PLANIFICATION D'EXÉCUTION : Scénario 2 - Adaptation Auto-Claude vers Auto-Gemini-CLI

Version: 1.0
Date: 2 janvier 2026
Équipe: Desjardins | Architect: André-Guy Bruneau
Budget: $13-34K | **Timeline**: 14 semaines (3.5 mois)
Début prévu: 6 janvier 2026 | **Livraison v1.0**: 1er mai 2026

TABLE DES MATIÈRES

1. Structure Exécution
2. Phase 0: Préparation (Semaines 1-2)
3. Phase 1: Swap API (Semaines 3-5)
4. Phase 2: Migration UI (Semaines 6-8)
5. Phase 3: Optimisation Gemini (Semaines 9-12)
6. Phase 4: Intégration & Release (Semaines 13-14)
7. Dépendances & Blockers
8. Métriques de Succès
9. Gestion des Risques
10. Budget Détaillé

STRUCTURE EXÉCUTION

Allocation Ressources

```
Équipe: 1.5 FTE (14 semaines)
├── Lead Architect (1.0 FTE)
│  ├── Design & decision-making
│  ├── Code review & integration
│  └── Stakeholder communication
│
└── Senior Developer (0.5 FTE)
├── Feature implementation
├── Testing & debugging
└── Documentation support
```

Timeline Critique

Phase 0: Préparation Semaines 1-2 (Jan 6-19, 2026)
Phase 1: Swap API Semaines 3-5 (Jan 20 - Feb 9, 2026)
Phase 2: Migration UI Semaines 6-8 (Feb 10 - Mar 2, 2026)
Phase 3: Optimisation Semaines 9-12 (Mar 3 - Mar 30, 2026)
Phase 4: Release Semaines 13-14 (Mar 31 - Apr 13, 2026)

Réserve: 2-3 semaines (buffer pour imprévu)

Total: 14 semaines → v1.0 livraison ~1er mai 2026

Gouvernance

Rôle	Responsable	Fréquence
Project Lead	André-Guy Bruneau	Quotidien
Sprint Reviews	Team + Stakeholders	Bi-hebdomadaire
Gate Decisions	Leadership + Arch	Fin de chaque phase
Risk Management	Project Lead	Hebdomadaire

PHASE 0: PRÉPARATION (Semaines 1-2)

Objectif: Sécuriser ressources, configurer infrastructure, analyser codebase
Deliverables: Repository setup, team onboarding, architectural ADRs
Success Criteria: MVP plan révisé, team 100% ready, no technical unknowns

Semaine 1 (6-12 janvier 2026)

Jour 1-2: Approval & Setup

- [ ] **Tâche 1.1**: Obtenir approbation budgétaire formelle Desjardins
    - **Owner**: André-Guy (Lead Arch)
    - **Deliverable**: Signed approval document
    - **Effort**: 4h (réunions + documentation)
    - **Dépendance**: Sponsor exécutif disponible
- [ ] **Tâche 1.2**: Créer repository GitHub (fork Auto-Claude)
    - **Owner**: André-Guy (Lead Arch)
    - **Deliverable**: `agbruneau/auto-gemini-cli` repository
    - **Effort**: 2h (fork + cleanup)
    - **Steps**:

Create private fork

gh repo fork auto-claude-repo --clone --private

Remove unnecessary files (Electron dependencies)

rm -rf electron/ dist/ build/
git commit -m "chore: remove electron dependencies"

Initialize fresh Git history

git log | head -20 # Verify history preserved

- [ ] **Tâche 1.3**: Créer projet management & backlog
  - **Owner**: André-Guy (Lead Arch)
  - **Deliverable**: GitHub Project avec tous les epics
  - **Effort**: 3h (structure + tagging)
  - **Structure**:
    ```
    GitHub Project: Auto-Gemini-CLI Execution
    ├── Phase 0: Préparation
    ├── Phase 1: API Swap
    ├── Phase 2: UI Migration
    ├── Phase 3: Optimisation
    └── Phase 4: Release
    ```
    Labels:
    - critical (blocks other work)
    - high (important, may block)
    - medium (nice to have)
    - testing (QA/test related)
    - documentation (docs/examples)
    - infrastructure (CI/CD, tools)

Jour 2-3: Codebase Analysis

- [ ] **Tâche 1.4**: Audit architectural Auto-Claude
  - **Owner**: Senior Dev + Lead Arch
  - **Deliverable**: Architecture analysis document
  - **Effort**: 8h (code review + documentation)
  - **Scope**:

Auto-Claude Architecture Audit

Current Structure

- Total LOC: ~[count]
- Main modules:
  - Agent engine: [module_name]
  - Claude SDK integration: [module_name]
  - Session management: [module_name]

        - Git workflows: [module_name]
        - UI/Electron layer: [module_name]

    ## Electron Dependencies

    - Direct: [list]
    - Transitive: [list]
    - Removal risk: [assessment]

    ## Claude-Specific Code

    - Token counting: [lines affected]
    - Streaming: [lines affected]
    - Error handling: [lines affected]
    - Model parameters: [lines affected]

    ## Reusable Modules

    - [module]: 100% reusable
    - [module]: 90% reusable (minor tweaks)
    - [module]: 50% reusable (needs refactor)

    ## Technical Debt

    - [issue]: Severity [HIGH/MEDIUM/LOW]
    - [issue]: Severity [HIGH/MEDIUM/LOW]

- [ ] **Tâche 1.5**: Identifier Gemini API incompatibilities
    - **Owner**: Senior Dev
    - **Deliverable**: Migration checklist
    - **Effort**: 6h
    - **Checklist**:

    # Claude → Gemini API Mapping

    ## Token Counting

    - [ ] Claude: countTokens(prompt)
    - [ ] Gemini: countTokens() method signature
    - [ ] Difference in handling: [document]

    ## Streaming

    - [ ] Claude: stream() event interface
    - [ ] Gemini: streaming in @google/genai v0.4
    - [ ] Adapter pattern needed: YES/NO

    ## Error Handling

    - [ ] Claude error codes: [list]
    - [ ] Gemini error codes: [list]
    - [ ] Mapping required: [details]

Rate Limiting

- [ ] Claude: per-model limits
- [ ] Gemini: 1000 req/24h free tier
- [ ] Queue system: Design required

Model Parameters

- [ ] temperature, topK, topP, etc.
- [ ] Direct compatibility: YES/NO
- [ ] Parameter translation needed: [list]

Function Calling

- [ ] Claude: tool_use blocks
- [ ] Gemini: function_calling (different syntax)
- [ ] Refactor required: [scope]

Jour 4-5: Team & Tooling

- [ ] **Tâche 1.6**: Team onboarding & Git setup
  - **Owner**: Lead Arch
  - **Deliverable**: Team setup complete
  - **Effort**: 4h
  - **Checklist**:

Team Onboarding

- [ ] Access to repository granted
- [ ] Desjardins VPN/network configured
- [ ] GitHub CLI setup (gh auth login)
- [ ] Local environment config:
  - Node 20+ installed
  - TypeScript 5+ installed
  - Oclif CLI installed globally
  - Gemini API key set (GEMINI_API_KEY env var)
- [ ] IDE setup: VS Code/Cursor with extensions
  - TypeScript language server
  - ESLint + Prettier
  - GitHub Copilot (optional)
- [ ] SSH keys configured for git

- [ ] **Tâche 1.7**: Setup CI/CD pipeline
  - **Owner**: Lead Arch
  - **Deliverable**: GitHub Actions workflows
  - **Effort**: 4h
  - **Workflows to create**:

.github/workflows/test.yml

```
name: Unit & Integration Tests
on: [push, pull_request]
jobs:
test:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v4
- uses: actions/setup-node@v4
with:
node-version: '20'
- run: npm ci
- run: npm run test
- run: npm run lint
```

