BudujemyAI_

MODUŁ 5: TWOJE CENTRUM DOWODZENIA AI

Darmowy kurs Claude Code: CLAUDE.md, Obsidian, pętla lekcji i skille

Jak sprawić, żeby AI znało Twoją firmę, planowało przed działaniem i nie powtarzało tych samych błędów

JEDNA PRACA, KTÓRĄ WYKONASZ

Zbudujesz w Claude Code centrum pracy firmy — CLAUDE.md, pętlę lekcji i pierwszy skill — i wykonasz w nim pierwsze zadanie z planem i weryfikacją.



Daniel Baranowski
BudujemyAI · wdrożenia AI w małych i średnich firmach
wiedza.budujemyai.pl · wydanie: listopad 2026

W skrócie

DLA KOGO	Dla właścicieli i menedżerów MŚP, którzy korzystają z czatu AI i chcą, żeby AI znało ich firmę i uczyło się na błędach.
JEDNA PRACA	**Zbudujesz w Claude Code centrum pracy firmy — CLAUDE.md, pętlę lekcji i pierwszy skill — i wykonasz w nim pierwsze zadanie z planem i weryfikacją.**
EFEKT	Masz folder firmy w Obsidianie z CLAUDE.md, tasks/todo.md, tasks/lessons.md i działającym skillem — gotowy do codziennej pracy z Claude Code.
CZAS	2 h
POZIOM	Średni
KOSZT	od 20 USD/mies. bez podatku (Claude Pro, stan: październik 2026); Obsidian i Handy 0 zł
NARZĘDZIA	Claude Code, Aplikacja Claude (desktop), Obsidian, Handy

Jak korzystać z tego e-booka

1. **Przeczytaj rozdziały 1–3** — zrozumiesz, po co to robisz i jak to działa.
2. **Zrób „Krok po kroku" z otwartym komputerem** — prompty skopiujesz z wersji online: wiedza.budujemyai.pl/ebooki/kurs-claude-code/.
3. **Sprawdź wynik checklistą kontroli jakości** — to najważniejszy rozdział.
4. **Wracaj do „Zestawu do powtórzenia"**, kiedy robisz to następnym razem.

Spis treści

01 Zanim zaczniesz

02 Dlaczego to ważne dla Twojej firmy

03 Jak to działa — w prostych słowach

04 Jak to robię u siebie

05 Ile to kosztuje

06 Bezpieczeństwo i dane firmy

07 Krok po kroku

08 Jak rozpoznać, że AI zrobiło dobrą robotę

09 Najczęstsze błędy

10 Zestaw do powtórzenia

11 Twój plan działania

12 Słowniczek

13 Źródła

14 Co dalej

ROZDZIAŁ 1

Zanim zaczniesz

Ta lekcja jest dla Ciebie, jeśli korzystasz z czatu AI i masz dość tłumaczenia od nowa, czym zajmuje się Twoja firma i czego AI ma nie robić. Nie musisz programować — Claude Code pracuje na zwykłych plikach: notatkach, ofertach, procedurach.

Jedna praca: zbudujesz w Claude Code centrum pracy firmy — CLAUDE.md z zasadami, pętlę lekcji, która zamienia każdą korektę w regułę, i pierwszy skill — a potem wykonasz w nim pierwsze zadanie: z planem przed działaniem i sprawdzeniem po.

Czego potrzebujesz:

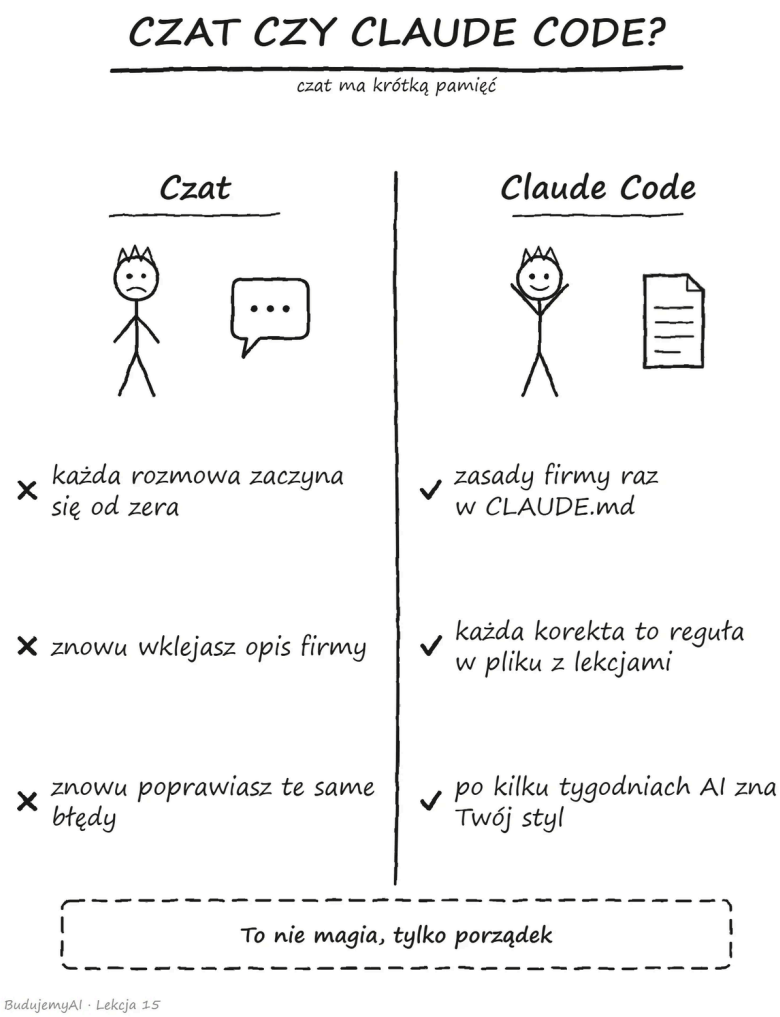
- Komputer z Windows 10 (wersja 1809 lub nowsza) albo macOS 13 lub nowszym, minimum 4 GB RAM, internet.
- Płatny plan Claude: Pro, Max, Team lub Enterprise — darmowy plan nie obejmuje Claude Code. Na start wystarczy Pro.
- Obsidian — darmowy program do notatek, także do użytku w firmie.
- Opcjonalnie: Handy do dyktowania, a na Windowsie Git for Windows (Claude Code działa bez niego, ale z nim ma więcej możliwości).

