

PRD-Engine System

ארכיטקטורה מלאה

מדריך ויזואלי שמסביר איך PRD-Engine עובד מאחורי הקלעים

גרסה: | 2.1.0 תאריך: פברואר 2026

1. איך הכל מתחבר

2. כובעים, לא סוכנים נפרדים

3. Sub-Agents (Task Tool)

4. DOC_SOURCE

5. lessons.md

6. checkpoint.json

7. prd-index.json

8. Cross-Review

9. מבנה הקבצים

10. Hooks

11. 18 כללי ברזל

12. Sweet Spot

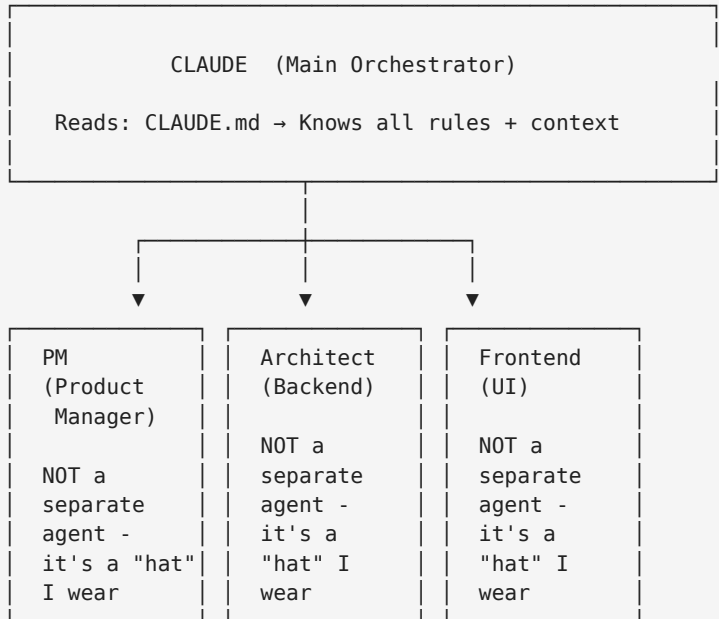
13. מבנה Epic

14. גמישות הוליסטית

15. אפס קצוות פתוחים

16. תרחיש מקצה לקצה

איך הכל מתחבר



עסקיות. כשהוא — Architect שאלות טכניות. כשהוא — Frontend שאלות UI/UX. אבל תמיד אותו. Claude. הרעיון: Claude הוא מוח אחד שמחליף 'כובעים'. כשהוא — PM הוא שואל שאלות

כובעים, לא סוכנים נפרדים

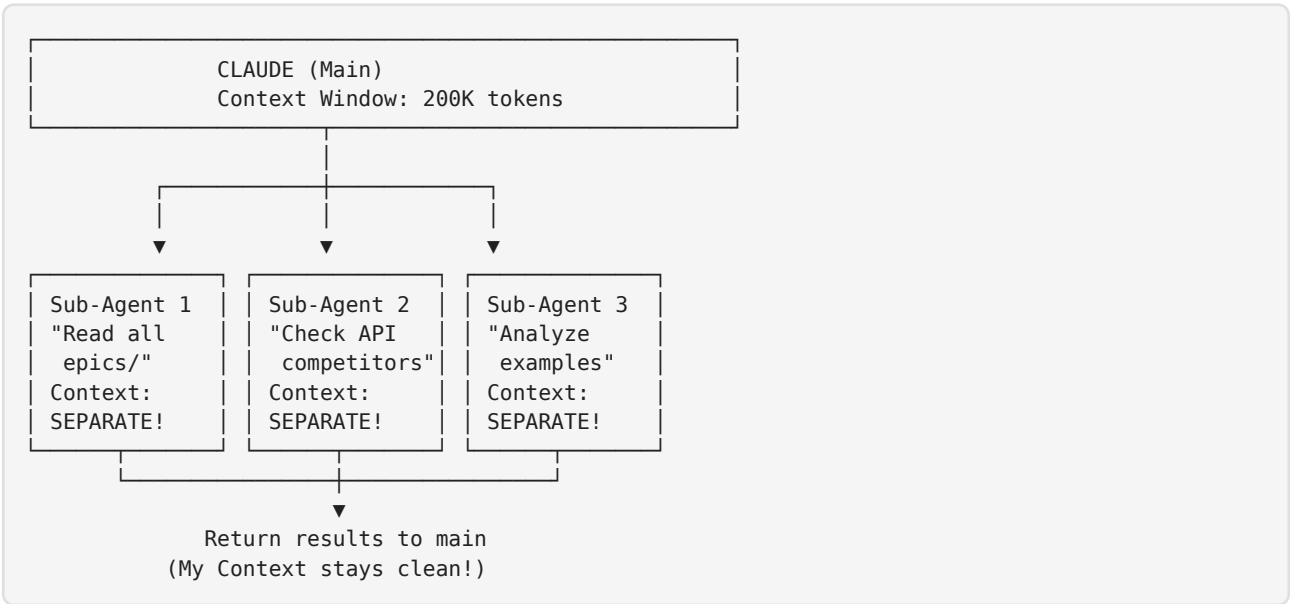
הבנה חשובה: כשאני 'PM' אני לא מפעיל סוכן אחר — אני פשוט קורא את ה-SKILL.md של PM ומתנהג לפיו.

```
User: "Let's spec a Login system"
|
v
Claude reads: product-manager/SKILL.md
|
v
Asks 9 business questions (PM hat)
|
v
"Switching to Architect!"
Claude reads: architect/SKILL.md
|
v
Asks 8 technical questions (Architect hat)
|
v
"Switching to Frontend!"
Claude reads: frontend/SKILL.md
|
v
Asks 11+1 UI questions (Frontend hat)
```