.github/workflows/e2e.yml

```
name: E2E Tests (Gemini)
on: [push]
jobs:
e2e:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v4
- uses: actions/setup-node@v4
- run: npm ci
- run: npm run test:e2e
env:
GEMINI_API_KEY: ${{ secrets.GEMINI_API_KEY }}
```

Semaine 2 (13-19 janvier 2026)

Jour 6-7: Architecture & Design

- [ ] **Tâche 2.1**: Write Architecture Decision Records (ADRs)
  - **Owner**: Lead Arch
  - **Deliverable**: 3-5 ADRs in `/docs/adr/`
  - **Effort**: 6h
  - **ADRs to document**:

    ## ADR-001: Why Oclif instead of Yargs/Commander

    - Context: Need native CLI framework for Gemini
    - Decision: Use Oclif (Salesforce-backed, TypeScript-first)
    - Rationale: Auto-Claude uses similar patterns
    - Consequences: Learning curve minimal, ecosystem strong

ADR-002: Session Persistence Strategy

- Context: Need to persist sessions (SQLite vs JSON)
- Decision: Keep SQLite (from Auto-Claude)
- Rationale: Proven, ACID, supports multi-session
- Consequences: SQLite dependency required

ADR-003: Rate Limiting Architecture

- Context: 1000 req/24h Gemini limit
- Decision: Token-bucket queue + persistent state
- Rationale: Fair distribution, survives restarts
- Consequences: Redis or SQLite-backed queue needed

ADR-004: Token Counter Validation

- Context: countTokens() API differs Claude vs Gemini
- Decision: Wrapper + unit tests with known inputs
- Rationale: Prevents context window overflows
- Consequences: Must validate against real Gemini API

ADR-005: Error Recovery Patterns

- Context: Gemini SDK may have breaking changes
- Decision: Adapter pattern for API layer
- Rationale: Decouples business logic from SDK
- Consequences: Extra abstraction layer (+10% complexity)

- [ ] **Tâche 2.2**: Create detailed Phase 1 spec
  - **Owner**: Lead Arch
  - **Deliverable**: `/docs/phase-1-spec.md`
  - **Effort**: 5h
  - **Content**:

Phase 1 Detailed Specification

Objective

Replace Claude SDK with Gemini SDK, verify basic functionality works

In-Scope

1. Gemini SDK integration (@google/genai v0.4)
2. Basic model.generateContent() parity
3. Token counter validation
4. Error mapping (Claude codes → Gemini codes)

Out-of-Scope

- UI/CLI layer (Phase 2)
- Performance optimization (Phase 3)
- Multi-model support

Acceptance Criteria

- [ ] Single task executes end-to-end with Gemini
- [ ] countTokens() validates context windows
- [ ] At least 80% of existing tests pass
- [ ] No blocking errors in E2E flow
- [ ] Documented API differences

Technical Details

SDK Migration

Before (Claude):

```
import { Anthropic } from "@anthropic-ai/sdk";
const client = new Anthropic({ apiKey: process.env.CLAUDE_API_KEY });
const response = await client.messages.create({ ... });
```

After (Gemini):

```
import { GoogleGenAI } from "@google/genai";
const client = new GoogleGenAI({ apiKey: process.env.GEMINI_API_KEY });
const response = await client.models.generateContent({ ... });
```

File Structure Changes

- Create: `src/adapters/gemini-api-adapter.ts`
- Refactor: `src/services/llm-service.ts` (abstraction layer)
- Delete: `src/integrations/claude-specific/*`
- Update: All tests to use adapter

Testing Strategy

- Unit tests: Adapter layer (mocked Gemini API)
- Integration: Mocked responses (don't hit real API yet)
- E2E: Real Gemini API (phase 1 gate decision)

Jour 8-10: Knowledge Transfer & Planning

- [ ] **Tâche 2.3**: Knowledge transfer from Auto-Claude author
    - **Owner**: Senior Dev (receiver)
    - **Deliverable**: Meeting notes + recorded session
    - **Effort**: 4h
    - **Topics to cover**:
        - Architecture decisions made in Auto-Claude
        - Known pain points & technical debt
        - Why certain patterns were chosen
        - Future roadmap intentions
        - Gotchas/tricks not documented
- [ ] **Tâche 2.4**: Finalize detailed task breakdown
    - **Owner**: Lead Arch
    - **Deliverable**: Updated GitHub Project backlog

    - **Effort**: 4h
    - **Output**: Every task has:
        - Clear acceptance criteria
        - Effort estimate (days, not story points)
        - Assigned owner
        - Blocking dependencies identified
        - Test strategy defined
- [ ] **Tâche 2.5**: Setup monitoring & observability
    - **Owner**: Lead Arch
    - **Deliverable**: Logging + error tracking configured
    - **Effort**: 3h
    - **Tools**:

Observability Stack

Logging

- Framework: Winston or Pino
- Levels: debug, info, warn, error
- Output: stdout + file (for debugging)

Error Tracking

- Tool: Sentry (free tier) or GCP Error Reporting
- Integration: Auto-catch unhandled errors
- Notifications: Slack #auto-gemini-errors channel

Performance Monitoring

- Framework: Native Node.js perf hooks
- Key metrics:
    - API response time (Gemini)
    - Token counting duration
    - Session load time
    - Memory usage

Health Checks

- Endpoint: `--health-check` command
- Metrics: Can call Gemini API, can access DB, can read config

Semaine 2 Closure

- [ ] **Tâche 2.6**: Pre-Phase-1 checkpoint
    - **Owner**: Lead Arch
    - **Deliverable**: Go/No-Go decision
    - **Effort**: 2h
    - **Checkpoint questions**:

Pre-Phase-1 Gate

- [ ] Do we understand Auto-Claude architecture fully?
- [ ] Have we identified all Gemini API incompatibilities?
- [ ] Is the team 100% ready technically?
- [ ] Do we have clear acceptance criteria for Phase 1?
- [ ] Are there any unknown blockers?
- [ ] Budget and timeline still realistic?

Decision: ☐ GO (proceed to Phase 1) ☐ NO-GO (re-plan)
If NO-GO:

- Identified blocker: [description]
- Time impact: [days/weeks]
- Mitigation: [strategy]

PHASE 1: SWAP API (Semaines 3-5)

Objectif: Remplacer Claude SDK par Gemini SDK, validée fonctionnalité basique
Livrable: API layer complète, tests > 80% passing, MVP end-to-end
Succès: Single task fonctionne avec Gemini, no compilation errors

Semaine 3 (20-26 janvier 2026)