Ile to zajmie: około 2 godzin — instalacja 15 min, CLAUDE.md 30–40 min, pierwsze zadanie z kontrolą 40 min, skill 20 min. Potem po każdej korekcie dopisujesz jedną regułę.

Poziom: średni. Jeśli umiesz zainstalować program i skopiować polecenie, dasz radę. Gdy „CLI" czy „MCP" brzmią obco, zajrzyj najpierw do lekcji o CLI, MCP i API.

ROZDZIAŁ 2

Dlaczego to ważne dla Twojej firmy



Czat z AI ma krótką pamięć. Każda nowa rozmowa zaczyna się od zera: znowu wklejasz opis firmy, znowu prosisz o formę „Pan/Pani", znowu poprawiasz te same błędy. Wynik jest „prawie dobry", a to „prawie" zjada Twój czas.

Typowa sytuacja: właścicielka biura rachunkowego co miesiąc prosi AI o mail przypominający klientom o dokumentach. AI za każdym razem pisze inaczej: raz myli termin, raz zwraca się do klientów na „ty", raz dopisuje usługę, której biuro nie świadczy. Po roku poprawek są dziesiątki — i żadna nie zostaje zapamiętana.

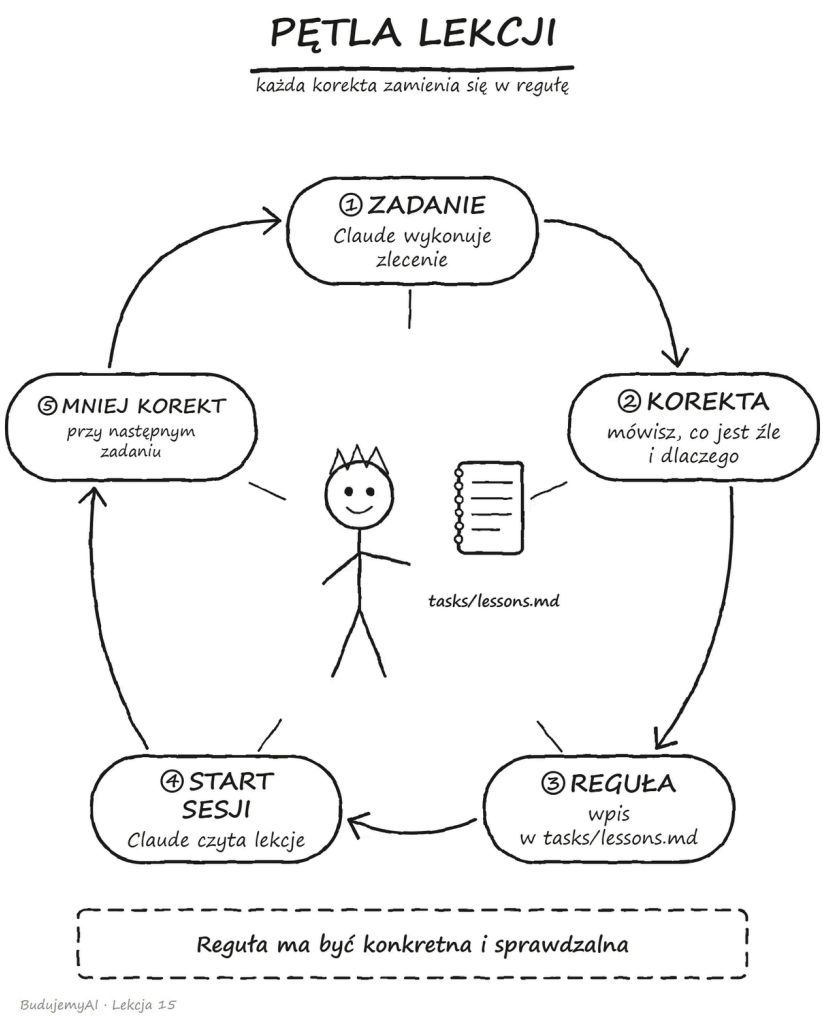
Claude Code odwraca ten układ. Zasady firmy zapisujesz raz w pliku CLAUDE.md, który AI czyta na starcie każdej sesji, a każdą korektę zamieniasz w regułę w pliku z lekcjami. Po kilku tygodniach AI zna Twój styl, terminy i Twoje „nigdy tego nie rób". To nie magia, tylko porządek — jak dobra instrukcja stanowiskowa.

Boris Cherny, twórca Claude Code, opisał publicznie, jak pracuje jego zespół: mają wspólny CLAUDE.md i dopisują do niego regułę za każdym razem, gdy Claude zrobi coś źle. Ta lekcja przenosi ten sposób pracy do małej firmy.

Nie obiecuję konkretnej liczby odzyskanych godzin — to zależy od ilości powtarzalnej pracy. Zmierz to: przez tydzień zapisuj, ile razy poprawiasz AI przy tym samym typie zadania, a po miesiącu z pętlą lekcji policz jeszcze raz.

ROZDZIAŁ 3

Jak to działa — w prostych słowach



Czat to konsultant przez telefon. Claude Code to pracownik przy Twoim biurku. Konsultant słyszy tylko to, co mu powiesz. Pracownik ma dostęp do segregatora: sam otwiera pliki, porównuje oferty, zapisuje dokumenty i — jeśli pozwolisz — uruchamia programy. Potrzebuje więc dobrej instrukcji i jasnych uprawnień.

Element	Analogia z firmy	Co robi
CLAUDE.md	instrukcja stanowiskowa	Claude czyta ją na starcie każdej sesji
tasks/todo.md	tablica z zadaniami	plan zadania jako lista do odhaczenia, na końcu „Review"
tasks/lessons.md	zeszyt błędów	każda korekta zapisana jako reguła na przyszłość
Tryb planowania	kosztorys przed remontem	Claude proponuje plan i niczego nie zmienia przed Twoją akceptacją

Element	Analogia z firmy	Co robi
Subagent	praktykant do researchu	przegląda wiele plików we własnym kontekście i oddaje podsumowanie
Skill	procedura w segregatorze	przepis na powtarzalne zadanie, uruchamiany np. poleceniem /oferta
Hook	automatyczny bezpiecznik	polecenie, które uruchamia się zawsze w danym momencie
Plugin	zestaw narzędzi w pudełku	paczka skilli, subagentów i hooków instalowana jednym poleceniem

Pamięć ma dwa piętra. CLAUDE.md piszesz Ty. Claude Code prowadzi też własną automatyczną pamięć, ale poza Twoim wzrokiem. Plik tasks/lessons.md jest jawny — czytasz go i poprawiasz w Obsidianie — dlatego pętlę prowadzimy w nim.