אני זוכר מה PM שאל, ואני יכול לשלב את זה כש-Architect שואל. אין 'העברת מידע' בין סוכנים —הכל אצלי. למה זה חשוב? כי כל ה-context נשאר אצלי.

Sub-Agents (Task Tool)

Sub-Agent הוא כן סוכן נפרד שרץ במקביל —משהו אחר לגמרי מ'כובעים'!



#	תרחיש	סוג	מה עושה
1	Session start / compact	Explore	קורא כל הקבצים + סיכום 60 שורות
2	קריאת DOC_SOURCE	Explore	קורא מסמך גדול + סיכום דרישות
3	בדיקת קישורים בין epics	Explore	בודק + epics חפיפות
4	Cross-Review	general-purpose	מנתח סתירות בין 3 Agents

כלל ברזל: כל סאב-אייגנט חייב — model: 'sonnet' —אף פעם לא Haiku, אף פעם לא Opus!

□ — DOC_SOURCE מסמך המקור

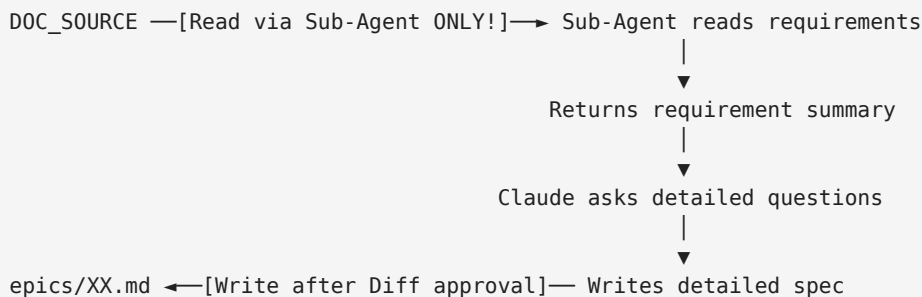
בתחילת כל אפיון חדש, המשתמש מספק קישור למסמך הדרישות שלו (Google Doc, Notion, וכו').

DOC_SOURCE = READ-ONLY!

- Source of truth for requirements
- Never write to it
- Never read in main context (too large!)
- Always read via Sub-Agent

OUTPUT = epics/ files only!

- .claude/memory/epics/XX-name.md
- Written only after Diff approval



3 אופציות בהתחלה (AskUserQuestionTool):

1. 'יש לי קישור' → תדביק ונתחיל.
2. 'אין לי עדיין' → לך ליצור מסמך ותחזור.
3. 'אפיון מאפס' → נשאל שאלות מפורטות בלי מסמך.

□ lessons.md = Long-term Memory

.claude/memory/lessons.md

Lesson 1: Question Format

- Mistake: Asked without options
- Fix: Always use AskUserQuestionTool with numbers

Lesson 2: Google Doc

- Mistake: Wrote to Doc before approval
- Fix: Always show Diff and wait for approval

Lesson 3: User preferences

- Hebrew in conversation, English for tech terms
- Tables with examples

איך זה עובד:

1. אתה מתקן אותי → אני מזהה שזה שיעור.
2. אני מוסיף ל-lessons.md
3. בכל Session חדש → סאב-אייגנט קורא lessons.md
4. לא חוזר על אותה טעות!

