LLM CONTEXT MANAGER

BY IDEO-LAB

FEB 2026

Le concept

Tu sépares 3 choses :

1. **Le projet "réel"** (CDCF, AF, contraintes, décisions, fichiers, livrables, backlog)

2. **Les sessions LLM** (éphémères, jetables)

3. **Un générateur de prompt** (qui recompose *au bon format* une "photo" du projet)

Résultat : quand tu sens la dérive/hallucinations, tu t'en fous → tu ouvres PCM → "Generate Restart Prompt" → tu colles → tu reprends.

Ce que ton outil doit stocker (les “artéfacts”)

AGENDA PROMPT GENERATOR

A) “Base context” (stable)

B) “Plan de référence”

C) “Progress context” (évolutif)

D) “Assets / fichiers”

E) “Prompts”

A) “Base context” (stable)

- Objectif / périmètre
- Contraintes (ex : “English-only in code”, règles UI Ideo-Lab, etc.)
- Glossaire, conventions, stack technique
- Définition of Done (DoD) / critères d’acceptation

B) “Plan de référence”

- CDCF (chapitres)
- AF (user stories / features)
- Architecture (modules, flux, responsabilités)
- Roadmap (phases, jalons)

C) “Progress context” (évolutif)

- Étapes validées (checkpoints)
- Décisions (ADR : Architecture Decision Records)
- TODO / backlog / risques
- Bugs connus + workaround
- “State snapshot” (où on en est exactement)

D) “Assets / fichiers”

- HTML/CSS/JS/Python/SQL (versionnés)
- Liens vers commits Git (ou hash interne)
- Fichiers générés (PDF guides, images, etc.)

E) “Prompts”

- Prompt initial (v1)
- Prompts de reprise (vN)
- Prompts “tâche” (ex : “génère modal #7”, “patch views.py”, etc.)

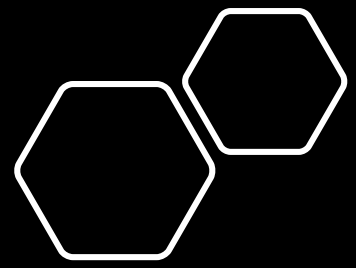
AGENDA MISE EN OEUVRE

- Schéma SQL minimal
- Mécanisme clé : “Checkpoint = photo de reprise”
- Génération du “Restart Prompt”
- UI/UX (simple et efficace)
- Workflow quotidien
- Bonus qui change tout : ADR + “Anti-régression”

Schéma SQL minimal (propre et scalable)

Tables cœur

- `project` : le projet
- `project_doc` : CDCF / AF / Architecture / Glossaire (documents structurés)
- `milestone` : jalons
- `checkpoint` : snapshot validé (date, résumé, état)
- `decision` : ADR (contexte, décision, impact)
- `task` : backlog (status, priorité)
- `asset` : fichiers (type, chemin, hash, version, contenu)
- `prompt_template` : templates de prompt (initial/restart/task)
- `prompt_run` : historique des prompts générés



Notes importantes

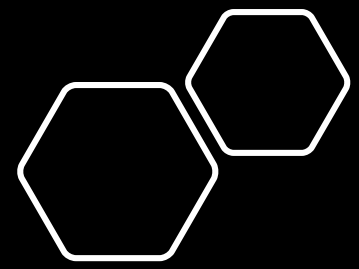
- **Ne stocke pas "tout le chat brut"** comme source de vérité.
- Stocke du **structuré** + un **résumé** + des **hash**.
- Les fichiers : soit en DB (TEXT) si petits, soit en storage (S3/local) avec hash.

Mécanisme clé : “Checkpoint = photo de reprise”

Un checkpoint contient :

- résumé ultra clair (10–20 lignes)
- "facts" (bullet points non ambigus)
- liste des fichiers pertinents + hash + dernière version
- tâches ouvertes + priorités
- décisions actives
- le prompt de reprise généré (ou générable)

Donc même si le LLM part en vrille, **ton checkpoint reste stable.**



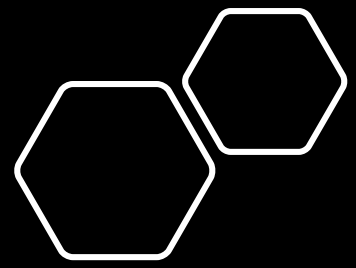
Génération du “Restart Prompt”’’

Le prompt doit être **compact mais complet**. Structure recommandée :

