BudujemyAI_

E-BOOK · LEKCJA 13 · MODUŁ 4

MODUŁ 4: PROFESJONALNY FUNDAMENT FIRMY

GitHub od zera — czym jest, co daje firmie i jak korzystać bezpiecznie

Dlaczego GitHub to przewaga w pracy z AI i jak bez programowania bezpiecznie zrobić pierwszy krok

JEDNA PRACA, KTÓRĄ WYKONASZ

Założysz bezpieczne konto GitHub, utworzysz prywatne repozytorium i zrobisz pierwszy commit bez sekretów.



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

W skrócie

DLA KOGO	Dla właściciela lub menedżera małej firmy, który chce trzymać prompty, procedury i kod tworzony z AI w jednym bezpiecznym miejscu.
JEDNA PRACA	**Założysz bezpieczne konto GitHub, utworzysz prywatne repozytorium i zrobisz pierwszy commit bez sekretów.**
EFEKT	Masz konto z 2FA i passkey, prywatne repozytorium z README i .gitignore, pierwszy sprawdzony commit i checklistę bezpieczeństwa.
CZAS	60–75 min
POZIOM	Start
KOSZT	0 zł (GitHub Free i GitHub Desktop)
NARZĘDZIA	GitHub, GitHub Desktop, aplikacja 2FA, menedżer haseł, Claude lub ChatGPT

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/github-od-zera/.
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 Ile to kosztuje

05 Krok po kroku

06 Jak to robię u siebie

07 Jak rozpoznać, że AI zrobiło dobrą robotę

08 Najczęstsze błędy

09 Zestaw do powtórzenia

10 Twój plan działania

11 Słowniczek

12 Źródła

13 Co dalej

ROZDZIAŁ 1

Zanim zaczniesz

Ta lekcja jest dla Ciebie, jeśli słyszysz „wrzuć to na GitHuba" i nie do końca wiesz, co to znaczy. Albo jeśli zaczynasz pracę z agentami AI, które same zmieniają pliki (Claude Code, Codex), i chcesz mieć nad tym kontrolę. Programować nie musisz.

Jedna praca: założysz bezpieczne konto GitHub, utworzysz prywatne repozytorium i zrobisz pierwszy commit bez sekretów. Dostaniesz też checklistę, która chroni przed najczęstszymi wpadkami.

Czego potrzebujesz	Szczegóły
Komputer	Windows 10 (64-bit) lub nowszy albo macOS 12 lub nowszy — tyle wymaga GitHub Desktop
Telefon	z aplikacją do kodów 2FA, np. Google Authenticator, Microsoft Authenticator lub Bitwarden Authenticator
Menedżer haseł	na hasło i kody odzyskiwania — jeśli go nie masz, zacznij od lekcji o Bitwardenie
Adres e-mail	firmowy, do którego będziesz mieć dostęp także za pięć lat
AI	darmowa wersja Claude, ChatGPT albo Gemini
Koszt	0 zł — GitHub Free i GitHub Desktop są darmowe
Czas i poziom	60–75 minut, poziom Start

> **i Interfejs jest po angielsku**
>
> GitHub i GitHub Desktop nie mają polskiej wersji. Nazwy przycisków i menu podaję w oryginale i pogrubiam — tak, żeby łatwo było je znaleźć na ekranie.

ROZDZIAŁ 2

Dlaczego to ważne dla Twojej firmy

Znasz pliki w rodzaju „oferta_FINAL_v3_poprawiona_ostateczna.docx”? To objaw jednego problemu: nie wiadomo, która wersja jest aktualna, kto zmienił cenę i jak to wyglądało tydzień temu. Każde takie szukanie kosztuje czas i pewność siebie przed klientem.

Praca z AI podnosi stawkę. W czacie wszystko przechodzi przez Twoje ręce. Agent AI — np. Claude Code albo Codex — zmienia pliki sam, czasem kilkanaście naraz. Bez zapisu historii nie wiesz, co dokładnie zrobił. Anthropic pisze wprost, że punkty kontrolne Claude Code nie zastępują kontroli wersji, a zmian wykonanych poleceniami w terminalu w ogóle nie cofają.

Wyobraź sobie firmę instalacyjną, która zleciła wykonawcy stronę www zrobioną z AI i prosty kalkulator wyceny. Pliki leżą na laptopie wykonawcy. Po pół roku trzeba zmienić cennik, a wykonawca ma inne zlecenia i odpisuje po tygodniu. Nikt w firmie nie wie, gdzie jest aktualna wersja ani co i kiedy było zmieniane.

Ta sama historia z GitHubem wygląda inaczej. Repozytorium należy do firmy, wykonawca dostaje do niego dostęp, a każda zmiana ma autora, datę i opis. Gdy współpraca się kończy, odbierasz dostęp w minutę, a cała historia zostaje u Ciebie. To jest profesjonalizm: Twoje zasoby są pod Twoją kontrolą.

Co konkretnie zyskujesz:

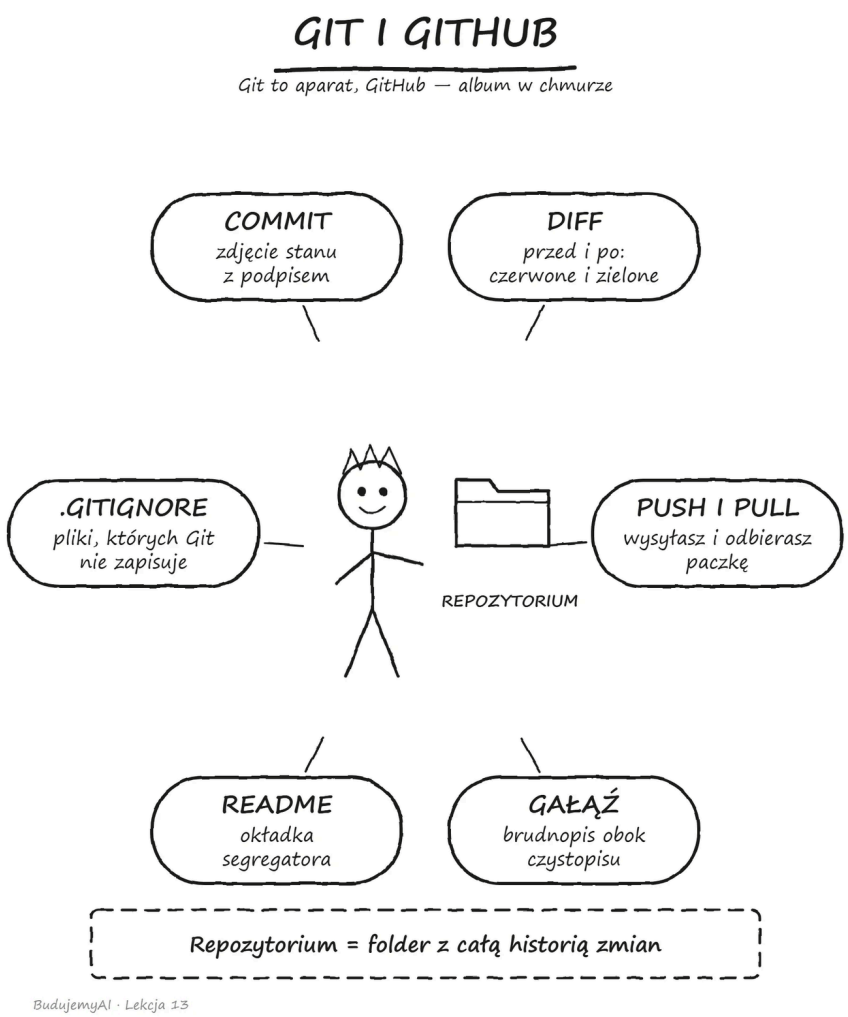
- **Maszynę czasu.** Każda zapisana wersja ma autora, datę i opis. Cofnięcie zmiany to kilka kliknięć.
- **Bezpieczny plac zabaw dla AI.** Agent może działać śmiało, bo każdą jego zmianę widzisz na czerwono i zielono, zanim ją zatwierdzisz.
- **Niezależność od wykonawców i awarii.** Kod strony, automatyzacji czy skryptu jest na koncie firmy w chmurze, a nie na czyimś laptopie.
- **Dostęp do open source.** Na GitHubie żyje większość darmowych narzędzi AI, z których korzystam — od Handy do dyktowania po wtyczki do Claude Code. Więcej w lekcji o topowych repozytoriach.

Uczciwie: GitHub nie zastąpi Dysku Google ani OneDrive. Skany umów, faktury i zdjęcia trzymaj tam, gdzie dziś. GitHub jest najlepszy dla tekstu i kodu, które się zmieniają: promptów, procedur, instrukcji dla AI, plików strony www i skryptów.

> **! RODO: repozytorium to nie baza klientów**
>
> Nie wkładaj do repozytorium danych osobowych klientów ani pracowników: eksportów z CRM, list z numerami PESEL, skanów dokumentów. Przekazanie takich danych zewnętrznemu dostawcy wymaga podstawy prawnej i zwykle umowy powierzenia, a z historii repozytorium trudno cokolwiek usunąć. To nie jest porada prawna — przy wątpliwościach zapytaj prawnika lub inspektora ochrony danych. O umowach powierzenia piszę w lekcji o Gemini i RODO.

ROZDZIAŁ 3

Jak to działa — w prostych słowach



Git to darmowy program, który zapisuje historię zmian w folderze na Twoim komputerze. **GitHub** to serwis w chmurze (należy do Microsoftu), który przechowuje te foldery razem z historią i pozwala nad nimi współpracować. Najprościej: Git to aparat, który robi zdjęcia Twojej pracy, a GitHub to album w chmurze, który pokazujesz wybranym osobom. Podobnie działają GitLab i Bitbucket.

Pojęcie	Analogia	Co to znaczy w praktyce
Repozytorium (repo)	segregator z pamięcią	folder projektu razem z całą historią zmian
Commit	zdjęcie stanu z podpisem	zapisany punkt w historii: co, kto, kiedy i dlaczego
Diff	zakreślacz	porównanie przed i po: usunięte na czerwono, dodane na zielono

Pojęcie	Analogia	Co to znaczy w praktyce
Push i pull	wysłanie i odebranie paczki	wysyłasz swoje commity do GitHuba albo pobierasz cudze
Gałąź (branch)	brudnopis obok czystopisu	wersja robocza, w której testujesz zmiany bez ruszania głównej
Pull request	prośba o przepisanie do czystopisu	„przejrzyj moje zmiany i włącz je do głównej wersji”
Issue	karteczka na tablicy zadań	zgłoszenie błędu, pomysł albo zadanie do zrobienia
README	okładka segregatora	plik z opisem: co tu jest i jak z tego korzystać
.gitignore	bramkarz z listą	lista plików, których Git ma nigdy nie zapisywać
Publiczne i prywatne	witryna i zaplecze	publiczne widzi cały internet, prywatne — tylko Ty i zaproszone osoby

Na co dzień: zmieniasz plik, GitHub Desktop pokazuje diff, wpisujesz krótki opis, klikasz **Commit**, potem **Push origin** — i zmiana jest w chmurze. To samo da się robić w terminalu (czym jest CLI, wyjaśniam w lekcji o CLI, MCP i API), ale na start wystarczy aplikacja z przyciskami.

Dlaczego agenci AI tak dobrze pracują z Gitem? Bo każda ich zmiana staje się diffem, który możesz przejrzeć, zatwierdzić albo cofnąć. Zaawansowani idą dalej: agent pracuje na osobnej gałęzi, a do głównej wersji jego zmiany trafiają po przeglądzie w pull requeście.

Bezpieczeństwo na GitHubie ma trzy warstwy: **konto** (2FA, passkey, kody odzyskiwania), **repozytorium** (prywatne, z plikiem .gitignore i alertami Dependabot) i **dostęp** (kto i czym się łączy: ludzie i tokeny). Wszystkie trzy ustawisz w tej lekcji.