Jour 11: Foundation & Gemini SDK Integration

- [ ] **Tâche 3.1**: Créer adapter pattern pour LLM
    - **Owner**: Senior Dev
    - **Deliverable**: `src/adapters/gemini-api-adapter.ts`
    - **Effort**: 6h
    - **Code structure**:

```
// src/adapters/types.ts
export interface LLMAdapter {
generateContent(request: GenerateRequest): Promise<GenerateResponse>;
countTokens(request: CountTokensRequest): Promise<CountTokensResponse>;
startChat(config: ChatConfig): Promise<ChatSession>;
}
export interface GenerateRequest {
prompt: string;
systemPrompt?: string;
model: string;
temperature?: number;
maxOutputTokens?: number;
}
export interface GenerateResponse {
content: string;
inputTokens: number;
outputTokens: number;
finishReason: 'stop' | 'max_tokens' | 'safety' | 'unknown';
}
// src/adapters/gemini-api-adapter.ts
import { GoogleGenAI } from "@google/genai";
```

```
export class GeminiAPIAdapter implements LLMAdapter {
private client: GoogleGenAI;
constructor(apiKey: string) {
this.client = new GoogleGenAI({ apiKey });
}
async generateContent(request: GenerateRequest):
Promise<GenerateResponse> {
const response = await this.client.models.generateContent({
model: request.model || "gemini-2.0-flash",
contents: request.prompt,
config: {
temperature: request.temperature,
maxOutputTokens: request.maxOutputTokens,
},
});
```

```
return {
  content: response.text || "",
  inputTokens: response.usage?.inputTokens || 0,
  outputTokens: response.usage?.outputTokens || 0,
  finishReason: this.mapFinishReason(response.candidates?.[0]?.fini
};
```

```
}
async countTokens(request: CountTokensRequest):
Promise<CountTokensResponse> {
const response = await this.client.models.countTokens({
model: request.model || "gemini-2.0-flash",
contents: request.prompt,
});
```

```
return {
  totalTokens: response.totalTokens,
};
```

```
}
private mapFinishReason(geminiReason?: string): string {
const mapping: Record<string, string> = {
'FINISH_REASON_STOP': 'stop',
'FINISH_REASON_MAX_TOKENS': 'max_tokens',
'FINISH_REASON_SAFETY': 'safety',
};
return mapping[geminiReason || ''] || 'unknown';
}
}
// src/services/llm-service.ts (refactored)
export class LLMService {
```

```
constructor(private adapter: LLMAdapter) {}
async chat(prompt: string, context?: ConversationContext): Promise<string> {
const response = await this.adapter.generateContent({
prompt,
systemPrompt: context?.systemPrompt,
model: context?.model || "gemini-2.0-flash",
});
```

```
return response.content;
```

```
}
async validateContext(prompt: string): Promise<boolean> {
const tokens = await this.adapter.countTokens({
prompt,
model: "gemini-2.0-flash",
});
return tokens.totalTokens < 1_000_000; // 1M token window
}
}
```

- [ ] **Tâche 3.2**: Update package.json dependencies
  - **Owner**: Senior Dev
  - **Deliverable**: Updated `package.json`
  - **Effort**: 1h
  - **Changes**:

```
{
"dependencies": {
"remove": ["@anthropic-ai/sdk"],
"add": ["@google/genai@^0.4.0"],
"keep": ["oclif (or similar)", "sqlite3", "commander or yargs"]
},
"devDependencies": {
"keep": ["typescript@5+", "vitest", "eslint", "prettier"]
}
}
npm uninstall @anthropic-ai/sdk
npm install @google/genai@^0.4.0
npm install # Reinstall lock file
```

- [ ] **Tâche 3.3**: Environment config & secrets
  - **Owner**: Lead Arch
  - **Deliverable**: `.env.example` + config module
  - **Effort**: 2h
  - **File: `src/config/environment.ts`**:

```
import dotenv from 'dotenv';
dotenv.config();
export const config = {
gemini: {
apiKey: process.env.GEMINI_API_KEY || '',
model: process.env.GEMINI_MODEL || 'gemini-2.0-flash',
maxTokens: parseInt(process.env.GEMINI_MAX_TOKENS || '1000000'),
},
app: {
logLevel: process.env.LOG_LEVEL || 'info',
dataDir: process.env.DATA_DIR || './data',
},
};
// Validation
if (!config.gemini.apiKey) {
throw new Error('GEMINI_API_KEY not set');
}
```

Jour 12-13: Basic Implementation

- [ ] **Tâche 3.4**: Implement GeminiAPIAdapter fully
    - **Owner**: Senior Dev
    - **Deliverable**: Complete adapter with streaming support
    - **Effort**: 8h
    - **Features**:
        - [x] generateContent() (unstreamed)
        - [x] generateContent() (streamed)
        - [x] countTokens()
        - [x] Error handling & mapping
        - [x] Rate limit handling (basic)
- [ ] **Tâche 3.5**: Create unit tests for adapter
    - **Owner**: Senior Dev
    - **Deliverable**: `tests/adapters/gemini-api-adapter.test.ts`
    - **Effort**: 6h
    - **Test coverage**:

```
describe('GeminiAPIAdapter', () => {
describe('generateContent', () => {
it('should return valid response for simple prompt', async () => {
// Mock Gemini API response
const response = await adapter.generateContent({
prompt: 'Hello world',
});
expect(response.content).toBeDefined();
expect(response.inputTokens).toBeGreaterThan(0);
});
```

```
it('should handle streaming responses', async () => {
  // Verify stream chunks accumulate
});

it('should map Gemini errors correctly', async () => {
  // Test error mapping
});
```

```
});
```

describe('countTokens', () => {
it('should accurately count tokens for known inputs', async () => {
const response = await adapter.countTokens({
prompt: 'Hello',
});
expect(response.totalTokens).toBe(2); // ~2 tokens for "Hello"
});
});
});

Semaine 4 (27 janvier - 2 février 2026)

Jour 14-15: Refactor Core Logic

- [ ] **Tâche 4.1**: Refactor Agent Engine for Gemini
  - **Owner**: Senior Dev + Lead Arch
  - **Deliverable**: Updated agent logic, tests passing
  - **Effort**: 10h
  - **Scope**:
    - Replace all `client.messages.create()` calls with adapter
    - Adapt function calling syntax (Claude `tool_use` → Gemini `function_calling`)
    - Handle streaming differences
    - Update session persistence
- [ ] **Tâche 4.2**: Update error handling & recovery
  - **Owner**: Senior Dev
  - **Deliverable**: Error handling module
  - **Effort**: 5h
  - **Error types to handle**:
    enum GeminiErrorType {
    AUTH_ERROR, // Invalid API key
    RATE_LIMIT_EXCEEDED, // 1000 req/24h hit
    INVALID_PROMPT, // Content safety block
    CONTEXT_TOO_LARGE, // > 1M tokens
    API_ERROR, // 5xx errors
    NETWORK_ERROR, // Timeout, DNS, etc.
    }
    // Recovery strategies
    interface ErrorRecovery {
    AUTH_ERROR: () => retry(0), // Don't retry
    RATE_LIMIT_EXCEEDED: () => queue(12h), // Queue for later
    INVALID_PROMPT: () => handle(refine), // Ask user to refine
    CONTEXT_TOO_LARGE: () => split(), // Chunk the prompt
    API_ERROR: () => retry(exponential), // Exponential backoff
    NETWORK_ERROR: () => retry(exponential),
    }
- [ ] **Tâche 4.3**: Run existing test suite
  - **Owner**: Senior Dev
  - **Deliverable**: Test report (aim for 70%+ passing)
  - **Effort**: 4h

    - **Command**:
      ```
      npm run test 2>&1 | tee test-report.txt
      ```
      ## Expected output:
      ## ✓ Tests: 150 passed, 45 failed (70% pass rate)
      ## ✓ No critical compilation errors
      ## ✗ Failed tests: [list related to Claude/Gemini differences]

Jour 16-17: Integration Testing

- [ ] **Tâche 4.4**: Create integration tests (mocked API)
    - **Owner**: Senior Dev
    - **Deliverable**: `tests/integration/gemini-integration.test.ts`
    - **Effort**: 6h
    - **Test scenarios**:
      ```
      describe('Integration: Agent with Gemini', () => {
      // Mock Gemini responses
      const mockGenAI = new MockGeminiAPI();
      it('should execute single task end-to-end', async () => {
      const result = await agent.executeTask('write hello world', {});
      expect(result.success).toBe(true);
      expect(result.output).toContain('hello');
      });
      it('should maintain session context across calls', async () => {
      const session = new Session();
      await agent.chat('my name is Bob', { session });
      const response = await agent.chat('what is my name?', { session });
      expect(response).toContain('Bob');
      });
      it('should handle rate limiting gracefully', async () => {
      mockGenAI.setRateLimitAfter(10);
      // Execute 15 tasks
      // Expect: first 10 succeed, 11-15 queued
      });
      });
      ```

Semaine 5 (3-9 février 2026)

Jour 18: Real API Testing (Optional)

- [ ] **Tâche 5.1**: Optional E2E test against real Gemini API
  - **Owner**: Senior Dev
  - **Deliverable**: E2E test results
  - **Effort**: 4h (optional, may skip if time-constrained)
  - **Caution**: Uses Gemini free tier quota
  - **Test**:

    # Set real API key

    export GEMINI_API_KEY="..."
    npm run test:e2e

    # Expected: Single task completes successfully

Jour 19-20: Phase 1 Validation

- [ ] **Tâche 5.2**: Code review & cleanup
  - **Owner**: Lead Arch
  - **Deliverable**: Code review checklist signed off
  - **Effort**: 4h
  - **Checklist**:

    # Code Review Phase 1

    - [ ] All adapter methods implemented
    - [ ] Error handling comprehensive
    - [ ] Tests cover happy path + error cases
    - [ ] No console.log() (use logger)
    - [ ] Type safety: no `any` types
    - [ ] Comments on complex logic
    - [ ] Package.json dependencies pinned
    - [ ] No secrets in code (use env vars)
    - [ ] Performance baseline established
- [ ] **Tâche 5.3**: Documentation update
  - **Owner**: Senior Dev
  - **Deliverable**: `/docs/MIGRATION_CLAUDE_TO_GEMINI.md`
  - **Effort**: 3h
  - **Content**:

    # Migration Guide: Claude → Gemini

What Changed

API Calls

Feature	Claude	Gemini	Adapter
Content generation	messages.create()	models.generateContent()	✓ Abstracted
Token counting	countTokens()	countTokens()	✓ Same signature
Streaming	stream()	streaming in config	✓ Handled
Function calling	tool_use blocks	function_calling	✓ Adapted

Known Differences

1. **Context Window**: 200K (Claude) → 1M (Gemini)
2. **Rate Limits**: Per-model (Claude) → 1000 req/24h free tier (Gemini)
3. **Error codes**: Different enum values
4. **Streaming format**: Different chunk structure

Implementation Notes

- Adapter pattern decouples SDK from business logic
- All LLM calls go through `LLMService`
- Tests mock the adapter, not Gemini API directly

Future Work

- [ ] Support for other Gemini models (Gemini 2.0 Pro, etc.)
- [ ] Multi-model support (Claude + Gemini)
- [ ] Caching integration (Gemini Cache API)

- [ ] **Tâche 5.4**: Phase 1 Gate Decision
  - **Owner**: Lead Arch
  - **Deliverable**: Go/No-Go for Phase 2
  - **Effort**: 2h
  - **Gate criteria**:

Phase 1 Gate Criteria

✓ Acceptance (GO to Phase 2) if:
- [ ] At least 1 end-to-end task works with Gemini
- [ ] 70%+ existing tests pass
- [ ] countTokens() validated (matches Gemini API)
- [ ] Error handling covers 80%+ of cases
- [ ] No blocking technical issues identified

✗ Re-planning (NO-GO) if:
- [ ] Critical Gemini API incompatibility discovered
- [ ] Performance unacceptable (> 2x Claude)
- [ ] Streaming broken or inconsistent
- [ ] Token counting unreliable

Decision: ☐ GO ☐ NO-GO
If GO: Proceed immediately to Phase 2
If NO-GO: Pivot criteria or timeline adjustment

PHASE 2: MIGRATION UI (Semaines 6-8)

Objectif: Remplacer Electron par Oclif, créer CLI avec TUI moderne
Livrable: CLI fully fonctionnel, feature parity avec Auto-Claude
Succès: User peut exécuter workflows via CLI, TUI responsive

Semaine 6 (10-16 février 2026)

Jour 21-22: Oclif Setup & Architecture

- [ ] **Tâche 6.1**: Initialize Oclif framework
  - **Owner**: Senior Dev
  - **Deliverable**: Oclif scaffolding created
  - **Effort**: 3h
  - **Steps**:

    # Initialize oclif in existing project

    npx oclif init

    # This creates:

    # ├── src/commands/ (CLI commands)

    # ├── src/index.ts (Entry point)

    # ├── bin/run.js (Binary wrapper)

    # └── oclif.json (Config)

- [ ] **Tâche 6.2**: Design CLI command structure
  - **Owner**: Lead Arch
  - **Deliverable**: Command hierarchy document
  - **Effort**: 4h
  - **CLI structure**:

Auto-Gemini-CLI Command Structure

```
auto-gemini
├── init # Initialize new workspace
│ └── --name # Workspace name
├── task # Create & execute task
│ ├── new # Define new task
│ ├── run ID # Execute task
│ ├── list # List tasks
│ └── delete ID # Remove task
├── chat # Interactive chat mode
│ ├── --session # Resume session
│ └── --model # Override model
├── config # Manage configuration
│ ├── set KEY VALUE # Set config
│ ├── get KEY # Get config
│ └── show # Show all config
├── session # Manage sessions
│ ├── list # List all sessions
│ ├── show ID # Show session details
│ ├── clear ID # Clear session
│ └── export ID # Export session history
├── status # Show system status
│ └── --watch # Live status
├── logs # View execution logs
│ ├── tail # Stream logs
│ ├── grep PATTERN # Search logs
└── help # Show help
```

Examples:

```
$ auto-gemini task new "write a CLI tool"
$ auto-gemini task run <task-id>
$ auto-gemini chat
$ auto-gemini config set model gemini-2.0-pro
```

- [ ] **Tâche 6.3**: Create base command classes
  - **Owner**: Senior Dev
  - **Deliverable**: Base command architecture
  - **Effort**: 4h
  - **File: src/base-command.ts**:

```
import { Command, Flags } from '@oclif/core';
import { logger } from './utils/logger';
import { config } from './config/environment';
export abstract class BaseCommand extends Command {
protected logger = logger;
protected config = config;
// Common flags
static baseFlags = {
debug: Flags.boolean({
description: 'Enable debug logging',
default: false,
}),
'config-dir': Flags.string({
description: 'Override config directory',
default: config.app.dataDir,
}),
};
async init(): Promise<void> {
if (this.flags.debug) {
this.logger.setLevel('debug');
}
}
}
```

Jour 23-24: Core Commands Implementation

- [ ] **Tâche 6.4**: Implement `init` command
  - **Owner**: Senior Dev
  - **Deliverable**: `src/commands/init.ts`
  - **Effort**: 4h
  - **Functionality**:
    ```
    // src/commands/init.ts
    export default class Init extends BaseCommand {
    async run(): Promise<void> {
    // 1. Ask for workspace name
    // 2. Create directory structure
    // 3. Initialize SQLite database
    // 4. Create default config file
    // 5. Validate Gemini API key
    // 6. Test API connection
    this.log('✓ Workspace initialized at ./auto-gemini');
    }
    }
    ```
- [ ] **Tâche 6.5**: Implement `task run` command
  - **Owner**: Senior Dev
  - **Deliverable**: `src/commands/task/run.ts`
  - **Effort**: 6h
  - **Functionality**:
    ```
    // src/commands/task/run.ts
    // Key features:
    // - Load task from database
    // - Execute with Gemini
    // - Show live progress (spinner)
    // - Stream output to terminal
    // - Save result to session
    ```

Semaine 6 Closure

- [ ] **Tâche 6.6**: Test basic CLI workflow
  - **Owner**: Senior Dev
  - **Deliverable**: CLI test results
  - **Effort**: 2h
  - **Test**:
    $ npm run dev -- init --name test-workspace
    $ npm run dev -- task new "hello world"
    $ npm run dev -- task run <id>

    # Expected: Should complete without errors

Semaine 7 (17-23 février 2026)

Jour 25-26: TUI Components

- [ ] **Tâche 7.1**: Create TUI components (Ink.js or Blessed)
  - **Owner**: Senior Dev
  - **Deliverable**: TUI component library
  - **Effort**: 8h
  - **Components**:
    ```
    // src/tui/components/
    ├── Spinner.tsx # Loading spinner
    ├── ProgressBar.tsx # Progress indication
    ├── StatusBar.tsx # Bottom status bar
    ├── TaskOutput.tsx # Streaming output display
    ├── ErrorDisplay.tsx # Error messages with colors
    └── InteractivePrompt.tsx # User input
    ```
- [ ] **Tâche 7.2**: Implement `chat` interactive mode
  - **Owner**: Senior Dev
  - **Deliverable**: `src/commands/chat.ts`
  - **Effort**: 6h
  - **Features**:
    - [x] REPL interface (readline-based)
    - [x] Session persistence
    - [x] Streaming responses
    - [x] Command history
    - [x] Exit gracefully

