potrafi spalić 50–300k tokenów na jednym obiegu.
- **Eksplozja subagentów** — agent może odpalić równoległe subagenty, każdy z własną kopią kontekstu. Przy 50 subagentach rachunek rośnie 50x.
- **Cache miss po przerwie** — TTL (time to live) cache zależy od dostawcy i ustawień: domyślnie 5 minut w Claude, 5–10 minut w OpenAI, 60 minut w Gemini (szczegóły w sekcji o TTL). Dwugodzinna przerwa obiadowa resetuje pamięć podręczną w Claude i OpenAI; w Gemini cache jeszcze działa, ale przekroczenie TTL wyłącza hit. Po przekroczeniu TTL następne pytanie liczone jest od zera.
- **Dużo MCP / dużo narzędzi** — schematy narzędzi też wchodzą do kontekstu.

---

6. Limity aplikacji vs API

Limity w aplikacjach

Dotyczą Claude Desktop, Claude Code, ChatGPT, Codex, Gemini, Antigravity. Liczone jako:

- liczba wiadomości w oknie 3h lub 5h,
- tygodniowy limit użycia,
- limity specyficzne dla modelu albo narzędzia,
- **shared bucket** w Anthropic: web, desktop i CLI dzielą ten sam budżet.

Limity API

Dotyczą deweloperów i integracji. Mierzone jako:

- input / output tokens,
- TPM (tokens per minute) i RPM (requests per minute),
- koszt per milion tokenów,
- limity per model, per tier.

API jest bardziej mierzalne, ale łatwiej wpaść w niespodziewany koszt przy automatyzacji, która utknęła w pętli.

---

7. Porównanie ekosystemów (lipiec 2026)

Anthropic — Claude / Claude Code / Desktop

- **Shared bucket** między web/desktop/CLI — intensywna sesja w CLI obniża limity w przeglądarce.
- Aktualne modele: Claude Sonnet 5, Claude Opus 4.8, Claude Haiku 4.5.
- Plany: Pro, Max 5x, Max 20x.
- Plany Max mają opcję **usage credits** po wyczerpaniu pakietu (dopłata wg stawek API zamiast twardej blokady), ale **to nie dzieje się automatycznie** — trzeba to ręcznie włączyć w ustawieniach konta (dodać metodę płatności i aktywować dodatkowe zużycie). Bez tej konfiguracji po wyczerpaniu limitu dostajesz zwykłą blokadę do resetu okna.
- Komendy: `/cost`, `/stats`, `/context`; można włączyć status line.
- Extended thinking nie jest osobno płatny, ale generuje tokeny wyjściowe (droższe 5x niż wejściowe).

OpenAI — ChatGPT / Codex

- Codex **dzieli** subskrypcję z ChatGPT, ale ma własne limity specyficzne dla CLI i chmury.
- Aktualne modele: GPT-5 / GPT-5.4 / GPT-5.5 (najnowsza generacja) w rodzinie GPT-5.x.
- Okna limitów są tu mieszane: ChatGPT Plus/Go w modelu GPT-5.5 używa okna 3h, Codex CLI używa okna 5h. Free używa okna 5h. Limity są dynamiczne — sprawdź aktualny stan w panelu konta.
- Tracking: komenda `/status` albo dedykowany dashboard w ustawieniach.

Google — Gemini / Antigravity

- **Compute-based quota** zamiast stałej liczby wiadomości. Trudniejsza do przewidzenia.
- Aktualne modele: Gemini 3.x Pro, Gemini Flash.
- UI rate-limit: pojedyncze pytanie tekstowe zjada mało, ale Antigravity z Deep Think na mocnym modelu Gemini potrafi zablokować konto na 24h, a nawet 7 dni.
- Komendy: `/statusline`, `/permissions`.
- Mniejsza kontrola nad eksplozją subagentów niż w Anthropic.

---

8. Jak nie przepalać tokenów i limitów

Przed zadaniem:

- [ ] Czy agent naprawdę potrzebuje całego folderu/pliku, czy tylko wybranego fragmentu?
- [ ] Czy mam mały zakres, jasny cel, źródła, granice i definicję gotowe?
- [ ] Czy wybrałem odpowiedni model, a nie „najmocniejszy z przyzwyczajenia”?
- [ ] Czy podłączyłem tylko potrzebne narzędzia / MCP?

W trakcie:

- [ ] Czy agent idzie w dobrą stronę? Jeśli nie — stop i korekta planu.
- [ ] Czy nie czyta w kółko tych samych plików?
- [ ] Czy odpowiedź nie jest za długa? Ogranicz `max_output_tokens` / długość odpowiedzi.
- [ ] Czy nie utknął w pętli retry? Jeśli tak — `/rewind`.

Po etapie:

- [ ] Czy mam `STATE.md` albo krótki raport stanu?
- [ ] Czy mogę zacząć nową rozmowę z krótkim stanem (zamiast doklejać starą historię)?
- [ ] Czy zrobić review mocniejszym modelem?
- [ ] Czy wynik ma dowody: pliki, testy, logi, linki?

---

9. Wzorzec: tani model robi, mocny model sprawdza

To najlepsza metoda kontroli kosztów przy dużych zadaniach.