שמירה רציפה checkpoint.json —

Claude Code עלול לעשות / compact בכל רגע, או ה-Session עלול להיסגר. בלי שמירה — הכל אבוד!
הבעיה:

```
{
  "timestamp": "2026-02-10T14:30:00",
  "epic": "user-authentication",
  "agent": "architect",
  "question_number": 5,
  "completed": ["Q1: Entities", "Q2: Relations",
               "Q3: APIs", "Q4: Validations"],
  "pending": "Q5: Error Codes",
  "doc_source": "https://docs.google.com/...",
  "notes": "User wants JWT, not sessions"
}
```

מטריצת שמירה — מתי שומרים מה:

אירוע	checkpoint	epic file	prd-index
תשובה משמעותית	☐	☐	☐
סיום שלב Agent	☐	☐	☐
Epic 100%	☐	☐	☐
50% Context	☐ + התראה	☐	☐
PreCompact	☐	—	—

PRD — prd-index.json תכנה

Agent צריך להכיר את כל ה-PRD שכבר נכתב כדי לשאול שאלות חכמות — אבל בלי לשרוף את כל ה-Context! הבעיה: כל

הפתרון: קובץ JSON (קומפקטי) ~ (500 tokens שמכיל את 'המפה' של כל מה שקיים).

```
{
  "epics_completed": 2,
  "epics": {
    "user-auth": {
      "entities": ["User", "Role", "Session"],
      "apis": ["/api/auth/login", "/api/users"],
      "relations": ["User->Role (N:N)"]
    },
    "product-catalog": {
      "entities": ["Product", "Category"],
      "apis": ["/api/products"],
      "relations": ["Product->Category (N:1)"]
    }
  },
  "global_entities": ["User", "Role", "Product"],
  "cross_epic_relations": ["Product->User (created_by)"]
}
```

איך Agent משתמש בזה:

1. prd-index.json → שי' 2 epics: user-auth, product-catalog' ארוק.
2. משתמש: 'בוא נאפיין מערכת הזמנות'
3. Architect שואל שאלה חכמה: 'ראיתי שיש Entity של Product ו-User. האם הזמנה מקושרת אליהם?'
4. זיהוי קישורים אוטומטי: 'ה- epic שפיע על: user-auth, product-catalog'

Cross-Review — 7 Checks

לפני שכותבים לקובץ, epic ועוברות 7 בדיקות חובה:

#	בדיקה	מה בודקים
1	PM Review	כל User Story מכוסה?
2	Architect Review	עקביות טכנית Entities — מתאימים ל-APIs?
3	Frontend Review	כל endpoint מופיע ב-UI?
4	Analytics Events	מינימום 12 events מתועדים
5	SEO Metadata	דומע לכל title, description, og:tags
6	i18n Consistency	כל הודעה בעברית + אנגלית
7	Deferred Docs	מה נדחה — מתועד ומנומק

מבנה הקבצים

```
.claude/
├── CLAUDE.md                <- Brain: 18 Iron Rules
├── settings.json            <- 3 Hooks (SessionStart, PreCompact, Stop)
├── settings.local.json      <- WebFetch permissions
├── memory/
│   ├── checkpoint.json      <- Persistent memory
│   ├── prd-index.json       <- ~200 tokens: where we stopped
│   ├── lessons.md          <- ~500 tokens: PRD map
│   ├── session-init.json    <- Lessons learned
│   ├── epics/
│   │   ├── 01-user-auth.md  <- Sub-agent verification
│   │   └── 02-product-catalog.md <- Dev-ready spec files
├── skills/prd-engine/
│   ├── SKILL.md             <- Engine
│   ├── config/workflow.json <- Main Orchestrator
│   ├── agents/
│   │   ├── product-manager/SKILL.md <- Workflow config (v2.1.0)
│   │   ├── architect/SKILL.md      <- 9 business questions
│   │   └── frontend/SKILL.md       <- 8 technical questions
│   ├── rules/
│   │   ├── 01-questions-format.md  <- 11+1 UI questions
│   │   ├── 02-review-crosscheck.md
│   │   └── 03-reflection.md
│   ├── templates/
│   │   ├── epic-template.md        <- Epic template (Parts A-D)
│   │   ├── checkpoint.json
│   │   └── prd-index.json
│   └── hooks/
│       ├── startup.sh            <- SessionStart
│       ├── pre-compact.sh        <- PreCompact
│       └── auto-checkpoint.sh    <- Stop (Reflection)
```

⚡ — אוטומציה Hooks

3 Hooks שרצים אוטומטית —בלי שהמשתמש צריך לעשות משהו:

Hook	מתי	מה עושה
SessionStart startup.sh	מתחיל Session	מדפיס " PRD-Engine v2.1.0 מוכן!"
PreCompact pre-compact.sh	/compact	timestamp מע checkpoint רמוש
Stop auto-checkpoint.sh	נגמר Session	checkpoint + reflection + lessons רמוש