Jour 27-28: Session Management UI

- [ ] **Tâche 7.3**: Implement `session list` and `session show`
  - **Owner**: Senior Dev
  - **Deliverable**: `src/commands/session/*.ts`
  - **Effort**: 5h
  - **Output format**:
    ```
    $ auto-gemini session list
    ID | Created | Tasks | Last Modified
    -----------+--+
    ```

```
sess-001 | 2026-01-20 | 5 | 1 hour ago
sess-002 | 2026-01-15 | 12 | 2 days ago
$ auto-gemini session show sess-001
Session: sess-001
Created: 2026-01-20 14:30:00
Tasks executed: 5
Total tokens: 45,000
History:
[14:30] Task: "write hello world" → ✓
[14:32] Task: "optimize code" → ✓
...
```

- [ ] **Tâche 7.4**: Implement `config` commands
    - **Owner**: Senior Dev
    - **Deliverable**: `src/commands/config.ts`
    - **Effort**: 3h

Semaine 8 (24-2 mars 2026)

Jour 29-30: Feature Completeness

- [ ] **Tâche 8.1**: Implement remaining commands
    - **Owner**: Senior Dev
    - **Deliverable**: All commands from Phase 2 spec
    - **Effort**: 8h
    - **Commands to complete**:
        - task list, task delete
        - logs tail, logs grep
        - status --watch
        - All help pages
- [ ] **Tâche 8.2**: Test complete CLI flow
    - **Owner**: Senior Dev
    - **Deliverable**: Comprehensive CLI tests
    - **Effort**: 6h
    - **Test scenarios**:

Scenario 1: New user workflow

```
$ auto-gemini init
$ auto-gemini task new "write hello world"
$ auto-gemini task run <id>
$ auto-gemini session show <id>
```

Scenario 2: Chat mode

```
$ auto-gemini chat
```

> Hello
> What is my name?
> exit

Scenario 3: Configuration

```
$ auto-gemini config set model gemini-2.0-pro
$ auto-gemini config get model
$ auto-gemini config show
```

Jour 31: Polish & Phase 2 Gate

- [ ] **Tâche 8.3**: Code cleanup & documentation
  - **Owner**: Senior Dev + Lead Arch
  - **Deliverable**: Clean codebase
  - **Effort**: 4h
  - **Checklist**:
    - [x] Remove console.log() (use logger)
    - [x] Add JSDoc comments
    - [x] CLI help text complete
    - [x] No broken links in docs
- [ ] **Tâche 8.4**: Phase 2 Gate Decision
  - **Owner**: Lead Arch
  - **Deliverable**: Go/No-Go for Phase 3
  - **Effort**: 2h
  - **Gate criteria**:

Phase 2 Gate Criteria

✓ Acceptance (GO to Phase 3) if:
- [ ] All core commands working (init, task, chat, config)
- [ ] CLI responsive and user-friendly
- [ ] TUI components smooth (no lag)
- [ ] Session persistence working
- [ ] Help documentation complete
- [ ] No critical bugs

✗ Re-planning (NO-GO) if:
- [ ] TUI performance poor (>500ms latency)
- [ ] Commands broken or inconsistent
- [ ] Session data corrupt

Decision: ☐ GO ☐ NO-GO

PHASE 3: OPTIMISATION GEMINI (Semaines 9-12)

Objectif: Optimiser pour Gemini specifics (rate limiting, context, caching)
Livrable: Production-ready system, performance baseline, documentation
Succès: Passe load tests, handles constraints gracefully, monitoring active

Semaine 9 (3-9 mars 2026)

Jour 32-33: Rate Limiting System

- [ ] **Tâche 9.1**: Implement token-bucket rate limiter
    - **Owner**: Senior Dev
    - **Deliverable**: `src/services/rate-limiter.ts`
    - **Effort**: 6h
    - **Strategy**:
      ```
      // Rate limit: 1000 requests / 24 hours
      // Strategy: Token-bucket queue with SQLite persistence
      interface RateLimitConfig {
      maxRequests: number; // 1000
      windowSeconds: number; // 86400 (24 hours)
      burstSize: number; // Allow 10 consecutive
      }
      class RateLimiter {
      async canMakeRequest(): Promise<boolean> {
      // Check if we have tokens available
      // If yes: consume token, return true
      // If no: queue request, return false
      }
      async queueRequest(task: Task): Promise<QueueID> {
      // Persist to DB with scheduled execution time
      // Emit event when task becomes executable
      }
      }
      ```
- [ ] **Tâche 9.2**: Implement request queue & scheduler
    - **Owner**: Senior Dev
    - **Deliverable**: Queue management system
    - **Effort**: 6h
    - **Features**:
        - [x] Persistent queue (SQLite)
        - [x] Scheduled execution (cron-like)
        - [x] Priority levels (high, normal, low)
        - [x] Retry logic (exponential backoff)

Jour 34-35: Context Window Management

- [ ] **Tâche 9.3**: Implement context manager for 1M tokens
    - **Owner**: Senior Dev
    - **Deliverable**: `src/services/context-manager.ts`
    - **Effort**: 8h
    - **Strategy**:
      ```
      class ContextManager {
      maxTokens: number = 1_000_000;
      currentTokens: number = 0;
      async validatePrompt(prompt: string): Promise<ValidationResult> {
      const tokens = await this.countTokens(prompt);
      if (currentTokens + tokens > maxTokens) {
      return {
      valid: false,
      ```

```
availableTokens: maxTokens - currentTokens,
suggestedChunking: calculateChunks(prompt),
};
}
return { valid: true };
}
async addToContext(content: string): Promise<void> {
const tokens = await this.countTokens(content);
this.currentTokens += tokens;
}
}
```

- [ ] **Tâche 9.4**: Implement context window cleanup
  - **Owner**: Senior Dev
  - **Deliverable**: Sliding window & compression
  - **Effort**: 4h
  - **Strategies**:
    - Sliding window (remove oldest messages)
    - Compression (summarize old context)
    - Chunking (split large tasks)

Semaine 10 (10-16 mars 2026)

Jour 36-37: Performance Optimization

- [ ] **Tâche 10.1**: Optimize Gemini API calls
  - **Owner**: Senior Dev
  - **Deliverable**: Performance baseline & optimizations
  - **Effort**: 8h
  - **Optimizations**:

    # Performance Optimizations

    1. **Streaming Optimization**
       - Enable streaming for large outputs
       - Emit chunks as they arrive (don't buffer)
       - Reduces apparent latency
    2. **Caching Strategy**
       - Cache countTokens() results (same prompt)
       - Cache API responses (if safe)
       - Use Gemini Cache API (if available)
    3. **Batch Requests** (future)
       - If multiple tasks queued, batch API calls
       - Reduce API overhead
    4. **Model Selection**
       - Small tasks: gemini-2.0-flash (faster, cheaper)
       - Complex tasks: gemini-2.0-pro (more powerful)
       - Auto-detect based on complexity
- [ ] **Tâche 10.2**: Implement monitoring & metrics
  - **Owner**: Senior Dev
  - **Deliverable**: Metrics collection system
  - **Effort**: 5h

- **Metrics to track**:

  ## Key Metrics

  - API Response Time (p50, p95, p99)
  - Tokens Used / Task
  - Queue Depth
  - Error Rate by Type
  - Session Duration
  - Success Rate

  Tools:
  - Prometheus (optional, for advanced monitoring)
  - Simple JSON logging for now