Obsidian to biurko, na którym leżą te pliki. Zapisuje notatki jako zwykłe pliki Markdown (tekst z prostym formatowaniem) w folderze na dysku, zwanym vaultem. Claude Code czyta i zapisuje te same pliki, a Ty na bieżąco widzisz, co napisał.

Uprawnienia, czyli ile wolno bez pytania. W aplikacji tryb wybierasz selektorem obok przycisku wysyłania, w terminalu klawiszami Shift+Tab:

Tryb	Co Claude robi bez pytania	Kiedy używać
Manual	tylko czyta	wrażliwe pliki, pierwsze dni
Accept edits	czyta i edytuje pliki w folderze	gdy przeglądasz zmiany po fakcie
Plan	czyta i planuje, nic nie zmienia	każde większe zadanie
Auto	prawie wszystko; osobny model blokuje ryzykowne akcje	długie zadania, którym ufasz
Bypass permissions	wszystko	tylko izolowane środowisko, nigdy komputer z danymi firmy

W nowszych wersjach sesje w terminalu startują w trybie Auto. Na początek przełącz się na Manual albo Plan — zobaczysz, jak Claude pracuje, zanim dasz mu więcej swobody.

Gdzie pracuje Claude. W aplikacji desktop wybierasz środowisko sesji: **Local** (Twój komputer — tak pracujesz w tej lekcji), **SSH** (Twój serwer z Linuksem lub macOS; aplikacja sama instaluje tam Claude Code) albo **Cloud** (infrastruktura Anthropic; sesja trwa po zamknięciu aplikacji). Remote Control to okienko do sesji działającej u Ciebie: włączasz je poleceniem /remote-control i sterujesz z telefonu lub przeglądarki, dopóki komputer jest włączony. Konfigurację 24/7 pokazuję w lekcji o Claude Code z telefonu.

Pętla samodoskonalenia w jednym zdaniu: zadanie → Twoja korekta → reguła w tasks/lessons.md → Claude czyta lekcje na starcie sesji → mniej korekt przy następnym zadaniu.

ROZDZIAŁ 4

Jak to robię u siebie

Mój vault w Obsidianie to centrum operacyjne firmy: strategia, treści, klienci, baza wiedzy. W głównym folderze leży CLAUDE.md, obok folder tasks z plikami todo.md i lessons.md. Pracuję na planie Max, w aplikacji desktop w trybie Local, a do zadań całodobowych mam serwer VPS sterowany z telefonu przez Remote Control. Przy ważnych decyzjach Claude zbiera „drugie opinie" od innych modeli (lekcja o radzie doradczej AI), a nową wiedzę podaje mi jako interaktywną tablicę HTML.

Mój CLAUDE.md ma taką strukturę:

Sekcja	Kluczowa zasada (cytat z mojego pliku)
1. Plan Mode domyślnie	„Wchodź w tryb planowania przy KAŻDYM nietrywialnym zadaniu (3+ kroki lub decyzje strukturalne)"
2. Strategia subagentów	„Jedno zadanie na jednego subagenta — skupiona egzekucja"
3. Pętla samodoskonalenia	„Po KAŻDEJ korekcie od użytkownika: zaktualizuj tasks/lessons.md o wzorzec błędu"
4. Weryfikacja przed „gotowe"	„Czy staff engineer by to zaakceptował?"
5. Elegancja z umiarem	„Pomijaj to przy prostych, oczywistych poprawkach — nie przekombinuj"
6. Autonomiczne naprawianie błędów	„Gdy dostajesz zgłoszenie błędu: po prostu je napraw"
Task Management i Core Principles	plan w tasks/todo.md z sekcją „Review"; prostota, zero prowizorek, minimalny zasięg zmian

Uczciwie o pochodzeniu: to popularny szablon, który społeczność złożyła z publicznych wskazówek Borisa Chernego — sam Boris tego pliku nie opublikował. W wątkach na X opisał zasady: większość sesji zaczyna od trybu planowania, każdy błąd Claude trafia jako reguła do CLAUDE.md, a najważniejsze jest danie Claude sposobu na sprawdzenie własnej pracy (według jego szacunku poprawia to jakość wyniku 2–3-krotnie). Linki są w Źródłach.

Dwa prawdziwe wpisy z mojego lessons.md, opisane ogólnie:

Lekcja 1: test, który prawie trafił do prawdziwej skrzynki

Testowaliśmy z Claude formularz kontaktowy. Lokalna kopia konfiguracji miała prawdziwy adres, pod który strona wysyła zgłoszenia, więc testowe zgłoszenie mogło trafić do mojego arkusza z zapytaniami i na moją pocztę. Reguła: przed każdym testem podmień wszystko, co wychodzi na zewnątrz (formularze, maile, integracje), na atrapy, a dane testowe oznaczaj „TEST — do usunięcia".

Lekcja 2: AI zgadło, czego chcę — i zgadło źle

Poprosiłem o ściągę konfiguracji nowego laptopa. Claude założył, że laptop ma być tylko okienkiem do serwera, a ja chciałem pełnej stacji roboczej. Reguła: przy nowym komputerze najpierw odwzoruj dotychczasowy setup, a serwer proponuj jako dodatek.

Druga lekcja dotyczy komunikacji: AI wypełniło lukę w moim poleceniu własnym założeniem. W kroku 5 uczysz się pisać polecenia, przy których nie musi zgadywać.

ROZDZIAŁ 5

Ile to kosztuje

Ceny bez podatku, stan: październik 2026 — przed zakupem sprawdź aktualny cennik.