1. **Role / contexte**
2. **Objectif**
3. **Contraintes non négociables**
4. **État actuel (checkpoint)**
5. **Livrables attendus maintenant**
6. **Fichiers à considérer (avec extraits ou hash + contenu si besoin)**
7. **Règles de réponse (patches/diffs, pas de refactor sauvage, etc.)**

Tu peux même générer 2 versions :

- "Restart Prompt – Short" (rapide)
- "Restart Prompt – Full" (quand ça a beaucoup bougé)

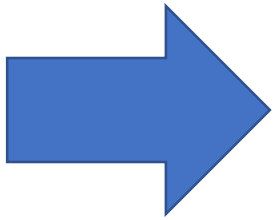


UI/UX (simple et efficace)

UI/UX (simple et efficace)

Dans Ideo-Lab, ça donnerait une page :

- Grid de cards :
  - 📌 Base Context
  - 🧩 CDCF
  - 🧠 Decisions (ADR)
  - ✅ Checkpoints
  - 📁 Assets / Files
  - 📄 Prompt Generator
  - 🗂️ Backlog



Chaque card ouvre une modal :

- liste + édition
- bouton Copy
- bouton Generate Restart Prompt
- bouton Create Checkpoint

Workflow quotidien

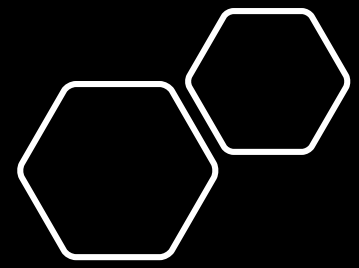
1. Tu démarres un projet → tu remplis **Base context** + **CDCF/AF** (même en brouillon)

2. À chaque “étape validée” :

    - tu cliques “Create Checkpoint”
    - tu attaches/valide les fichiers
    - tu notes 2–3 décisions si besoin

3. Quand la session LLM fatigue :

    - “Generate Restart Prompt (latest checkpoint)”
    - copier/coller dans nouvelle session
    - continuer



Bonus qui change tout : ADR + “Anti-régression”

Ajoute une règle :

- **Chaque décision importante** devient un ADR.

- Les ADR sont injectés dans le prompt de reprise.
  Ça évite exactement ce que tu décris : “le modèle change le code toutes les 5 minutes”.

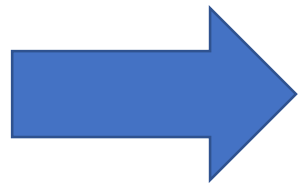
Plan de réalisation

AGENDA PLAN DE REALISATION

- Phase 1 — MVP (1 page + tables)
- Phase 2 — Discipline projet
- Phase 3 — Industrialisation

Phase 1 — MVP (1 page + tables)

- CRUD Project
- CRUD Docs (CDCF/AF)
- CRUD Assets (fichiers)
- Checkpoints
- Prompt Generator (restart)



Phase 2 — Discipline projet

- Decisions (ADR)
- Backlog + status
- Diff d'assets (hash + comparaison simple)
- Export ZIP "Project Snapshot"

Phase 3 — Industrialisation

- Liaison Git (commit/ref)
- Permissions (superuser edit)
- Templates de prompt par type de tâche
- Import/export JSON (backup)

implémentation dans Ideo-Lab

Agenda Implémentation Django/Python

- 1) Django app : structure
- 2) models.py (DB = source de vérité)
- 3) Hash + upsert d’asset
- 4) Générateur de “Restart Prompt”
- 5) Bootstrapping immédiat
- 6) Prochaine étape ,,,,,,,,,

- `models.py` (les tables ci-dessus)
- `admin.py` + pages UI (kcards + modals)
- services : `prompt_builder.py` + `checkpoint_builder.py`
- une commande management : `project_export` (export JSON + assets)
- et un template HTML style IDEO-Lab

2) `models.py` (DB = source de vérité)

```python
# prompt_llm/models.py
from __future__ import annotations

from django.conf import settings
from django.db import models


class Project(models.Model):
    key = models.SlugField(max_length=80, unique=True)
    name = models.CharField(max_length=200)
    description = models.TextField(blank=True, default="")
    is_active = models.BooleanField(default=True)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self) -> str:
        return f"{self.key} - {self.name}"
```

```python
class ProjectDoc(models.Model):
    DOC_TYPES = [
        ("base_context", "Base Context"),
        ("cdcf", "CDCF"),
        ("af", "AF"),
        ("architecture", "Architecture"),
        ("glossary", "Glossary"),
        ("constraints", "Constraints"),
        ("dod", "Definition of Done"),
        ("notes", "Notes"),
    ]

    project = models.ForeignKey(Project, on_delete=models.CASCADE, r
    doc_type = models.CharField(max_length=40, choices=DOC_TYPES)
    title = models.CharField(max_length=200, blank=True, default="")
    content = models.TextField(blank=True, default="")
    version = models.PositiveIntegerField(default=1)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

3) DJANGO ADMIN TOOL