Jour 38-39: Testing & Validation

- [ ] **Tâche 10.3**: Create load tests
  - **Owner**: Senior Dev
  - **Deliverable**: Load testing suite
  - **Effort**: 6h
  - **Scenarios**:
    ```
    // Load test: Execute 100 tasks sequentially
    // Measure: API latency, queue behavior, memory growth
    async function loadTest() {
    for (let i = 0; i < 100; i++) {
    const task = task-${i};
    const start = Date.now();
    await agent.executeTask(task);
    const duration = Date.now() - start;
    console.log(Task ${i}: ${duration}ms);
    }
    }
    // Expected results:
    // - p50: < 500ms
    // - p95: < 2000ms
    // - Memory: stable (no leaks)
    // - Queue: processes at rate limit
    ```
- [ ] **Tâche 10.4**: Create stress tests
  - **Owner**: Senior Dev
  - **Deliverable**: Stress testing suite
  - **Effort**: 4h
  - **Scenarios**:
    - 1000+ token context
    - Very long prompts
    - Rapid-fire requests
    - Network failures/timeouts

Semaine 11 (17-23 mars 2026)

Jour 40-41: Error Handling & Resilience

- [ ] **Tâche 11.1**: Comprehensive error handling
  - **Owner**: Senior Dev
  - **Deliverable**: Error recovery strategies for all Gemini errors
  - **Effort**: 6h
  - **Error types handled**:
    - [x] Rate limiting (RESOURCE_EXHAUSTED)
    - [x] Auth failures (PERMISSION_DENIED)
    - [x] Content safety (INVALID_ARGUMENT)
    - [x] Context window overflow
    - [x] Network timeouts
    - [x] Malformed responses
- [ ] **Tâche 11.2**: Implement circuit breaker
  - **Owner**: Senior Dev
  - **Deliverable**: Circuit breaker pattern implementation
  - **Effort**: 4h
  - **States**:
    - CLOSED: Normal operation
    - OPEN: API failing, reject requests
    - HALF_OPEN: Testing if API recovered

Jour 42-43: Documentation & Examples

- [ ] **Tâche 11.3**: Write comprehensive documentation
  - **Owner**: Senior Dev + Lead Arch
  - **Deliverable**: /docs/USER_GUIDE.md, /docs/API.md
  - **Effort**: 8h
  - **Content**:

    # User Guide

    ## Installation

    npm install -g auto-gemini-cli

    # or

    npx auto-gemini-cli init

    ## Quick Start

    ```
    $ auto-gemini init
    $ auto-gemini task new "write hello world"
    $ auto-gemini task run <id>
    ```

Configuration

API Key

```
export GEMINI_API_KEY="..."
auto-gemini config set apiKey $GEMINI_API_KEY
```

Rate Limiting

- Free tier: 1000 requests/24h
- Monitor queue: `auto-gemini status --watch`
- Scheduled execution: Automatic

Troubleshooting

Rate Limit Exceeded

- Tasks queued automatically
- Check status: `auto-gemini status`
- Manually clear: `auto-gemini session clear <id>`

Context Window Full

- Auto-chunking enabled
- Or: Reduce task scope
- Or: Start new session

- [ ] **Tâche 11.4**: Create example workflows
  - **Owner**: Senior Dev
  - **Deliverable**: Example scripts & documentation
  - **Effort**: 4h
  - **Examples**:
    - Code generation workflow
    - Documentation writing
    - Bug analysis workflow

Semaine 12 (24-30 mars 2026)

Jour 44-45: Integration Testing

- [ ] **Tâche 12.1**: Run comprehensive integration tests
  - **Owner**: Senior Dev
  - **Deliverable**: Integration test suite results
  - **Effort**: 8h
  - **Test coverage**:

Integration Tests

- [ ] Rate limiter queues requests correctly
- [ ] Context manager prevents overflow
- [ ] Error recovery works for each error type
- [ ] Streaming continues after network hiccup
- [ ] Sessions persist across restarts
- [ ] Parallel tasks don't corrupt state
- [ ] Memory stable over 24h simulation

- [ ] **Tâche 12.2**: Performance benchmarking
    - **Owner**: Senior Dev
    - **Deliverable**: Performance report
    - **Effort**: 4h
    - **Baseline metrics**:
      Task Execution Time:
        - Simple (< 100 tokens): 200-500ms
        - Medium (100-1000 tokens): 500-2000ms
        - Large (1000+ tokens): 2000-5000ms

      Memory Usage:
        - Idle: < 50MB
        - Single session: < 100MB
        - Multi-session (10): < 200MB

      API Overhead:
        - countTokens() call: ~50ms
        - generateContent() call: ~200-1000ms (varies)

Jour 46-47: Phase 3 Gate

- [ ] **Tâche 12.3**: Production readiness review
    - **Owner**: Lead Arch + Senior Dev
    - **Deliverable**: Production readiness checklist
    - **Effort**: 6h
    - **Checklist**:

Production Readiness Checklist

Code Quality

- [ ] No console.log() statements
- [ ] All errors have appropriate recovery
- [ ] Logging is comprehensive
- [ ] Type safety: no `any` types
- [ ] Performance baseline established
- [ ] Memory leaks tested & verified none

Operations

- [ ] Health check endpoint works
- [ ] Logging to file for analysis
- [ ] Error tracking integrated (Sentry)
- [ ] Metrics exported (Prometheus optional)
- [ ] Documentation complete & accurate

Security

- [ ] API key never logged
- [ ] Secrets not in repository
- [ ] SQLite database encrypted (optional)
- [ ] Rate limiting prevents abuse
- [ ] Input validation comprehensive

Testing

- [ ] Unit tests: > 80% coverage
- [ ] Integration tests: all critical paths
- [ ] Load tests: < 2s p95 latency
- [ ] Stress tests: handles edge cases
- [ ] E2E tests: against real Gemini API

Deployment

- [ ] Build process automated
- [ ] Docker image created (optional)
- [ ] Executable binary tested
- [ ] Installation documented
- [ ] Upgrade path documented

- [ ] **Tâche 12.4**: Phase 3 Gate Decision
  - **Owner**: Lead Arch
  - **Deliverable**: Go/No-Go for Phase 4
  - **Effort**: 2h
  - **Gate criteria**:

Phase 3 Gate Criteria

✓ Acceptance (GO to Phase 4) if:
- [ ] Rate limiting working correctly
- [ ] Context manager prevents overflows
- [ ] Performance meets baseline
- [ ] Error handling comprehensive
- [ ] Load tests pass (p95 < 2s)
- [ ] No known bugs
- [ ] Production readiness > 90%

✗ Re-planning (NO-GO) if:
- [ ] Performance far below baseline
- [ ] Rate limiting buggy
- [ ] Memory leaks detected

Decision: ☐ GO ☐ NO-GO

PHASE 4: INTÉGRATION & RELEASE (Semaines 13-14)

Objectif: Tester e2e, finaliser documentation, déployer v1.0
Livrable: v1.0 release, documentation complète, support ready
Succès: v1.0 dans npm registry, utilisable en production

Semaine 13 (31 mars - 6 avril 2026)

Jour 48-49: Final Testing

- [ ] **Tâche 13.1**: Execute comprehensive E2E tests
    - **Owner**: Senior Dev
    - **Deliverable**: E2E test results (real Gemini API)
    - **Effort**: 8h
    - **Test scenarios**:

Scenario 1: Fresh install

npm install -g auto-gemini-cli
auto-gemini init
auto-gemini config set apiKey $GEMINI_API_KEY
auto-gemini task new "write a fibonacci function"
auto-gemini task run <id>

Expected: Task completes successfully

Result stored in session

Scenario 2: Long-running task

auto-gemini task new "analyze this large repository" # 500K+ tokens
auto-gemini task run <id> --watch

Expected: Auto-chunks, shows progress, completes

Scenario 3: Rate limiting

for i in {1..20}; do
auto-gemini task new "task $i''auto - geminitaskruntask-$i --no-wait
done
auto-gemini status --watch

Expected: First 10 run, rest queued

Queue processes as rate limit allows

Scenario 4: Error recovery

(Manually trigger network failure)

Expected: Circuit breaker engages, retries work

- [ ] **Tâche 13.2**: User acceptance testing (if stakeholders available)
  - **Owner**: Desjardins stakeholders + Lead Arch
  - **Deliverable**: Feedback & sign-off
  - **Effort**: 4h
  - **Feedback areas**:
    - CLI UX (is it intuitive?)
    - Performance (is it fast enough?)
    - Documentation (is it clear?)
    - Reliability (does it work consistently?)