Element	Cena	Uwagi
Claude Pro	20 USD/mies. lub 17 USD/mies. przy płatności rocznej	wystarczy na start
Claude Max 5x / 20x	100 / 200 USD/mies.	5 lub 20 razy większy limit niż Pro
Claude Team	25 USD/os./mies. (20 USD przy płatności rocznej); Premium 125 USD (100 USD)	od 2 osób, warunki komercyjne
Obsidian	0 zł	licencja komercyjna dobrowolna (cennik)
Handy	0 zł	open source, licencja MIT

ROZDZIAŁ 6

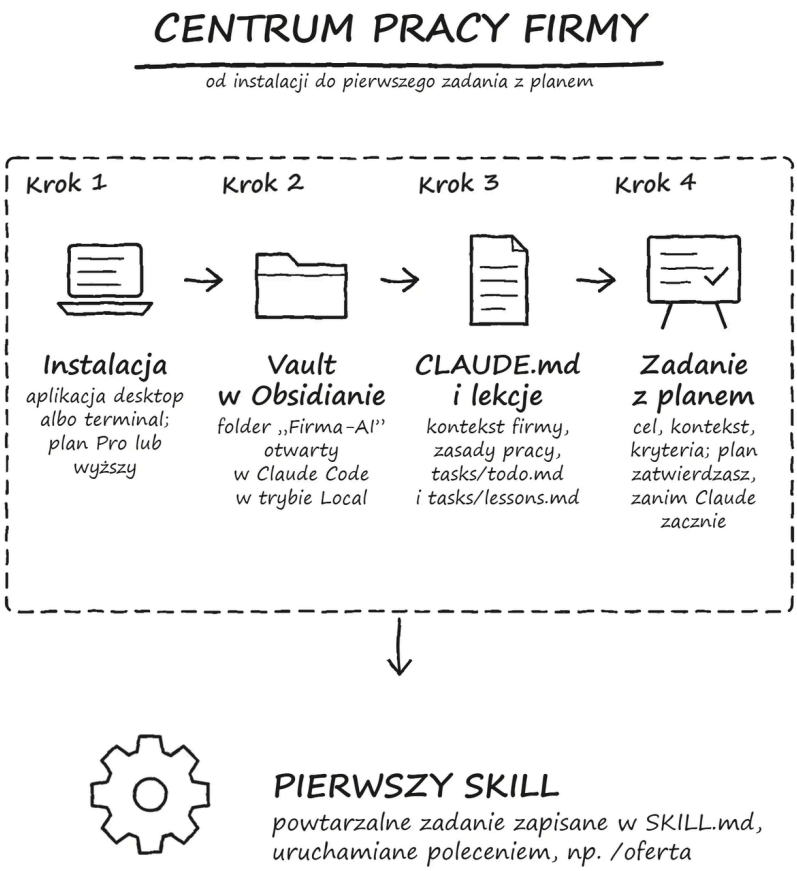
Bezpieczeństwo i dane firmy

Claude Code widzi folder, który mu otworzysz, a treść przeczytanych plików trafia do modelu w chmurze. Dlatego:

- **Osobny vault dla AI** — bez danych osobowych klientów; dane wrażliwe tylko zanonimizowane.
- **Zero sekretów w plikach.** Hasła i klucze trzymaj w menedżerze haseł (lekcja o Bitwardenie), nigdy w CLAUDE.md.
- **Ustawienia prywatności.** Na planach Free, Pro i Max sam decydujesz w ustawieniach konta, czy rozmowy — także z Claude Code — mogą służyć do ulepszania modeli (przy zgodzie dane są przechowywane do 5 lat, bez niej 30 dni). Na Team, Enterprise i w API domyślnie nie służą do trenowania (zasady danych).
- **RODO.** Umowa powierzenia (DPA) Anthropic jest częścią warunków komercyjnych (Team, Enterprise, API); Pro i Max działają na warunkach konsumenckich. Do danych osobowych klientów wybierz plan komercyjny i skonsultuj to z prawnikiem lub inspektorem ochrony danych. To nie jest porada prawna.
- **Kopia zapasowa.** Punkty kontrolne Claude Code (/rewind) cofają tylko edycje zrobione jego narzędziami. Przed większymi porządkami skopiuj folder albo używaj gita (lekcja o GitHubie).

ROZDZIAŁ 7

Krok po kroku



Krok 1: Zainstaluj Claude Code i zaloguj się

Najprościej przez aplikację desktop: pobierz ją ze strony claude.com/download, zainstaluj, zaloguj się kontem z planem Pro lub wyższym i kliknij zakładkę **Code** u góry okna. Jeśli aplikacja proponuje zmianę planu, Twój plan nie obejmuje Claude Code.

Wolisz terminal? Wklej polecenie z oficjalnej dokumentacji — w PowerShellu na Windowsie:

```powershell
irm https://claude.ai/install.ps1 | iex
```

albo w Terminalu na macOS:

```
TERMINAL

curl -fsSL https://claude.ai/install.sh | bash
```

Potem w nowym oknie terminala wpisz `claude --version` (pokaże wersję) i `claude doctor` (diagnoza instalacji). Logujesz się przy pierwszym uruchomieniu polecenia `claude` — otworzy się przeglądarka. Taka instalacja aktualizuje się sama.

Krok 2: Załóż folder firmy w Obsidianie i otwórz go w Claude Code

1. Zainstaluj Obsidian i utwórz nowy vault, np. „Firma-AI".
2. Załóż foldery: tasks, oferty, procedury, notatki, szablony. Do folderu oferty wrzuć 2–3 ostatnie oferty (bez danych osobowych), a w głównym folderze utwórz plik cennik.md.
3. W aplikacji, w zakładce Code, wybierz **Local**, kliknij **Select folder** i wskaż vault. W terminalu przejdź do folderu poleceniem `cd` i wpisz `claude`.
4. Na pytanie o zaufanie do folderu odpowiadaj „tak" tylko dla własnych folderów.

Skille trafią do folderu .claude, który bywa ukryty w Obsidianie i Finderze — to normalne.