18 כללי ברזל —סקירה מהירה

#	כלל	בקצרה
0	סאב-אייגנטים	Sonnet + DOC_SOURCE via sub-agent
1	שאלות מובנות	AskUserQuestionTool + options
2	מודולריות	500 שורות מקסימום
3	שמירה רציפה	checkpoint אחרי כל תשובה
4	אפס קצוות פתוחים	כל פרט מוגדר
5	Plan Mode	חובה לפני משימה משמעותית
6	Cross-Review	7 בדיקות חובה
7	Sweet Spot	MVP / Future / User Decides
8	Epics = Dev-Ready	כל epic מוכן לפיתוח
9	Diff לפני כתיבה	הצג שינויים + אישור
10	גמישות הוליסטית	שאלות = SKILL נקודת פתיחה
11	לולאת שיפור	lessons.md
12	קריאת DOC_SOURCE	בכל Session דרך סאב-אייגנט
13	PRD Context	prd-index.json
14	החלפת כובעים	הודעה + קריאת SKILL.md
15	Analytics	12+ events per epic
16	Design System	colors + typography + spacing
17	Reflection	בסוף כל session

Sweet Spot = MVP vs Future

ב-, Architect כל שאלה טכנית מופרדת ל- 3 רמות:

רמה	משמעות	דוגמה
חובה ל-MVP	בלי זה ה- epic לא עובד	Entity: User + email + password
המלצה לעתיד	יחסוך refactor בהמשך	role_history שדה
המשתמש מחליט	יש 2 דרכים — תבחר	JWT or Session-based?

מבנה 4 — Epic חלקים

epics/03-order-system.md

Part A - Business Requirements (PM)

- └ User Stories (2-4)
- └ Acceptance Criteria (8-12)
- └ User Roles Table
- └ Edge Cases / Funnel
- └ KPIs Tables
- └ 2030 Recommendations
- └ Key Decisions

Part B - Technical Architecture (Architect)

- └ Entities (fields, indexes, rules)
- └ Relations (FK behavior)
- └ API Endpoints (Auth + Rate Limit)
- └ Validations (HE + EN)
- └ Error Codes (7 categories)
- └ Logging & Monitoring
- └ Dependencies (3 categories)
- └ 2030 Recommendations
- └ Key Decisions

Part C - Frontend Specification

- └ ASCII Wireframes
- └ Error Display (Inline/Banner/Toast)
- └ Responsive Breakpoints
- └ Accessibility (WCAG AA)
- └ i18n System
- └ Design System
- └ 2030 Recommendations
- └ Key Decisions

Part D - Cross-Review (7 checks)

- └ Analytics Events (12+)
- └ SEO Metadata
- └ i18n Consistency
- └ Review Summary

Key Decisions (All Agents) - Unified Table

ל- → Cursor / Claude Code / Copilot / Windsurf / Bolt המפתח מתחיל לפתח יישורים, בלי שאלות נוספות.
היתרון: המשתמש לוקח את הקובץ → מעביר

גמישות הוליסטית — שאלות חכמות

השאלות ב- SKILL.md הן נקודת פתיחה, לא רשימה סגורה!

SKILL.md = Minimum required + direction
Agent Brain = Goes deeper as needed

Example:

PM asks (from SKILL.md): "Who is the user?"
User answers: "Store manager"

PM continues (from intelligence):

- "Can a store manager manage multiple stores?"
- "Difference between internal and external manager?"
- "Does the manager see all employees or just their team?"

Rules:

- Ask mandatory questions from SKILL.md
- Add your own questions as needed
- Go deeper when there's ambiguity
- Don't ignore mandatory questions
- Don't ask irrelevant questions

4 אפס קצוות פתוחים

כל פרט חייב להיות מוגדר — לא 'יהיה משהו':

לא מספיק	מספיק
"יוצג הודעת שגיאה"	"יוצג: אירעה שגיאה בשמירה. אנא נסה שוב."
"הכפתור יעשה submit"	"לחיצה: POST /api/x → toast spinner / הודעה אדומה"
"validation יהיה"	"שם: חובה, מינ' 2 תווים. אימייל: פורמט. טלפון: 10 ספרות"
"המשתמש יוכל למחוק"	"popup האם למחוק? → כפתור אדום → toastנמחק"

תרחיש מקצה לקצה Workflow —מלא

Session starts

Read CLAUDE.md (iron rules)
Read checkpoint.json (if exists)
Sub-agent reads: SKILL.md + rules/ + lessons
+ prd-index.json + epics/ + DOC_SOURCE

"Found 2 existing epics. Continue?"

Step 0: AskUserQuestionTool (source + target)

PM Stage: 9 business questions
(reads product-manager/SKILL.md)
+ smart follow-ups + Sub-Agent if needed

"Switching to Architect!"

Architect Stage: 8 technical questions
"Found Entity User in prd-index, link or new?"
Sweet Spot: MVP / Future / User Decides

"Switching to Frontend!"

Frontend Stage: 11+1 UI questions
Design System + i18n + Accessibility

Cross-Review: 7 mandatory checks
Contradictions? → Fix → Re-review

Diff → Approval → Write to epics/XX.md

Update checkpoint.json + prd-index.json
"Epic ready for development!"

50% Context? → Save → New Session