Jour 50-51: Documentation & Examples

- [ ] **Tâche 13.3**: Finalize documentation
  - **Owner**: Senior Dev + Lead Arch
  - **Deliverable**: Complete documentation suite
  - **Effort**: 8h
  - **Documents to create/update**:
    - README.md (overview, quick start)
    - INSTALLATION.md (detailed install)
    - USER_GUIDE.md (step-by-step)
    - API.md (command reference)
    - TROUBLESHOOTING.md (common issues)
    - ARCHITECTURE.md (technical details)
    - MIGRATION.md (from Auto-Claude)
    - CONTRIBUTING.md (for future contributors)
- [ ] **Tâche 13.4**: Create changelog & release notes
  - **Owner**: Lead Arch
  - **Deliverable**: CHANGELOG.md, release notes
  - **Effort**: 2h
  - **Content**:

CHANGELOG

v1.0.0 - 2026-05-01

Major Features

- ✨ Gemini CLI: Full adaptation of Auto-Claude for Gemini API
- ✨ Rate limiting: Automatic queue management (1000 req/24h)
- ✨ Context management: Handle up to 1M token contexts
- ✨ Interactive chat: REPL mode with session persistence
- ✨ Performance: Streaming responses, fast token counting

Breaking Changes

- Electron UI removed (CLI only)
- Config file format slightly different
- API key env var: GEMINI_API_KEY (was CLAUDE_API_KEY)

Migration

- See MIGRATION.md for moving from Auto-Claude

Known Limitations

- Free tier: 1000 requests/24h (Google limit)
- No multi-model support yet
- No caching API integration yet

Contributors

- André-Guy Bruneau (Lead Architect)
- [Team members]

Semaine 14 (7-13 avril 2026)

Jour 52-53: Build & Publishing

- [ ] **Tâche 14.1**: Create build artifacts
  - **Owner**: Lead Arch
  - **Deliverable**: Packaged binaries & npm package
  - **Effort**: 4h
  - **Artifacts**:

    ## Build executable

    npm run build

    ## Package for different platforms (optional)

    npm run package:macos
    npm run package:linux
    npm run package:windows

Output:

dist/

├── auto-gemini-cli-1.0.0.tar.gz

├── auto-gemini-cli-1.0.0.exe

└── auto-gemini-cli-1.0.0.dmg

- [ ] **Tâche 14.2**: Publish to npm registry
    - **Owner**: Lead Arch
    - **Deliverable**: npm package published
    - **Effort**: 2h
    - **Steps**:

      ## Verify package.json

      npm version major # This is v1.0.0

      ## Test publish (optional)

      npm publish --dry-run

      ## Publish to npm

      npm publish

      ## Verify

      npm search auto-gemini-cli
      npm view auto-gemini-cli@1.0.0
- [ ] **Tâche 14.3**: Create GitHub release
    - **Owner**: Lead Arch
    - **Deliverable**: GitHub release page
    - **Effort**: 1h
    - **Content**:
        - Release notes (copy from CHANGELOG)
        - Binaries attached
        - Installation instructions
        - Known issues

Jour 54-55: Release & Support

- [ ] **Tâche 14.4**: Announce release
    - **Owner**: Lead Arch
    - **Deliverable**: Release announcement
    - **Effort**: 2h
    - **Channels**:
        - GitHub releases
        - Email to stakeholders
        - Internal documentation
        - (Optional: blog post, Twitter, etc.)
- [ ] **Tâche 14.5**: Setup support & issue tracking
    - **Owner**: Lead Arch
    - **Deliverable**: Support guidelines, issue templates
    - **Effort**: 2h
    - **Checklist**:

Support Setup

- [ ] GitHub issues enabled
- [ ] Issue templates created (.github/ISSUE_TEMPLATE/)
- [ ] Discussions enabled
- [ ] Slack channel #auto-gemini-cli-support
- [ ] Email support alias set up
- [ ] Response time SLAs defined

Issue Templates:

- Bug Report
- Feature Request
- Question
- Documentation Issue

Jour 56: Final Validation & Gate

- [ ] **Tâche 14.6**: v1.0 Sign-off
    - **Owner**: Lead Arch + Desjardins Leadership
    - **Deliverable**: Official sign-off
    - **Effort**: 2h
    - **Final checklist**:

v1.0 Release Sign-Off

Technical

- [ ] All tests passing (unit, integration, E2E)
- [ ] Performance baseline met
- [ ] No critical bugs
- [ ] Documentation complete
- [ ] Code reviewed & approved

Operational

- [ ] npm package published
- [ ] GitHub release created
- [ ] Support process ready
- [ ] Monitoring active

Stakeholder

- [ ] Desjardins approval received
- [ ] Budget tracking complete
- [ ] Timeline met (or documented variance)
- [ ] Deliverables accepted

Sign-off: ☐ APPROVED FOR PRODUCTION
Date: _______________
By: André-Guy Bruneau (Lead Architect)
For: Desjardins

- [ ] **Tâche 14.7**: Post-Release Planning
  - **Owner**: Lead Arch
  - **Deliverable**: Roadmap for future versions
  - **Effort**: 2h
  - **Future work**:

Post-v1.0 Roadmap

v1.1 (2-3 months after v1.0)

- [ ] Gemini Cache API integration
- [ ] Multi-model support (Claude + Gemini)
- [ ] Web UI (optional)
- [ ] Plugin system

v2.0 (6+ months after v1.0)

- [ ] Distributed execution (multi-machine)
- [ ] Advanced scheduling
- [ ] Workflow orchestration
- [ ] Team collaboration features

Community

- [ ] Contribute guidelines
- [ ] Plugin template
- [ ] Community examples repo

DÉPENDANCES & BLOCKERS

Dépendances Critiques

Phase 0:
└─→ Approval budgétaire Desjardins (CRITICAL)
└─→ Phase 1 begins

Phase 1:
└─→ Gemini API stable & available (CRITICAL)
└─→ Codebase analysis complete
└─→ Phase 2 begins

Phase 2:
└─→ Phase 1 complete (at least MVP working)
└─→ Oclif framework understood
└─→ Phase 3 begins

Phase 3:
└─→ Phase 2 complete (CLI working)
└─→ Load testing tools available
└─→ Phase 4 begins

Phase 4:
└─→ Phase 3 complete (optimization done)
└─→ npm account available
└─→ Release to npm

Potential Blockers & Mitigation

Blocker	Impact	Probability	Mitigation
Gemini API breaking change	2-3 weeks delay	MEDIUM	Monitor API status, use adapters
Codebase more coupled than expected	1-2 weeks delay	MEDIUM	Design refactoring upfront
Team member unavailable	Timeline slip	LOW	Cross-train both team members
Performance unacceptable	1 week delay	LOW	Optimize streaming, caching
Rate limiting impossible to work around	Pivot needed	LOW	Fall back to Scenario 3 (Claude)
Gemini free tier discontinued	Costs increase	LOW	Switch to paid tier or Scenario 3

Risk Mitigation Actions

- **Weekly risk review**: Every Monday, assess blockers
- **Slack #auto-gemini-risks**: Real-time communication
- **Decision framework**: If blocker > 2 days impact, escalate immediately
- **Buffer time**: 2-3 weeks reserved for unexpected delays