Krok 3: Napisz swój CLAUDE.md

Poproś Claude o wywiad (prompt w Zestawie do powtórzenia) albo skopiuj szablon do pliku CLAUDE.md w głównym folderze vaulta i uzupełnij nawiasy. To skrót mojej struktury, przerobiony z programowania na zwykłą firmę.

```MARKDOWN
# CLAUDE.md — [nazwa firmy]

## Kontekst firmy
- Firma: [nazwa, branża, region]; klienci: [kto kupuje]
- Ton: [np. forma Pan/Pani, konkretnie, bez żargonu]
- Foldery: oferty/, procedury/, notatki/, szablony/

## Zasady pracy
1. Najpierw plan: przy zadaniu z 3+ krokami pokaż plan i poczekaj na akceptację; gdy coś idzie źle, zaplanuj od nowa
2. Przeglądanie wielu plików zlecaj subagentom; wraca tylko podsumowanie
3. Na początku sesji przeczytaj tasks/lessons.md; po każdej mojej korekcie dopisz tam wpis z regułą
4. Bez „gotowe” bez dowodu: lista zmienionych plików + tabela kryterium → gdzie spełnione
5. Pytanie kontrolne: czy nasz najlepszy fachowiec przyjąłby to bez poprawek?
6. Ceny, terminy i dane klientów tylko z plików; gdy czegoś brakuje, zapytaj
7. Najprostsze rozwiązanie; nie ruszaj niczego poza zakresem zadania

## Zadania
- Plan zapisz w tasks/todo.md jako listę do odhaczenia, odhaczaj na bieżąco
- Na końcu sekcja „Review”: co zrobiono, jak sprawdzono, co zostało

## Nigdy
- Nie zapisuj haseł, kluczy ani numerów kont w plikach
- Nie wysyłaj niczego na zewnątrz bez mojej wyraźnej zgody
- Nie usuwaj plików (przenoś do archiwum/) i nie zmieniaj folderu .obsidian/
```

Trzy zasady dobrego CLAUDE.md według dokumentacji Anthropic:

- **Krótko.** Celuj w mniej niż 200 linii — dłuższego pliku Claude gorzej przestrzega. Przy każdej linii pytaj: czy bez niej Claude popełni błąd? Jeśli nie, wytnij.
- **Sprawdzalnie.** „Pisz krótko” nie działa. „Mail do klienta: maks. 120 słów, forma Pan/Pani” działa.
- **Bez sprzeczności.** Przy dwóch sprzecznych zasadach Claude może wybrać dowolną. Zasadę uparcie pomijaną możesz oznaczyć słowem „WAŻNE” — ale tylko jedną.

Sprawdź, czy plik działa: w sesji wpisz `/context` — CLAUDE.md powinien być na liście plików pamięci (Memory files). Polecenie `/memory` otwiera go do edycji.

> **! CLAUDE.md to rada, nie zamek**
>
> Claude zwykle stosuje się do CLAUDE.md, ale bez gwarancji. To, co ma się dziać zawsze albo nigdy, ustawiasz twardo: regułami uprawnień (/permissions) albo hookami.

Krok 4: Dodaj tasks/todo.md i tasks/lessons.md

W folderze tasks utwórz dwa pliki. todo.md zostaw pusty — Claude zapisze tam plan zadania. lessons.md zacznij od formatu wpisu:

```markdown
# Lekcje — czytaj na początku każdej sesji

### RRRR-MM-DD — krótki opis błędu
- Co poszło nie tak:
- Reguła na przyszłość:
```

Najważniejsze jest pole „Reguła na przyszłość". „Pomyliłem termin" to nie reguła. „Terminy bierz tylko z pliku procedury/terminy.md, a gdy terminu brak, zapytaj" — to jest reguła: konkretna i sprawdzalna.

Chcesz, żeby lekcje wczytywały się same? Dopisz w CLAUDE.md import `Lekcje: @tasks/lessons.md` i pilnuj, żeby wpisy były zwięzłe — zajmują miejsce w kontekście.

Krok 5: Pisz polecenia jak zlecenie dla wykonawcy i dyktuj je

Claude Code jest tak dobry, jak Twoje zlecenie. Dobre polecenie ma pięć części:

Część	Pytanie	Przykład
Cel	co ma powstać i po co?	szablon oferty na remont łazienki
Kontekst	na czym ma się oprzeć?	3 oferty z folderu oferty, cennik.md
Kryteria akceptacji	po czym poznam, że jest dobrze?	6 sekcji, zero zgadywanych cen
Ograniczenia	czego nie ruszać?	istniejących ofert
Weryfikacja	jak Claude ma to sprawdzić?	tabela: wymaganie → gdzie spełnione

„Zrób szablon oferty" daje „coś". Polecenie z pięcioma częściami daje wynik, który sprawdzisz w kilka minut.

Dyktuj zamiast pisać. Mówiąc, naturalnie podajemy więcej kontekstu. Ja używam Handy — darmowej aplikacji open source, która działa offline: naciskasz skrót, mówisz, a tekst pojawia się w dowolnym polu, bez wysyłania nagrania w świat. Wbudowane dyktowanie Claude Code w terminalu (`/voice`) też obsługuje polski (ustaw go w `/config`), ale nagranie trafia na serwery Anthropic. Więcej w lekcji o pisaniu głosem.

Dyktowane polecenie bywa chaotyczne, więc na końcu dodaj: „Zanim zaczniesz, powtórz w 3 punktach cel, kryteria i ograniczenia". Jeśli Claude źle zrozumiał, poprawiasz przed pracą, a nie po.

Krok 6: Wykonaj pierwsze zadanie w trybie planowania

1. Włącz tryb planowania: w aplikacji wybierz **Plan** w selektorze trybu; w terminalu naciskaj Shift+Tab, aż na pasku stanu zobaczysz napis „plan mode on", albo zacznij polecenie od `/plan`.
2. Wklej zlecenie według wzoru poniżej.

3. Przeczytaj plan: czy ma kroki, listę plików, kryteria i sposób sprawdzenia? W terminalu Ctrl+G otwiera plan w edytorze do poprawki.
4. Przy pierwszych zadaniach zatwierdź opcją **Yes, manually approve edits** — akceptujesz każdą zmianę osobno. Zły plan? Wybierz **No, keep planning** i powiedz, co zmienić.