ROZDZIAŁ 4

Ile to kosztuje

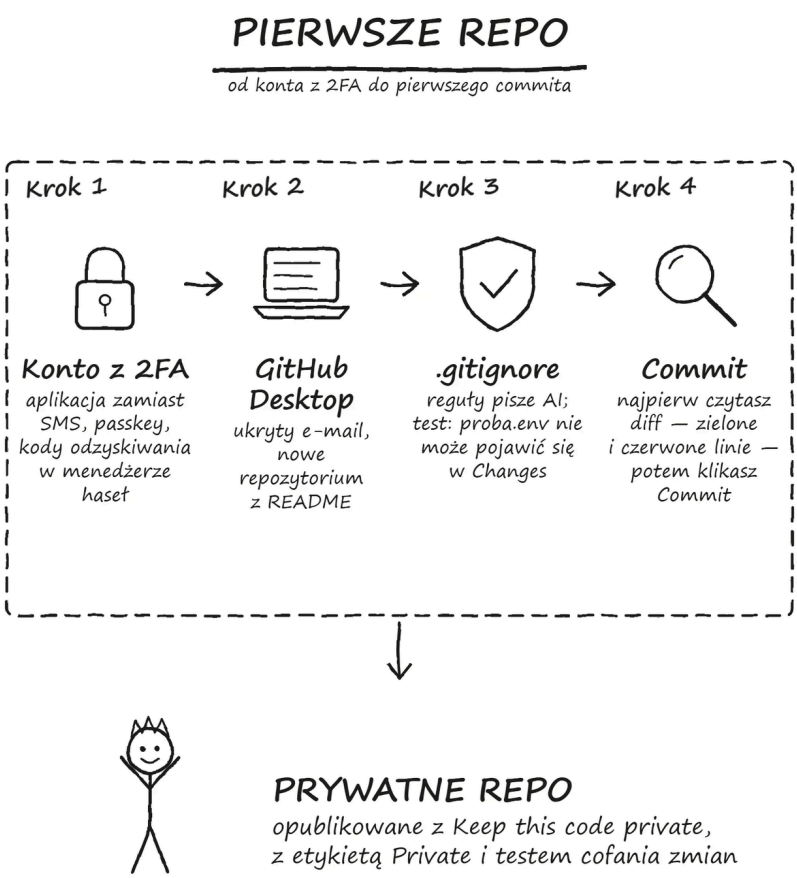
Plan	Cena (stan: październik 2026)	Kiedy wystarczy
GitHub Free — konto osobiste	0 zł	Na start: nielimitowane repozytoria prywatne i współpracownicy, 2 000 minut automatyzacji GitHub Actions miesięcznie
GitHub Free — organizacja	0 zł	Firma z pracownikami lub wykonawcami: wspólne repozytoria i role dostępu
GitHub Team	4 USD za osobę miesięcznie (ok. 16 zł)	Gdy potrzebujesz m.in. ochrony gałęzi i wymaganych recenzji w repozytoriach prywatnych
GitHub Enterprise	od 21 USD za osobę miesięcznie (ok. 82 zł)	Duże organizacje: SSO, audyt, zgodność

Przeliczenia według kursu NBP z 2 października 2026 (1 USD = 3,89 zł). Na stronie cennika stawki Team i Enterprise są oznaczone jako ceny na pierwsze 12 miesięcy — przed zakupem sprawdź cenę odnowienia.

Ważne ograniczenie: na darmowym koncie GitHub skanuje sekrety (hasła, klucze) tylko w repozytoriach **publicznych**. Skanowanie prywatnych to płatny dodatek GitHub Secret Protection — 19 USD miesięcznie za każdą aktywną osobę commitującą, dla organizacji w planach Team i Enterprise. Dlatego Twoją pierwszą linią obrony są plik .gitignore i przegląd zmian przed każdym commitem.

ROZDZIAŁ 5

Krok po kroku



Krok 1: Załóż konto i od razu włącz 2FA

1. Wejdź na github.com i kliknij **Sign up**. Podaj firmowy e-mail, długie hasło wygenerowane w menedżerze haseł i nazwę użytkownika. Nazwa jest publiczna i pojawia się w adresach, więc wybierz coś neutralnego, np. imię i nazwisko.
2. Potwierdź adres e-mail kodem z wiadomości od GitHuba.
3. Kliknij swoje zdjęcie profilowe w prawym górnym rogu → **Settings** → w sekcji „Access" wybierz **Password and authentication** → **Enable two-factor authentication**.
4. Zeskanuj kod QR aplikacją uwierzytelniającą i wpisz sześciocyfrowy kod. GitHub wyraźnie zaleca aplikację zamiast SMS-ów.
5. W części „Save your recovery codes" kliknij **Download**. Zapisz kody w menedżerze haseł (np. jako bezpieczną notatkę), usuń plik z folderu Pobrane i dopiero wtedy kliknij **I have saved my recovery codes**.

6. Na tej samej stronie, w części „Passkeys", kliknij **Add a passkey**. Passkey zastępuje hasło i kod jednym krokiem i jest odporny na fałszywe strony logowania (phishing).

GitHub i tak wymaga 2FA od kont aktywnie współtworzących kod: po powiadomieniu daje 45 dni (plus 7 dni prolongaty), a potem blokuje logowanie bez 2FA. Lepiej mieć to z głowy od pierwszego dnia.

> ✓ **Jedno konto = jedna osoba**
>
> Regulamin GitHuba mówi wprost, że jednego loginu może używać tylko jedna osoba. Nie zakładaj konta „biuro" dla całego zespołu. Każdy ma własne konto, a firma — organizację (krok 8).

Krok 2: Ukryj swój e-mail i zainstaluj GitHub Desktop

1. Na github.com wybierz **Settings** → **Emails** i zaznacz **Keep my email addresses private**. Dzięki temu commity będą podpisane technicznym adresem GitHuba (w domenie users.noreply.github.com), a nie Twoim prawdziwym e-mailem.
2. Pobierz GitHub Desktop ze strony desktop.github.com. Na Windows uruchom pobrany instalator, na Macu rozpakuj plik zip i uruchom aplikację.
3. Na ekranie powitalnym kliknij **Sign in to GitHub.com**. Otworzy się przeglądarka — zaloguj się passkeyem i zatwierdź dostęp dla aplikacji.
4. Ustaw autora commitów. Windows: **File** → **Options** → **Git**. macOS: **GitHub Desktop** → **Settings** → **Git**. W polu **Name** wpisz imię i nazwisko, a z listy **Email** wybierz adres kończący się na users.noreply.github.com.