MÉTRIQUES DE SUCCÈS

Phase-by-Phase Success Metrics

Phase	Key Metrics	Target	Measurement
0	Approval %	100%	Budget signed
	Codebase understood	Yes	ADRs written
	Team ready	Yes	Env setup complete
1	API swap working	Yes	E2E test passes
	Tests passing	70%+	Test report
	Token counter validated	Yes	countTokens tests
2	Commands functional	100%	CLI tests pass
	TUI responsive	<500ms	Latency test
	User can complete workflow	Yes	User test successful
3	Rate limiting works	Yes	Queue test passes
	Context management works	Yes	1M token test
	Performance baseline met	Yes	Benchmarks pass
4	v1.0 released to npm	Yes	npm publish successful
	Documentation complete	Yes	All docs reviewed
	Zero critical bugs	Yes	Bug report review

Overall Project Success Metrics

Timeline: 14 weeks (3.5 months) ✓ vs original 19 weeks
Cost: $13-34K ✓ vs original $100-150K
Features: 100% of Phase 1-4 scope ✓
Quality: >80% test coverage ✓
Performance: p95 < 2s ✓
Uptime (day 1 of release): >95% ✓
User satisfaction: >4/5 ✓
Zero production hotfixes needed ✓

GESTION DES RISQUES

Risk Register

Risk #1: Gemini API Incompatibility

- **Description**: Critical incompatibility between Claude and Gemini APIs discovered during Phase 1
- **Probability**: MEDIUM (50%)
- **Impact**: HIGH (2-3 weeks delay)
- **Mitigation**: Adapter pattern designed to isolate API differences
- **Owner**: Senior Dev
- **Action**: Daily API testing, early detection

Risk #2: Performance Unacceptable

- **Description**: Gemini API slower than Claude, impacting user experience
- **Probability**: LOW (20%)
- **Impact**: MEDIUM (1 week delay)
- **Mitigation**: Implement streaming, caching, optimize before Phase 4
- **Owner**: Senior Dev
- **Action**: Performance profiling in Phase 3

Risk #3: Rate Limiting Not Workable

- **Description**: 1000 req/24h limit too restrictive for use cases
- **Probability**: LOW (15%)
- **Impact**: HIGH (complete pivot)
- **Mitigation**: Pivot to Scenario 3 (Claude Code) as backup
- **Owner**: Lead Arch
- **Action**: Have Scenario 3 plan ready

Risk #4: Team Member Unavailable

- **Description**: One team member becomes unavailable mid-project
- **Probability**: LOW (10%)
- **Impact**: MEDIUM (1-2 weeks delay)
- **Mitigation**: Cross-train both team members early
- **Owner**: Lead Arch
- **Action**: Shared knowledge sessions Week 1-2

Risk #5: Scope Creep

- **Description**: Additional features requested mid-project
- **Probability**: MEDIUM (60%)
- **Impact**: MEDIUM (timeline slip)
- **Mitigation**: Strict change control, defer to v1.1
- **Owner**: Lead Arch
- **Action**: Weekly stakeholder alignment

Risk Monitoring

- **Weekly risk review**: Every Monday in standup
- **Escalation path**: If risk impacts > 2 days, escalate to Lead Arch
- **Risk board**: GitHub project "Risks" tab
- **Contingency time**: 2-3 weeks buffer built into timeline

BUDGET DÉTAILLÉ

Cost Breakdown

```
PHASE 0: Préparation (2 weeks)
├── Lead Architect (1 FTE): 1.0 × 2 × $200/hr = $1,600
├── Senior Dev (0.5 FTE): 0.5 × 2 × $200/hr = $800
└── Subtotal: = $2,400

PHASE 1: Swap API (3 weeks)
├── Lead Architect (1 FTE): 1.0 × 3 × $200/hr = $2,400
├── Senior Dev (0.5 FTE): 0.5 × 3 × $200/hr = $1,200
├── Testing tools (optional): = $500
└── Subtotal: = $4,100

PHASE 2: UI Migration (3 weeks)
├── Lead Architect (0.5 FTE): 0.5 × 3 × $200/hr = $1,200
├── Senior Dev (1 FTE): 1.0 × 3 × $200/hr = $2,400
├── Design review: = $300
└── Subtotal: = $3,900

PHASE 3: Optimization (4 weeks)
├── Lead Architect (0.5 FTE): 0.5 × 4 × $200/hr = $1,600
├── Senior Dev (1 FTE): 1.0 × 4 × $200/hr = $3,200
├── Load testing tools: = $500
├── Monitoring (Sentry free): = $0
└── Subtotal: = $5,300

PHASE 4: Release (2 weeks)
├── Lead Architect (1 FTE): 1.0 × 2 × $200/hr = $1,600
├── Senior Dev (0.5 FTE): 0.5 × 2 × $200/hr = $800
├── Documentation review: = $300
└── Subtotal: = $2,700

INFRASTRUCTURE & TOOLS:
├── GitHub (free): = $0
├── npm (free): = $0
├── Google Gemini (free tier): = $0
├── Sentry (free): = $0
├── Slack (existing): = $0
└── Subtotal: = $0

─────────────────────────────────────────────────────────

TOTAL PROJECT COST: $18,400
```

Contingency (10% for unknowns): $ 1,840

TOTAL BUDGET (with contingency): $20,240

APPROVED BUDGET RANGE: $13-34K
STATUS: ✓ WITHIN RANGE

Budget Allocation by Cost Center

Labor Costs (FTE Hours):
├── Lead Architect: 1.0 × 14 weeks × 40h/week × $200/hr = $11,200
├── Senior Dev: 0.5 × 14 weeks × 40h/week × $200/hr = $5,600
└── Total Labor: = $16,800

Tools & Services: = $1,300
├── Testing tools: $500
├── Monitoring: $300 (Sentry optional)
├── Design review: $300
├── Documentation: $200

Contingency (10%): = $1,840

TOTAL: = $20,240

Cost Savings Opportunities

- **Reuse Auto-Claude code**: Saves ~30% development (vs Scenario 1)
- **Leverage Gemini free tier**: No API costs (1000 req/24h)
- **Use free tools**: GitHub, npm, Sentry free tier
- **Cross-training**: Avoids hiring additional staff

Cost Tracking

Weekly cost tracking:

- Team hours logged (time tracking)
- Tool subscriptions tracked
- Contingency reserve managed
- Budget variance reported

Monthly stakeholder report:

- Actual vs planned spending
- Forecast to completion
- Risk adjustments

CHECKLIST FINALE

Pre-Launch (Semaine 13)

- [ ] All phases completed
- [ ] All tests passing (>90%)
- [ ] Documentation reviewed
- [ ] Performance baseline met
- [ ] Security audit passed
- [ ] Stakeholder sign-off obtained

Launch (Semaine 14)

- [ ] npm package published
- [ ] GitHub release created
- [ ] Announcement sent
- [ ] Support team ready
- [ ] Monitoring active
- [ ] Rollback plan documented

Post-Launch (First 2 weeks)

- [ ] Monitor error rates (target: <1%)
- [ ] Track user feedback
- [ ] Respond to support requests (<4h)
- [ ] Document lessons learned
- [ ] Plan v1.1 features

CONCLUSION

Planification complète pour Scénario 2: Adaptation Auto-Claude vers Auto-Gemini-CLI

- **Durée totale**: 14 semaines (3.5 mois), livraison ~1er mai 2026
- **Effort**: 1.5 FTE (Lead Architect + Senior Developer)
- **Budget**: $13-34K (plus ou moins contingency)
- **Approche**: Itérative par phases, avec gates de décision claire
- **Succès défini**: v1.0 production-ready, documenté, testé, releasé

Prochaines étapes:

1. Obtenir approbation budgétaire formelle de Desjardins
2. Assembler équipe (Lead Arch + Senior Dev)
3. Commencer Phase 0 semaine de 6 janvier 2026
4. Hold bi-weekly stakeholder reviews

Contact & Questions: André-Guy Bruneau (agbruneau)