› PROMPT

```
Cel: przygotuj szablon oferty na remont łazienki, który będę wypełniać dla nowych klientów.
Kontekst: w folderze oferty są 3 moje ostatnie oferty, ceny są w pliku cennik.md. Klienci to
właściciele mieszkań, piszemy w formie Pan/Pani.
Kryteria akceptacji: sekcje zakres prac, materiały, harmonogram, cena, warunki płatności,
gwarancja; puste miejsca oznaczone [DO UZUPEŁNIENIA]; żadnych cen spoza cennik.md.
Ograniczenia: nie zmieniaj istniejących plików; wynik zapisz jako szablony/oferta-lazienka.md.
Weryfikacja: na koniec pokaż listę zmienionych plików i tabelę wymaganie → gdzie spełnione.
Najpierw przedstaw plan i poczekaj na akceptację.
```

◆ **Jak wygląda dobry plan (przykład ilustracyjny)**

1. Przeczytam 3 oferty i cennik.md, wypiszę wspólne sekcje.
2. Zapiszę szablony/oferta-lazienka.md z 6 sekcjami — jedyny nowy plik.
3. Sprawdzę, czy każda kwota występuje w cennik.md, i pokażę tabelę wymagań.
4. Zapiszę plan i sekcję „Review" w tasks/todo.md.

Nie masz ofert? Ten sam schemat zadziała dla przypomnienia o dokumentach albo odpowiedzi na opinię w Google.

Krok 7: Sprawdź wynik, popraw i zapisz lekcję

1. **Dowód.** Claude ma pokazać listę zmienionych plików i tabelę kryteriów. Samo „gotowe" nie wystarcza.
2. **Zmiany.** W aplikacji po edycjach pojawia się wskaźnik zmian, np. `+12 -1` — kliknij go, a zobaczysz widok różnic plik po pliku i przycisk **Review code** do samooceny Claude. W terminalu napisz: „Pokaż, co dokładnie zmieniłeś w każdym pliku".
3. **Druga para oczu.** Poproś subagenta o niezależny przegląd (prompt w Zestawie). Ma świeży kontekst, więc nie broni pracy głównego Claude.
4. **Korekta.** Napisz, co jest źle i dlaczego, a na końcu dodaj: „Zaktualizuj tasks/lessons.md, żeby ten błąd się nie powtórzył". To rada Borisa Chernego przeniesiona na plik z lekcjami.
5. **Reset.** Poprawiasz to samo drugi raz i dalej źle? Dokumentacja Anthropic radzi wtedy wyczyścić rozmowę poleceniem `/clear` i zacząć od lepszego polecenia.

Esc zatrzymuje Claude w trakcie pracy, a podwójny Esc albo `/rewind` cofa rozmowę i zmiany.

Krok 8: Zamień powtarzalne zadanie w skill

Boris Cherny radzi zamieniać w skill wszystko, co robisz częściej niż raz dziennie — w małej firmie wystarczy „co tydzień". Skill to folder z plikiem SKILL.md. Poproś Claude: „Utwórz skill oferta na podstawie tego, co właśnie zrobiliśmy" albo zapisz plik `.claude/skills/oferta/SKILL.md` sam:

```markdown
---
name: oferta
description: Przygotowuje ofertę dla klienta na podstawie szablonu i notatek ze spotkania. Używaj, gdy proszę o ofertę lub wycenę.
disable-model-invocation: true
---

Przygotuj ofertę dla: $ARGUMENTS

1. Przeczytaj szablony/oferta-lazienka.md, cennik.md i wskazane notatki.
2. Uzupełnij pola, dla których masz dane; resztę oznacz [DO UZUPEŁNIENIA].
3. Ceny bierz wyłącznie z cennik.md.
4. Zapisz wynik w oferty/robocze/ z datą w nazwie pliku.
5. Na koniec pokaż brakujące dane i tabelę kryterium → spełnione/niespełnione.
```

- `name` to nazwa polecenia: wpisujesz np. `/oferta łazienka 6 m², notatki/spotkanie-klient-A.md`, a tekst po nazwie trafia w miejsce `$ARGUMENTS`.
- `description` mówi Claude, do czego służy skill.
- `disable-model-invocation: true` sprawia, że skill uruchamiasz tylko Ty — przy dokumentach dla klientów to bezpieczniejsze.

Skill z folderu projektu działa w tym vaulcie, a skill w `~/.claude/skills/` — we wszystkich projektach. Wczytuje się dopiero przy użyciu, więc w przeciwieństwie do CLAUDE.md nie obciąża każdej rozmowy.

> **i Subagenci, hooki i pluginy — kiedy po nie sięgnąć**
>
> Subagent: gdy trzeba przejrzeć wiele plików — dopisz „użyj subagentów". Hook: gdy coś ma się dziać zawsze; poproś np. „Napisz hook, który pokaże powiadomienie, gdy czekasz na moją decyzję", i sprawdź go poleceniem /hooks. Plugin: gotowe paczki z oficjalnego katalogu (polecenie /plugin). Plugin działa z Twoimi uprawnieniami, więc instaluj tylko zaufane — jak je oceniać, pokazuję w lekcji o repozytoriach GitHub.

ROZDZIAŁ 8

Jak rozpoznać, że AI zrobiło dobrą robotę



AI pracuje tu na pięciu odcinkach i na każdym sprawdzasz je inaczej.

1. Plan. Dobry plan ma kroki, listę plików, kryteria i sposób sprawdzenia. Sygnał ostrzegawczy: plan bez kroku weryfikacji albo zmiany w plikach, o których nie było mowy.

2. Wykonanie. Claude ma pokazać dowód, a nie deklarację: listę zmienionych plików, tabelę „kryterium → gdzie spełnione", wynik sprawdzenia. Przejrzenie dowodu jest szybsze niż powtarzanie sprawdzenia samemu — tak radzi też dokumentacja Anthropic.

3. Wynik. Zrób trzy testy:

- *Test faktów* — poproś o tabelę „liczba lub nazwa → plik źródłowy" i sprawdź losowo 3 pozycje. Cena spoza cennika to błąd krytyczny.
- *Test zakresu* — w widoku zmian nie ma plików spoza zadania.

- *Test najlepszego fachowca* — staff engineer to najbardziej doświadczony inżynier w zespole, stąd moje pytanie „Czy staff engineer by to zaakceptował?". W Twojej firmie zapytaj: czy mój najlepszy fachowiec podpisałby to bez poprawek?

4. Pętla lekcji. Po korekcie w tasks/lessons.md jest wpis z regułą, a przy następnym podobnym zadaniu błąd nie wraca. Raz w tygodniu zrób test zimnego startu: nowa sesja i pytanie „Jakie są zasady pracy w tym folderze i co zrobisz po mojej korekcie?". Odpowiedź ma wynikać z CLAUDE.md.