Krok 3: Utwórz pierwsze repozytorium

1. W GitHub Desktop wybierz **File** → **New repository...**.
2. **Name**: np. `firma-wiedza` — małe litery, myślniki, bez polskich znaków.
3. **Description**: „Prompty, procedury i szablony firmy".
4. **Local path**: folder na dysku, np. `C:\Repozytoria` na Windows albo folder `Repozytoria` w katalogu domowym na Macu. Unikaj folderów synchronizowanych przez OneDrive, Dysk Google czy Dropbox — dwa systemy synchronizacji naraz potrafią sobie przeszkadzać.
5. Zaznacz **Initialize this repository with a README**. Listy **Git ignore** i **License** zostaw na **None** — plik .gitignore przygotujesz w następnym kroku, a licencja nie jest potrzebna w repozytorium prywatnym.
6. Kliknij **Create repository**. GitHub Desktop od razu zapisze automatyczny commit „Initial commit" z plikiem README — zobaczysz go w zakładce **History**. To Twój punkt startowy.
7. Wybierz **Repository** → **Show in Explorer** (na Macu **Show in Finder**) i załóż foldery `prompty`, `procedury` i `szablony`. Do folderu `prompty` wrzuć plik `odpowiedz-na-zapytanie.md` z Twoim najlepszym promptem, np. do odpowiadania na zapytania z formularza. Pliki `.md` to zwykły tekst

(Markdown) — otworzysz je w Notatniku. Git nie zapisuje pustych folderów, więc pozostałe zapełnisz w kolejnych dniach.

Krok 4: Przygotuj z AI plik .gitignore i sprawdź, czy działa

Wklej ten prompt do Claude, ChatGPT lub Gemini i uzupełnij nawiasy kwadratowe:

› PROMPT

```
Jesteś doświadczonym administratorem repozytoriów Git i pomagasz małej firmie.
Przygotuj plik .gitignore dla prywatnego repozytorium „firma-wiedza”. Trzymam w nim pliki
Markdown: prompty, procedury i szablony ofert.
Pracuję na: [Windows / macOS]. Pliki edytuję w: [Notatnik / Word / Obsidian / VS Code].

Wyklucz:
- pliki z sekretami: .env i jego warianty, klucze, certyfikaty, pliki z tokenami,
- pliki systemowe i tymczasowe mojego systemu i edytora,
- eksporty danych (CSV, XLSX) i kopie zapasowe,
- folder dane-klientow/.

Zasady:
- przy każdej regule krótki komentarz po polsku (linia zaczynająca się od #),
- nie wykluczaj plików .md ani folderów prompty/, procedury/, szablony/,
- na końcu podaj 3 nazwy plików testowych, którymi sprawdzę, że reguły działają.
```

1. Przeczytaj odpowiedź: każda reguła ma mieć zrozumiały dla Ciebie komentarz — jeśli nie ma, dopytaj AI.
2. W GitHub Desktop wybierz **Repository** → **Repository settings...** → zakładka **Ignored files**. Wklej treść od AI i kliknij **Save**.
3. Test: w folderze repozytorium utwórz puste pliki `proba.env` i `klienci-export.csv`. Na Windows najprościej w Notatniku: **Zapisz jako**, typ **Wszystkie pliki**, pełna nazwa z rozszerzeniem.
4. Spójrz na zakładkę **Changes** w GitHub Desktop. Plików testowych **nie może** tam być. Jeśli są, reguła ma literówkę albo plik .gitignore nie leży w głównym folderze repozytorium.
5. Usuń pliki testowe.

Na przyszłość: gdy na liście zmian pojawi się plik, którego nie chcesz zapisywać, kliknij go prawym przyciskiem i wybierz **Ignore file (add to .gitignore)** — albo **Ignore all .csv files (add to .gitignore)** dla całego typu plików.

Krok 5: Napisz README, przejrzyj zmiany i zrób pierwszy własny commit

Poproś AI o README:

› PROMPT

```
Napisz plik README.md dla prywatnego repozytorium mojej firmy.
Firma: [branża, miasto, liczba osób].
Repozytorium „firma-wiedza” zawiera: [prompty do obsługi klientów, procedury, szablony ofert].

Sekcje README:
1. Co tu jest (2–3 zdania).
2. Struktura folderów.
3. Jak dodać lub zmienić plik (GitHub Desktop: zmiana → opis → Commit → Push origin).
4. Czego tu NIE trzymamy: hasła, klucze, tokeny, dane osobowe klientów, skany dokumentów.
5. Kto odpowiada za repozytorium i data ostatniego przeglądu.

Pisz prosto, po polsku, bez żargonu. Nie wymyślaj plików, poleceń ani narzędzi, których nie
wymieniłem — w brakujących miejscach wstaw [DO UZUPEŁNIENIA].
```

1. Otwórz `README.md` w Notatniku, zastąp treść tekstem od AI i uzupełnij miejsca [DO UZUPEŁNIENIA].
2. W GitHub Desktop, w zakładce **Changes**, zobaczysz listę plików z polami wyboru. Kliknij każdy plik i przeczytaj diff — dodane linie są na zielono, usunięte na czerwono.
3. Sprawdź listę: tylko README, .gitignore i Twój prompt. Plik, którego się nie spodziewasz, odznacz i dodaj do .gitignore.
4. W polu **Summary** wpisz: „Dodaj README, .gitignore i pierwszy prompt”. W opcjonalnym polu **Description** napisz, dlaczego wprowadzasz zmianę.
5. Kliknij **Commit to main**. Na dole pojawi się „Committed just now” i przycisk **Undo** — jeśli od razu zauważysz błąd, cofniesz nim commit, zanim cokolwiek wyślesz.