```python
@admin.register(Project)
class ProjectAdmin(admin.ModelAdmin):
    list_display = ("key", "name", "is_active", "updated_at", "creat
    list_filter = ("is_active",)
    search_fields = ("key", "name")
    ordering = ("key",)


@admin.register(ProjectDoc)
class ProjectDocAdmin(admin.ModelAdmin):
    list_display = ("project", "doc_type", "title", "version", "upda
    list_filter = ("doc_type", "project")
    search_fields = ("project__key", "project__name", "doc_type", "t
    ordering = ("project__key", "doc_type", "-version", "-updated_at
    readonly_fields = ("created_at", "updated_at")

    def get_queryset(self, request) -> QuerySet:
        return super().get_queryset(request).select_related("projec
```

```python
@admin.register(Decision)
class DecisionAdmin(admin.ModelAdmin):
    list_display = ("project", "key", "title", "status", "updated_at
    list_filter = ("status", "project")
    search_fields = ("project__key", "key", "title", "context", "dec
    ordering = ("project__key", "status", "key")
    readonly_fields = ("created_at", "updated_at")

    def get_queryset(self, request) -> QuerySet:
        return super().get_queryset(request).select_related("project


@admin.register(Task)
class TaskAdmin(admin.ModelAdmin):
    list_display = ("project", "priority", "status", "title", "updat
    list_filter = ("status", "priority", "project")
    search_fields = ("project__key", "title", "description")
    ordering = ("project__key", "priority", "status", "-updated_at")
    readonly_fields = ("created_at", "updated_at")

    def get_queryset(self, request) -> QuerySet:
        return super().get_queryset(request).select_related("project
```

4) Cron : Management Command

```
def add_arguments(self, parser):
    parser.add_argument("--project", required=True, help="Projec
    parser.add_argument(
        "--checkpoint",
        default="latest",
        help='Checkpoint selector: "latest" or an integer id',
    )
    parser.add_argument(
        "--no-checkpoint",
        action="store_true",
        help="Build prompt from project docs only (ignore checkp
    )
    parser.add_argument(
        "--cache",
        action="store_true",
        help="If using a checkpoint, cache the generated prompt
    )
```

3) Exemples d’utilisation

- Dernier checkpoint (sans cache) :

bash Copier le code

```
python manage.py prompt_llm_restart_prompt --project prompt_llm
```

- Dernier checkpoint + cache dans `generated_prompt` :

bash Copier le code

```
python manage.py prompt_llm_restart_prompt --project prompt_llm --ca
```

- Sans checkpoint (docs only) :

bash Copier le code

```
python manage.py prompt_llm_restart_prompt --project prompt_llm --nc
```

2) UI “grand public” : MVP sérieux (Django)

Objectif : une page **/prompt-llm/** avec ton style Ideo-Lab (cards → modals) :

Pages / modules

1. **Project Dashboard**

- cards : Docs / Assets / Checkpoints / Prompt Generator / Tasks / Decisions

2. **Docs editor**

- texte + versioning (auto increment)
- Save

3. **Assets manager**

- liste des fichiers
- “Open” → textarea editor
- Save (→ `upsert_asset()` en backend)
- (option) “Import from pasted file”

4. **Checkpoints**

- “Create checkpoint” (title/summary/facts/open_items)
- sélection assets à pinner (checkbox)
- bouton “Generate & Cache Prompt”

5. **Prompt Generator**

- bouton “Generate Restart Prompt”
- affichage + bouton **Copy**

3) Prochaine étape immédiate

Tu veux qu'on fasse l'UI **en premier** (logique), donc je te propose de produire maintenant :

- `prompt_llm/urls.py`
- `prompt_llm/views.py`
- `templates/prompt_llm/dashboard.html`
- `templates/prompt_llm/partials/*.html` (modals + fragments HTMX)
- mini CSS hooks (tu brancheras tes classes Ideo-Lab)

Dis juste : "**go UI MVP**" et je te fournis **le code complet** (routes + views + templates) pour :

- créer/choisir un projet
- éditer docs
- créer/éditer assets en DB
- créer checkpoint + pin assets