5. Sam CLAUDE.md. Jeśli Claude pyta o rzeczy zapisane w pliku, sformułowania są niejasne. Jeśli łamie zasadę, która tam jest, plik jest pewnie za długi. W nowszych wersjach polecenie `/doctor prompt-audit` wyszuka sprzeczne i nieaktualne instrukcje, a zmiany wprowadzi dopiero na Twoją prośbę.

Druga opinia z umiarem. Subagent poproszony o szukanie luk zwykle jakieś znajdzie, nawet gdy praca jest dobra. Każ mu zgłaszać tylko braki wpływające na poprawność albo kryteria, resztę traktuj jako sugestie.

Mierz, nie zgaduj. Prowadź tabelę: zadanie, liczba korekt, czas poprawek. Gdy korekt przy tym samym typie zadania ubywa, pętla działa. Gdy nie — lekcje są za ogólne albo CLAUDE.md za długi.

✓ **Checklista kontroli jakości**

- [ ] Plan miał kroki, pliki, kryteria i sposób sprawdzenia — i to ja go zatwierdziłem
- [ ] Claude pokazał dowód: listę zmienionych plików i tabelę kryteriów
- [ ] W widoku zmian nie ma plików spoza zadania
- [ ] 3 losowe fakty sprawdziłem w plikach źródłowych
- [ ] Mój najlepszy fachowiec przyjąłby wynik bez poprawek
- [ ] Każda korekta ma regułę w tasks/lessons.md
- [ ] CLAUDE.md ma mniej niż 200 linii i nie zawiera sprzeczności

ROZDZIAŁ 9

Najczęstsze błędy

1. **CLAUDE.md jak regulamin na 20 stron.** Skutek: ważne zasady giną w szumie. Jak uniknąć: mniej niż 200 linii, procedury przenoś do skilli.
2. **Poprawianie w kółko w jednej rozmowie.** Skutek: kontekst pełen nieudanych prób, odpowiedzi coraz gorsze. Jak uniknąć: po dwóch nieudanych korektach `/clear`, wpis w lessons.md i lepsze polecenie.
3. **„Gotowe" bez dowodu.** Skutek: błąd wychodzi dopiero u klienta. Jak uniknąć: kryteria akceptacji w każdym zleceniu i zasada „dowód przed gotowe" w CLAUDE.md.
4. **Plan przy literówce, brak planu przy dużym zadaniu.** Skutek: strata czasu albo praca w złym kierunku. Jak uniknąć: tryb planowania od 3 kroków; zmianę, którą da się opisać jednym zdaniem, zlecaj wprost.
5. **Wszystko w jednym oknie.** Skutek: oferta miesza się ze stroną www i Claude myli wątki. Jak uniknąć: `/clear` między niezwiązanymi tematami, osobne sesje.
6. **Pełna swoboda na komputerze z danymi firmy.** Skutek: zmiany, których nie chciałeś. Jak uniknąć: Manual lub Plan na start, Auto przy zaufanych zadaniach, Bypass permissions tylko w izolowanym środowisku.
7. **Sekrety i dane osobowe w notatkach.** Skutek: trafiają do kontekstu modelu. Jak uniknąć: hasła w menedżerze haseł, dane klientów zanonimizowane, osobny vault dla AI.
8. **Lekcje bez reguły.** Skutek: wpis „pomyliłem się" niczego nie zmienia. Jak uniknąć: każdy wpis kończy się regułą, którą da się sprawdzić.

ROZDZIAŁ 10

Zestaw do powtórzenia

Prompt: wywiad, z którego powstanie CLAUDE.md

› PROMPT

```
Przeprowadź ze mną wywiad, z którego powstanie CLAUDE.md dla mojej firmy. Pytaj po kolei o: działalność i klientów, ton komunikacji, ważne foldery, powtarzalne zadania, czego AI nie może robić i jak sprawdzam, że praca jest dobra. Pomiń oczywiste pytania. Na koniec zapisz CLAUDE.md (maks. 80 linii) w sekcjach: Kontekst firmy, Zasady pracy, Zadania, Nigdy.
```

Prompt: zlecenie zadania

› PROMPT

```
Cel: [co ma powstać i po co]
Kontekst: [pliki, dla kogo, ważne fakty]
Kryteria akceptacji: [3–6 sprawdzalnych warunków]
Ograniczenia: [czego nie zmieniać, gdzie zapisać wynik, ton, długość]
Weryfikacja: na koniec pokaż listę zmienionych plików i tabelę kryterium → gdzie spełnione.
Najpierw przedstaw plan i poczekaj na akceptację. Zanim zaczniesz, powtórz w 3 punktach cel, kryteria i ograniczenia.
```

Prompt: po korekcie

› PROMPT

```
To jest błąd: [co jest źle i dlaczego]. Popraw to. Potem dopisz wpis do tasks/lessons.md: data — krótki opis błędu, „Co poszło nie tak", „Reguła na przyszłość". Reguła ma być konkretna i sprawdzalna. Pokaż mi dopisany wpis.
```

Prompt: niezależny przegląd przez subagenta

› PROMPT

```
Użyj subagenta, żeby niezależnie sprawdził wynik ostatniego zadania względem planu w tasks/todo.md: czy każde kryterium akceptacji jest spełnione, czy liczby i nazwy mają źródło w plikach i czy nic poza zakresem nie zostało zmienione. Zgłoś tylko braki, które wpływają na poprawność lub kryteria — bez uwag o stylu.
```

Prompt: comiesięczne przycinanie CLAUDE.md

› PROMPT

```
Przejrzyj CLAUDE.md i tasks/lessons.md. Zaproponuj: (1) zasady do usunięcia, bo nic nie wnoszą albo się powtarzają, (2) lekcje, które wracają tak często, że warto przenieść je do CLAUDE.md, (3) sprzeczności. Niczego nie zmieniaj — pokaż tabelę: fragment → propozycja → uzasadnienie.
```

Ściąga poleceń

Polecenie lub skrót	Co robi
Shift+Tab (terminal)	przełącza tryby uprawnień, w tym tryb planowania
`/plan` przed poleceniem	jedno polecenie w trybie planowania
Ctrl+G	otwiera plan w edytorze
Esc / Esc Esc lub `/rewind`	zatrzymuje Claude / cofa rozmowę i zmiany
`/clear` , `/compact`	czyści rozmowę / streszcza długą rozmowę
`/context` , `/memory`	pokazuje wczytane pliki pamięci / otwiera je do edycji
`/init`	tworzy startowy CLAUDE.md na podstawie folderu
`/hooks` , `/plugin`	przegląd hooków, katalog pluginów