Krok 6: Opublikuj repozytorium jako prywatne i przetestuj maszynę czasu

1. Kliknij **Publish repository** na górnym pasku.
2. Sprawdź, czy pole **Keep this code private** jest zaznaczone (domyślnie jest, ale sprawdzaj za każdym razem). Jeśli należysz do organizacji, z listy **Organization** wybierzesz, gdzie trafi repozytorium.
3. Kliknij **Publish repository**.
4. Wybierz **Repository → View on GitHub**. Obok nazwy repozytorium musi być etykieta **Private**.
5. Test maszyny czasu: zmień jedno zdanie w pliku z promptem, zrób commit z opisem „Test: zmiana zdania” i kliknij **Push origin**. Przejdź do zakładki **History**, kliknij ten commit prawym przyciskiem i wybierz **Revert changes in commit**. Powstanie nowy commit, który odwraca zmianę — wyślij go przez **Push origin**. W historii zostają oba wpisy, więc zawsze widać, co się działo.

Dla chętnych: to samo w terminalu. GitHub CLI publikuje repozytorium jednym poleceniem — zamiast przycisku **Publish repository** (na Macu instalacja przez `brew install gh`):

```powershell
winget install --id GitHub.cli --source winget
gh auth login
cd C:\Repozytoria\firma-wiedza
gh repo create firma-wiedza --private --source=. --push
```

Kluczowa jest flaga `--private`. Po instalacji otwórz nowe okno terminala, inaczej polecenie `gh` nie zadziała.

Krok 7: Włącz zabezpieczenia repozytorium i konta

1. W repozytorium na github.com kliknij **Settings**, a w bocznym menu, w sekcji „Security and quality", wybierz **Advanced Security**. Przy **Dependency graph** i **Dependabot alerts** kliknij **Enable**, jeśli są wyłączone — to darmowe. Dependabot przyda się, gdy w repozytorium pojawi się kod z zewnętrznymi bibliotekami, np. strona www: ostrzeże o znanych lukach. Alerty znajdziesz w zakładce **Security and quality**.
2. Na koncie: **Settings** → **Code security** → upewnij się, że **Push protection for yourself** jest włączone. Ta funkcja (domyślnie aktywna) blokuje wysłanie rozpoznanych sekretów do repozytoriów publicznych.
3. Na koncie: **Settings** → **Applications** (sekcja „Integrations"). Przejrzyj autoryzowane aplikacje i przy nieznanych kliknij **Revoke**.

Krok 8: Daj dostęp z głową — ludzie i tokeny

Ludzie. W repozytorium na koncie osobistym: **Settings** → **Collaborators** → **Add people**. Uwaga: w prywatnym repozytorium na koncie osobistym współpracownik zawsze dostaje prawo zapisu — dostępu tylko do odczytu nie ma. Jeśli potrzebujesz różnych poziomów (księgowa czyta, wykonawca zmienia), załóż darmową organizację: **Settings** → **Organizations** → **New organization**, plan Free. Istniejące repozytorium przeniesiesz w jego ustawieniach: **Settings** → sekcja „Danger Zone" na dole strony → **Transfer**. W organizacji masz pięć ról repozytorium:

Rola	Co może	Dla kogo w firmie
Read	czytać i komentować	księgowa, klient, audytor
Triage	porządkować zgłoszenia, bez zmian w plikach	asystentka, koordynator
Write	zmieniać pliki (push)	pracownik, wykonawca
Maintain	zarządzać repozytorium bez operacji niszczących	kierownik projektu
Admin	wszystko, łącznie z usunięciem repozytorium	właściciel (1–2 osoby)

Tokeny. Gdy aplikacja albo automatyzacja potrzebuje dostępu do repozytorium (np. Obsidian na telefonie, skrypt na serwerze), nigdy nie dawaj jej swojego hasła. Utwórz token fine-grained:

1. **Settings → Developer settings → Personal access tokens → Fine-grained tokens → Generate new token**.
2. Nazwa mówi, do czego służy token, np. „obsidian-telefon”.
3. **Expiration**: wybierz datę — 90 dni albo rok. GitHub pozwala też na token bez daty ważności, ale nie korzystaj z tej opcji.
4. **Repository access**: **Only select repositories** → wybierz jedno repozytorium.
5. **Permissions**: tylko to, czego potrzeba — np. **Contents** z dostępem **Read and write** do synchronizacji albo **Read-only**, jeśli aplikacja ma tylko czytać.
6. Kliknij **Generate token**, skopiuj token od razu (GitHub pokazuje go tylko raz) i zapisz w menedżerze haseł. Nigdy nie wklejaj go do czatu z AI, do pliku w repozytorium ani do maila.
7. Wpisz w kalendarz przypomnienie tydzień przed końcem ważności.

GitHub sam zaleca tokeny fine-grained zamiast starszych tokenów „classic”, które dają dostęp do wszystkich Twoich repozytoriów naraz.