1. Tani/szybki model (Haiku 4.5, GPT-5 mini, Gemini 2.5 Flash-Lite) wykonuje pracę: parsowanie, formatowanie, klasyfikacja, proste zmiany.
2. Mocny model (Claude Sonnet 5 / Opus 4.8, GPT-5.5, Gemini 3.x Pro) robi review, plan albo syntezę.

Przykładowo: tani model przeczyta 100 plików i wyciągnie kandydatów. Mocny model zrobi audyt bezpieczeństwa wybranych fragmentów. Tanie modele czytają 200-stronicową umowę i robią jednostronicową ściągę. Mocny model pisze na jej podstawie odpowiedź dla klienta.

Kiedy to przesada: krótka notatka, prosta literówka, szybka inspiracja. Wtedy wystarczy model klasy średniej.

---

10. Checklista „przed kliknięciem Enter”

- [ ] Czy cel jest jasny? Co ma być na końcu?
- [ ] Czy podałem źródła? (pliki, folder, linki, granice)
- [ ] Czy podałem ograniczenia? (czego agent nie może ruszać, nie może wysyłać, nie może kasować)
- [ ] Czy ustaliłem standard jakości? (np. krótko, po polsku, do przesłania bez poprawek)
- [ ] Czy ustaliłem definicję gotowe? (co oznacza, że zadanie jest skończone)
- [ ] Czy poprosiłem o dowody? (plik, link, test, log)

---

11. Najczęstsze mity

Mit 1: „Płacę za ostatnią wiadomość”

Nie. Płacisz za całą historię + pliki + instrukcje + odpowiedź.

Mit 2: „Duże okno kontekstowe znaczy, że mogę wrzucać wszystko”

Możesz, ale to nie znaczy, że warto. Duży kontekst jest droższy i mniej precyzyjny.

Mit 3: „Plan Pro/Plus = unlimited”

Nie. To większy limit, ale z oknami 3h/5h, rate limitami i guardrailami.

Mit 4: „Najmocniejszy model zawsze jest najlepszy”

Nie. Najmocniejszy jest najlepszy do decyzji, syntezy i review. Do roboczych kroków lepszy jest tańszy.

Mit 5: „Cache niczego nie psuje”

Cache pomaga, ale ma TTL (5 min w API, do 1h w subskrypcji). Po przerwie cache wygasa i następne pytanie liczone jest od zera.

Mit 6: „Agent sam się ogarnie”

Nie zawsze. Agent potrzebuje granic, źródeł i standardu. Bez tego marnuje tokeny.

Mit 7: „Kłótnia z modelem to dobry pomysł”

Nie. Wchodzenie w długą wymianę zdań to najdroższa rzecz w AI. Lepiej `/clear` lub `/rewind` i precyzyjne nowe polecenie.

---

12. Co jest stabilne, a co się zmienia

Stabilne

- tokeny jako jednostka,
- długi kontekst kosztuje więcej,
- input taniej niż output,
- duże pliki = dużo tokenów,
- kompaktowanie i streszczanie pomagają,
- review mocnym modelem ma sens przy ważnych zadaniach,
- `CLAUDE.md` / `AGENTS.md` jako pamięć projektu.

Zmienne — sprawdzaj aktualnie

- konkretne ceny modeli i planów,
- limity wiadomości w oknach 3h/5h,
- liczby tokenów dla poszczególnych modeli,
- cache TTL,

- rate limits API,
- dostępność modeli w danym planie,
- „unlimited” (zwykle nie oznacza naprawdę nieograniczonego użycia).

---

13. Prosta reguła dla początkujących

1. **Mały folder, mały plik.** Nie cały dysk.
2. **Plan przed wykonaniem.** Najpierw co i jak, potem działaj.
3. **Nie podłączaj wszystkich narzędzi.** Tylko te, których zadanie potrzebuje.
4. **Po etapie zapisz stan do pliku.** `STATE.md`, `RAPORT.md`, `DECYZJE.md`.
5. **Nowa rozmowa po kamieniu milowym.** Krótki stan + zadanie.
6. **Mocny model tylko tam, gdzie stawka jest wysoka.** Reszta tanim modelem.
7. **Agent zostawia dowody.** Pliki, testy, logi, linki.

To rozwiązuje 80% problemów z kosztami i limitami.

---

14. Oficjalne linki

Ceny i limity zmieniają się często, dlatego materiał nie podaje twardych tabel. Przed ważną decyzją budżetową sprawdź aktualny stan:

Anthropic (Claude, Claude Code, Desktop)

- Cennik i plany: https://www.anthropic.com/pricing
- Claude Code docs: https://code.claude.com/docs
- API docs (token counting, prompt caching, rate limits): https://docs.anthropic.com/
- Usage credits dla planów płatnych: https://support.claude.com/

OpenAI (ChatGPT, Codex, API)

- Cennik: https://openai.com/pricing
- Codex: https://openai.com/codex
- API rate limits: https://platform.openai.com/docs/guides/rate-limits
- Tokenizer: https://platform.openai.com/tokenizer

Google (Gemini, Antigravity, Gemini CLI)

- Gemini docs: https://ai.google.dev/gemini-api/docs
- Gemini pricing: https://ai.google.dev/gemini-api/docs/pricing
- Gemini rate limits: https://ai.google.dev/gemini-api/docs/rate-limits
- Google One AI / Gemini w subskrypcjach: https://one.google.com/about/google-ai-plans