Checklista: start i koniec sesji

- [ ] Start: właściwy folder i tryb (Plan lub Manual przy nowym typie zadania)
- [ ] Start: Claude przeczytał tasks/lessons.md
- [ ] Każde zlecenie ma cel, kontekst, kryteria, ograniczenia i weryfikację
- [ ] Koniec: dowód, przegląd zmian i sekcja „Review" w tasks/todo.md
- [ ] Koniec: każda korekta zapisana jako reguła w tasks/lessons.md
- [ ] Raz w tygodniu: test zimnego startu i przegląd CLAUDE.md

ROZDZIAŁ 11

Twój plan działania

- **Dzień 1:** instalacja, vault „Firma-AI”, CLAUDE.md z wywiadu.
- **Dzień 2:** pierwsze zadanie w trybie planowania i pełna kontrola jakości według checklisty.
- **Dzień 3:** drugie zadanie tego samego typu i pierwsze wpisy w tasks/lessons.md.
- **Dzień 4:** skill z powtarzalnego zadania; uruchom go na dwóch przykładach i porównaj wyniki.
- **Dzień 5:** niezależny przegląd przez subagenta i test zimnego startu.
- **Dzień 6:** zainstaluj Handy i podyktuj 3 zlecenia według wzoru z kroku 5.
- **Dzień 7:** policz korekty z tygodnia, przytnij CLAUDE.md i wybierz następny skill. Potem przejdź do lekcji o radzie doradczej AI.

ROZDZIAŁ 12

Słowniczek

- **Claude Code** — asystent AI od Anthropic, który pracuje bezpośrednio na plikach w wybranym folderze: czyta je, zapisuje i uruchamia polecenia.
- **CLAUDE.md** — plik z instrukcjami, który Claude Code czyta na starcie każdej sesji.
- **Kontekst** — wszystko, co AI ma przed oczami w rozmowie. Ma ograniczoną pojemność, a gdy się zapełnia, jakość spada.
- **Tryb planowania** — tryb, w którym Claude proponuje plan i niczego nie zmienia przed Twoją akceptacją.
- **Pętla samodoskonalenia** — nawyk zamieniania każdej korekty w regułę zapisaną w tasks/lessons.md.
- **Subagent** — pomocniczy agent z własnym, czystym kontekstem, który oddaje podsumowanie.
- **Skill** — zapisana procedura (folder z plikiem SKILL.md), uruchamiana poleceniem /nazwa.
- **Hook** — polecenie uruchamiane automatycznie w określonym momencie pracy Claude, niezależnie od decyzji AI.
- **Plugin** — paczka skilli, subagentów i hooków instalowana jako całość.
- **Vault** — folder z notatkami Obsidiana, czyli zwykłe pliki Markdown na Twoim dysku.
- **Remote Control** — sterowanie sesją działającą na Twoim komputerze z telefonu lub przeglądarki.

ROZDZIAŁ 13

Źródła

- Claude Code — instalacja — polecenia, wymagania, plany
- Aplikacja desktop — pierwsze kroki — zakładka Code, środowiska
- Tryby uprawnień — tryby i akceptacja planu
- Pamięć i CLAUDE.md — lokalizacje, długość, importy
- Skille — format SKILL.md
- Subagenci — wbudowane i własne subagenty
- Hooki — automatyczne polecenia
- Pluginy — katalog i bezpieczeństwo
- Best practices Claude Code — weryfikacja, plan, kontekst
- Boris Cherny — wątek z 2.01.2026 — jego setup: plan, CLAUDE.md, weryfikacja
- Boris Cherny — wątek z 31.01.2026 — wskazówki zespołu Claude Code
- Cennik Claude — plany Pro, Max, Team (stan: październik 2026)

Co dalej

NASTĘPNA LEKCJA · 16 z 21

Rada doradcza z superinteligencji: Claude Code + Codex + Grok (+ Buzz)

Przeprowadzisz pierwsze posiedzenie rady doradczej AI i podejmiesz decyzję z testem na 48 godzin.

Obejrzyj wideo

Kanał BudujemyAI na YouTube

Dołącz do Klubu BudujemyAI

Pytania, Kawa z AI co tydzień i live konsultacyjny.

klub.budujemyai.pl

Konsultacja hybrydowa

Dwa spotkania online i e-book PDF z rozwiązaniami dla Twojej firmy — 499 zł. Na start: bezpłatne 15 min.

wiedza.budujemyai.pl/konsultacja/

Pozostałe e-booki — za darmo

21 lekcji ścieżki „Jedna praca" i mapa wiedzy.

wiedza.budujemyai.pl/ebooki/

ANKIETA · 1 MINUTA

Jak mógłbym dodatkowo rozwinąć według Ciebie ten wątek, bądź co jeszcze Cię ciekawi w tym temacie?

Odpowiedz tutaj: wiedza.budujemyai.pl/baza-wiedzy/kurs-claude-code/#ankieta — czytam każdą odpowiedź, a najczęstsze prośby zamieniam w kolejne lekcje.

O autorze



Daniel Baranowski — prowadzi BudujemyAI. Wdraża AI i automatyzacje w małych i średnich firmach, buduje strony www z pomocą AI (także 3D) na własnym systemie, w którym klient sam edytuje stronę, pisząc do AI po polsku.

Na co dzień pracuje z Claude Code, Codexem i Grokiem jako „radą doradczą", a całą firmę prowadzi z vaulta w Obsidianie. W tej bazie pokazuje dokładnie to, czego sam używa — prosto i konkretnie.

ai@baranowski.consulting · budujemyai.pl · wiedza.budujemyai.pl

---

© 2026 BudujemyAI · Daniel Baranowski. Wydanie: listopad 2026. Możesz przesłać ten e-book dalej w niezmienionej formie.

Ceny, nazwy planów i funkcje narzędzi AI zmieniają się często — przed decyzją zakupową sprawdź aktualne informacje u producenta. Treść ma charakter edukacyjny i nie jest poradą prawną ani podatkową.