ROZDZIAŁ 6

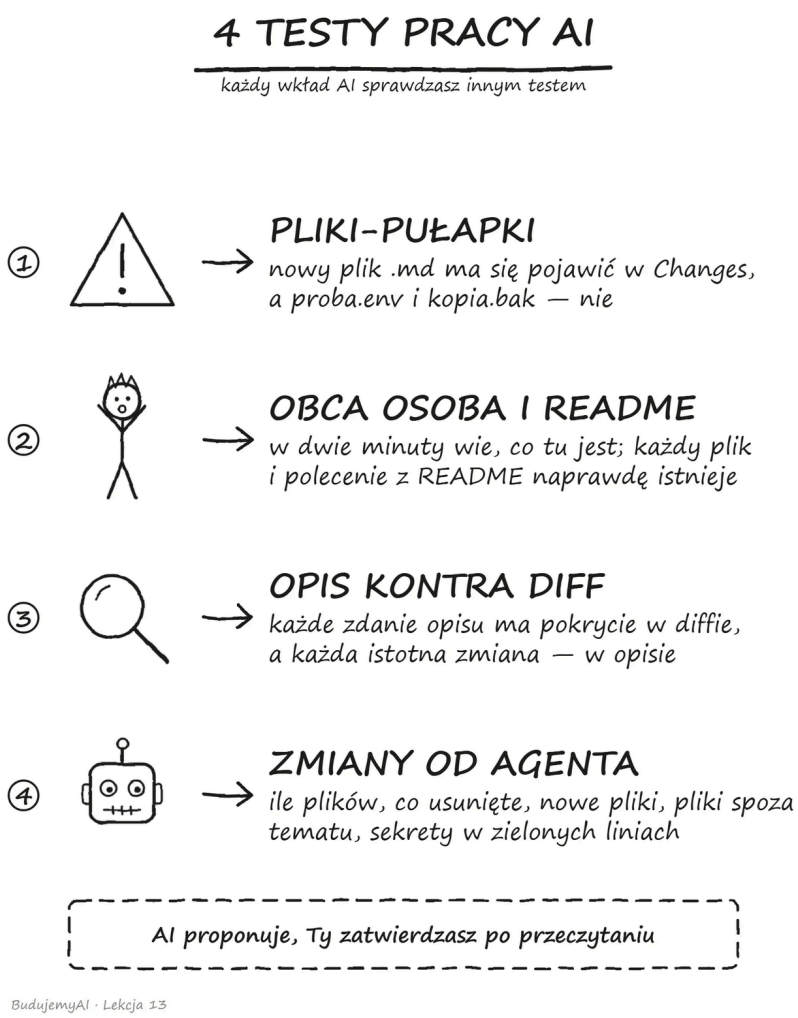
Jak to robię u siebie

Centrum operacyjne mojej firmy to vault Obsidiana — notatki, procedury, plan zadań i plik `CLAUDE.md` z zasadami dla AI. Trzymam go w prywatnym repozytorium na GitHubie. Na komputerze i na serwerze pracuje na nim Claude Code, a na telefonie Obsidian z wtyczką Git.

- **Telefon łączy się tokenem fine-grained** — dostęp tylko do tego jednego repozytorium, tylko potrzebne uprawnienia, ważność na rok i przypomnienie w kalendarzu przed końcem.
- **Plik ustawień wtyczki Git jest w .gitignore.** Ustawienia konkretnego urządzenia nie mają czego szukać w historii repozytorium.
- **Każda zmiana od agenta przechodzi przez diff.** Claude Code robi większość pracy, Codex służy mi za drugą opinię, ale commit zatwierdzam dopiero po przeczytaniu zmian. Więcej o tej pętli w kursie Claude Code, a o pracy z telefonu — w lekcji o Claude Code z telefonu.
- **Strony klientów wdrażają się same po push.** Lekcja z praktyki: gdy po zmianie zapory na serwerze powiadomienia z GitHuba (webhooki) przestały docierać, strona po cichu została na starej wersji. Dlatego po każdym wdrożeniu sprawdzam, czy na stronie naprawdę jest nowa treść.

ROZDZIAŁ 7

Jak rozpoznać, że AI zrobiło dobrą robotę



W tej lekcji AI pomaga Ci w czterech miejscach: pisze .gitignore, README i opisy commitów, a później — jako agent — samo zmienia pliki. Każdy z tych wkładów sprawdzasz innym testem. Zasada wspólna: AI proponuje, Ty zatwierdzasz po przeczytaniu.

Test 1: pliki-pułapki dla .gitignore

Utwórz w repozytorium pliki `proba.env`, `klienci-export.csv` i `kopia.bak`. Żaden nie może pojawić się w zakładce **Changes**. Potem odwrotnie: utwórz nowy plik `.md` w folderze `prompty` — ten musi się pojawić. Jeśli go nie widać, AI wykluczyło za dużo (typowy błąd: reguła `*.md` albo cały folder).

Test 2: obca osoba i polowanie na zmyślenia w README

Daj README na dwie minuty komuś, kto nie zna repozytorium. Ta osoba powinna odpowiedzieć: co tu jest, gdzie dodać nową procedurę i czego nie wolno tu wkładać. Następnie sprawdź każdy plik, folder i polecenie wymienione w README. Jeśli README każe uruchomić `npm install` w repozytorium z samymi plikami tekstowymi, AI to zmyśliło — tak samo jak potrafi zmyślać nazwy pakietów, o czym więcej w lekcji o halucynacjach.

Test 3: opis commita kontra diff

Każde zdanie opisu musi mieć pokrycie w diffie, a każda istotna zmiana w diffie — w opisie. Opis „Popraw literówki" przy diffie, w którym zmieniła się cena, to czerwona flaga, niezależnie od tego, czy opis napisało AI, czy człowiek.

Test 4: zmiany od agenta w pięciu pytaniach

1. Ile plików się zmieniło — tyle, ile wynikało z prośby?
2. Co zostało usunięte? Czerwone linie czytaj najuważniej.
3. Czy pojawiły się nowe pliki, o które nikt nie prosił (konfiguracja, `.env`, eksport)?
4. Czy zmiany dotknęły plików spoza tematu zadania?
5. Czy w zielonych liniach są hasła, klucze, tokeny, adresy e-mail lub numery klientów?

Sygnały ostrzegawcze

Zatrzymaj się, gdy AI proponuje:

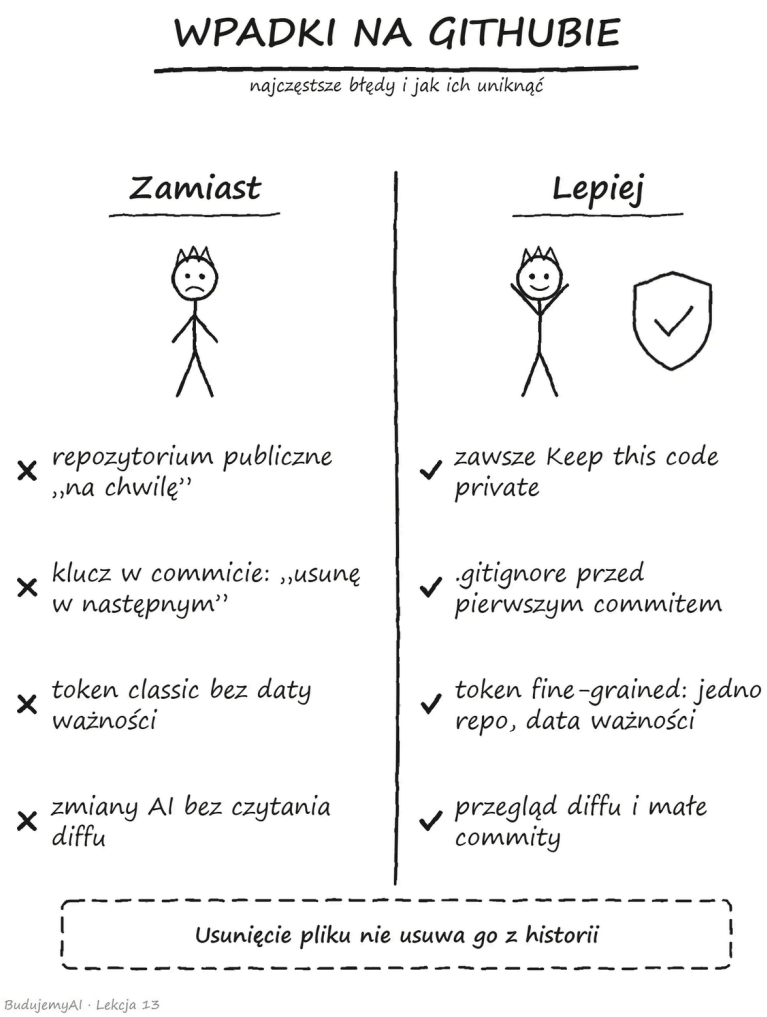
- „Ustaw repozytorium jako publiczne, będzie prościej."
- „Wybierz token bez daty ważności, żeby nie wygasł."
- „Użyj tokena classic z pełnym dostępem do repozytoriów."
- „Dodaj plik .env do repozytorium, żeby zespół miał konfigurację."
- „Wyłącz na chwilę 2FA."
- polecenie z `--force` (np. `git push --force`) bez wyjaśnienia, co nadpisze.

✓ **Checklista przed każdym Push origin**

- [ ] Lista plików w **Changes** zawiera tylko to, co chcesz zapisać
- [ ] Diff każdego pliku przeczytany — zwłaszcza czerwone, usunięte linie
- [ ] Brak .env, kluczy, tokenów, haseł, eksportów CSV/XLSX i danych osobowych
- [ ] Opis commita mówi, co i dlaczego, i zgadza się z diffem
- [ ] README nie wspomina plików ani poleceń, których nie ma w repozytorium
- [ ] Repozytorium ma etykietę **Private**, jeśli nie ma być publiczne

ROZDZIAŁ 8

Najczęstsze błędy



1. **Repozytorium publiczne „na chwilę”.** Skutek: kod, prompty i notatki widzi cały internet, a kopie mogą zostać u innych, nawet gdy później zmienisz ustawienie. Jak uniknąć: zawsze **Keep this code private**; publiczne tylko świadomie, po przeglądzie zawartości.
2. **Klucz lub hasło w commicie z myślą „usunę w następnym”.** Skutek: usunięcie pliku nie usuwa go z historii — klucz wciąż da się odczytać. Jak uniknąć: .gitignore przed pierwszym commitem. Gdy już się stało, najpierw unieważnij klucz i wygeneruj nowy (tak zaleca GitHub), dopiero potem czyść historię.
3. **Konto bez 2FA albo kody odzyskiwania w folderze Pobrane.** Skutek: przejęcie konta albo utrata dostępu po zgubieniu telefonu. Jak uniknąć: 2FA z aplikacją, passkey, kody w menedżerze haseł.
4. **Token classic bez daty ważności.** Skutek: jeden wyciek oznacza dostęp do wszystkich Twoich repozytoriów, bez końca. Jak uniknąć: tylko tokeny fine-grained — jedno repozytorium, minimalne uprawnienia, data ważności.

5. **Zatwierdzanie zmian AI bez czytania diffu.** Skutek: agent po cichu usuwa akapit, zmienia cenę albo dodaje plik z konfiguracją. Jak uniknąć: małe zadania, przegląd diffu, małe commity.
6. **Wspólne konto dla całego zespołu.** Skutek: nie wiadomo, kto co zmienił, odejście pracownika oznacza zmianę dostępu wszystkim, a regulamin GitHuba tego zabrania. Jak uniknąć: każdy ma własne konto, firma — organizację z rolami.
7. **Dane klientów w repozytorium.** Skutek: ryzyko naruszenia RODO, a dane zostają w historii. Jak uniknąć: repozytorium na tekst i kod, dane osobowe w systemach do tego przeznaczonych, eksporty w .gitignore.
8. **Opisy commitów typu „zmiany" albo „poprawki".** Skutek: historia jest bezużyteczna, gdy za pół roku szukasz, kiedy zmienił się cennik. Jak uniknąć: pierwsza linia mówi co, opis — dlaczego.

ROZDZIAŁ 9

Zestaw do powtórzenia

Prompt: opis commita

› PROMPT

```
Na podstawie listy zmian zaproponuj opis commita po polsku.
- Summary: tryb rozkazujący, maksymalnie 60 znaków, co zmieniono (np. „Dodaj procedurę wystawiania faktur w KSeF”).
- Description: 2–4 punkty — co i dlaczego.
Opisuj tylko to, co jest na liście. Jeśli czegoś nie wiesz, zapytaj, zamiast zgadywać.

Zmiany:
[lista plików + 1–2 zdania o każdej zmianie — bez haseł i danych klientów]
```

Prompt: audyt listy plików przed commitem

Ten prompt dostaje tylko nazwy plików, nie ich treść — dzięki temu nic poufnego nie trafia do czatu.

› PROMPT

```
Jesteś audytorem bezpieczeństwa. Zaraz wyślę zmiany do prywatnego repozytorium na GitHubie.
Oto lista nazw plików z zakładki Changes (bez zawartości):
[wklej listę]

1. Wskaż pliki, które mogą zawierać sekrety (hasła, klucze API, tokeny, certyfikaty) albo dane osobowe — z uzasadnieniem.
2. Zaproponuj reguły do .gitignore dla tych plików.
3. Podaj 5 słów lub fraz, których mam poszukać w treści pozostałych plików przed commitem.
Nie proś o wklejenie zawartości plików.
```

Prompt: przegląd zmian od agenta AI

Do wklejenia w sesji Claude Code albo Codex, zanim zgodzisz się na commit.

› PROMPT

```
Zanim cokolwiek zatwierdzisz:
1. Pokaż listę zmienionych plików i w 1–2 zdaniach streść zmiany w każdym — co i dlaczego.
2. Osobno wypisz: usunięte fragmenty, nowe pliki i zmiany w plikach, o które nie prosiłem.
3. Sprawdź, czy w zmianach nie ma haseł, kluczy, tokenów ani danych osobowych.
4. Zaproponuj opis commita (Summary do 60 znaków + 2–4 punkty).
Nie rób commita ani push bez mojej wyraźnej zgody.
```

Szablon .gitignore na start

Jeśli trzymasz w repozytorium szablony w Excelu, usuń linię z `*.xlsx`.

TEKST

```
# Sekrety — nigdy w repozytorium
.env
.env.*
*.key
*.pem
*.p12

# Eksporty i dane klientów
*.csv
*.xlsx
dane-klientow/

# Pliki systemowe
.DS_Store
Thumbs.db
desktop.ini

# Pliki tymczasowe i kopie
*.tmp
*.bak
~$*
```

Checklista: kwartalny przegląd bezpieczeństwa GitHuba

- [ ] 2FA włączone, passkey działa, kody odzyskiwania są w menedżerze haseł
- [ ] Wszystkie firmowe repozytoria mają etykietę Private (poza świadomie publicznymi)
- [ ] Lista współpracowników i członków organizacji aktualna — osoby, które odeszły, usunięte
- [ ] Każdy token ma datę ważności i dostęp tylko do potrzebnych repozytoriów; nieużywane usunięte
- [ ] **Settings** → **Applications**: brak nieznanych aplikacji z dostępem
- [ ] Alerty Dependabot przejrzane w zakładce **Security and quality**
- [ ] Test plików-pułapek dla .gitignore przechodzi
- [ ] README ma aktualną datę przeglądu i osobę odpowiedzialną

Szablon: macierz dostępu

Osoba	Konto GitHub	Repozytorium	Rola	Od	Do	Kto zatwierdził
[imię i nazwisko]	[login]	[nazwa]	[Read / Write / Admin]	[data]	[data lub „do odwołania”]	[osoba]

Osoba	Konto GitHub	Repozytorium	Rola	Od	Do	Kto zatwierdził
[token: do czego]	—	[nazwa]	[Contents: Read-only / Read and write]	[data]	[data wygaśnięcia]	[osoba]

ROZDZIAŁ 10

Twój plan działania

Dziś (60–75 min): kroki 1–6 — konto z 2FA i passkeyem, GitHub Desktop, prywatne repozytorium z README i .gitignore, pierwszy commit i test maszyny czasu.

W tym tygodniu: krok 7 (Dependabot, przegląd aplikacji). Przenieś do repozytorium trzy procedury i pięć najlepszych promptów — każdą zmianę zapisz osobnym commitem z opisem.

W tym miesiącu: krok 8 — wypełnij macierz dostępu, a jeśli pracujesz z wykonawcami, załóż organizację firmy. Zrób pierwszy projekt z agentem AI na repozytorium (start: kurs Claude Code) i dodaj do kalendarza kwartalny przegląd bezpieczeństwa.

ROZDZIAŁ 11

Słowniczek

- **Git** — darmowy program, który zapisuje historię zmian w plikach.
- **GitHub** — serwis w chmurze do przechowywania repozytoriów i współpracy.
- **Repozytorium** — folder projektu z pełną historią zmian.
- **Commit** — zapisany punkt w historii: co się zmieniło, kto, kiedy i dlaczego.
- **Diff** — zestawienie zmian: usunięte linie na czerwono, dodane na zielono.
- **Push i pull** — wysłanie Twoich zmian do GitHuba i pobranie zmian z GitHuba.
- **Gałąź (branch)** — równoległa wersja robocza, w której testujesz zmiany bez ruszania głównej.
- **Pull request** — prośba o przejrzenie zmian i włączenie ich do głównej wersji.
- **.gitignore** — plik z listą plików, których Git ma nie zapisywać.
- **2FA i passkey** — drugi składnik logowania (kod z aplikacji) oraz klucz kryptograficzny, który zastępuje hasło i kod jednym krokiem.
- **Token fine-grained** — klucz dostępu dla aplikacji, ograniczony do wybranych repozytoriów, uprawnień i czasu.
- **Dependabot** — darmowa funkcja GitHuba, która ostrzega o lukach w bibliotekach używanych przez Twój kod.

ROZDZIAŁ 12

Źródła

- GitHub — Pricing — plany i ceny.
- About mandatory two-factor authentication — obowiązek 2FA i terminy.
- Configuring two-factor authentication — włączanie 2FA, kody odzyskiwania.
- About passkeys — passkey zamiast hasła i kodu.
- GitHub security features — zabezpieczenia darmowe i płatne.
- About push protection — blokowanie sekretów przy wysyłce.
- Configuring Dependabot alerts — włączanie Dependabota.
- Managing your personal access tokens — tokeny fine-grained.
- Permission levels for a personal account repository — uprawnienia współpracowników.
- Creating your first repository using GitHub Desktop — pierwszy commit w GitHub Desktop.
- Removing sensitive data from a repository — gdy sekret trafi do repozytorium.
- Claude Code — Checkpointing — punkty kontrolne a kontrola wersji.

Co dalej

NASTĘPNA LEKCJA · 14 z 21

Dzielenie się czatami AI w firmie — wspólna przestrzeń zespołu

Założysz wspólną przestrzeń AI zespołu i spiszesz jednostronicową politykę dzielenia się czatami.

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/github-od-zera/#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.