Правила работы с Cursor и AI-инструментами в Ruby/Rails проектах с примерами промптов

ChatGPT Version

Разработка на Ruby с AI: меньше рутины, больше смысла

by Thinknetika and Eugene Leontev

Thought for a couple of seconds

1. Style & Conventions

- **Единый стиль именования**
  Позволяет AI-ассистентам точнее предсказывать и генерировать код, снижая количество конфликтов стиля.
  - Пример хорошего промпта:
    > “Напиши метод `calculate_total_price` в соответствии с Ruby Style Guide — snake_case для названий методов и переменных.”
  - Пример плохого промпта:
    > “Создай метод для подсчёта цены.”

- **Стандартизированные сниппеты**
  Храните готовые шаблоны запросов в Cursor — ускоряет написание повторяющихся конструкций и повышает качество кода.
  - Пример хорошего промпта:
    > “Используй сниппет `AR_FIND_OR_CREATE` для генерации `Model.find_or_create_by(attributes)` с ручным вводом атрибутов.”
  - Пример плохого промпта:
    > “Сделай find_or_create_by без контекста.”

- **Type signatures (RBS/TypeProf аннотации)**
  Добавляйте аннотации типов — AI-модели лучше понимают структуру методов и их контракты.
  - Пример хорошего промпта:
    > “Добавь RBS аннотацию для метода `def total(price: Float, tax: Float) → Float`.”
  - Пример плохого промпта:
    > “Напиши типы для всего приложения без указания конкретного метода.”

2. Prompt Design

- **Ясные и короткие промпты**
  Уточняйте цель и контекст в нескольких предложениях — повышает точность ответов.
  - Пример хорошего промпта:
    > “Нужна валидация email в модели User: строка должна содержать ‘@’ и не быть пустой.”
  - Пример плохого промпта:
    > “Добавь валидацию.”

- **Примеры ввода-вывода**
  Показывайте sample input/output вместе с запросом — помогает Cursor генерировать более релевантный код.
  - Пример хорошего промпта:
    > “Метод `format_date` принимает `‘2025-07-13’`, возвращает `‘13 July 2025’`.”
  - Пример плохого промпта:
    > “Сделай форматирование даты.”

- **Ограничение области поиска**
  При автодополнении задавайте scope (модуль/директория) — снижает шум и исключает нерелевантные предложения.
  - Пример хорошего промпта:
    > “В директории `app/services/payments/` сгенерируй `PaymentProcessor#charge`.”
  - Пример плохого промпта:
    > “Сгенерируй PaymentProcessor#charge где-нибудь.”

- **Локализация запроса**
  Указывайте Ruby/Rails-версию и стек гемов — исключает несовместимость автодополнений.
  - Пример хорошего промпта:
    > “Для Rails 7.1 и Ruby 3.2 с ActiveRecord 7.1 напиши валидацию уникальности email.”
  - Пример плохого промпта:
    > “Для старого Rails напиши валидацию.”

- **Разбиение больших запросов**
  Делите комплексные задачи на шаги — AI-модель даёт более точные и управляемые ответы.
  - Пример хорошего промпта:
    > “Шаг 1: сгенерируй миграцию для таблицы orders. Шаг 2: опиши модель Order с ассоциацией к User.”
  - Пример плохого промпта:
    > “Сделай всю логику заказов сразу одной командой.”

- **Итеративная генерация**
  Сначала получайте skeleton, затем просите детали — контролируете качество и полноту кода.
  - Пример хорошего промпта:
    > “Дай skeleton контроллера ProductsController со стандартными CRUD.”
  - Пример плохого промпта:
    > “Сразу напиши весь контроллер целиком.”

- **Сохраняйте минимальный контекст**
  Держите промпты компактными, но информативными — уменьшаете задержку и ускоряете отклик.
  - Пример хорошего промпта:
    > “Метод `apply_discount(order, code)` — кратко: применить скидку по коду.”
  - Пример плохого промпта:
    > “Полный dump модели Order, User, Coupon… и ещё 10 строк пояснений.”

- **Контроль версий промптов**
  Храните промпты в репозитории с changelog — гарантирует воспроизводимость и прозрачность.
  - Пример хорошего промпта:
    > “Запиши в `prompts.md`: ‘Добавление дружелюбной ошибки при невалидном promo_code’.”
  - Пример плохого промпта:
    > “Не сохраняй промпты, просто копируй их в чат.”

3. Testing & Quality Assurance

- **Интеграция с тестами**
  Просите Cursor генерировать и обновлять тесты рядом с кодом — гарантирует покрытие и облегчает рефакторинг.
  - Пример хорошего промпта:
    > “Сгенерируй RSpec-тест для `PaymentProcessor#charge(amount)` с моками HTTP.”
  - Пример плохого промпта:
    > “Просто проверь, что charge работает.”

- **Юнит-мокирование через AI**
  Генерируйте заглушки и моки для внешних сервисов — ускоряет написание изолированных тестов.
  - Пример хорошего промпта:
    > “Создай мок `Stripe::Charge.create` возвращающий успешный ответ.”
  - Пример плохого промпта:
    > “Напиши тесты для Stripe.”

- **AI-асистированный рефакторинг**
  Используйте шаги «Extract method», «Inline variable» — безопасно улучшает код без регресса.
  - Пример хорошего промпта:
    > “Extract method: вынеси валидацию email в `validate_email_format`.”
  - Пример плохого промпта:
    > “Рефакторни всё приложение.”

- **Код-ревью с AI**
  Просите Cursor отмечать «code smells» при открытии PR — упрощает работу ревьюеров и ускоряет процесс.
    - Пример хорошего промпта:
      > “Проанализируй PR #42 и укажи места с высоким Cyclomatic Complexity.”
    - Пример плохого промпта:
      > “Скажи, плохой ли этот код.”

- **Проверка безопасности**
  Регулярно запускайте AI-аудит на поиск SQL-инъекций и XSS — повышает надёжность и защиту.
  - Пример хорошего промпта:
    > “Проверь метод `search_users(query)` на риск SQL-инъекций.”
  - Пример плохого промпта:
    > “Безопасность проверь.”

- **Порог доверия**
  Всегда проверяйте AI-сгенерированный код тестами и ручным ревью, даже если он выглядит корректно — предотвращает ошибки.
  - Пример хорошего промпта:
    > “Добавь тесты для каждой ветки метода `calculate_discount`.”
  - Пример плохого промпта:
    > “Доверься AI и сразу коммить.”

4. CI & Automation

- **Интеграция с CI**
  Включайте шаг запуска Cursor-скриптов в пайплайн для автоматического обновления документации и тестов.
  - Пример хорошего промпта:
    > “Добавь в .gitlab-ci.yml задачу `cursor:generate-docs`, которая запускает `cursor export-docs`.”
  - Пример плохого промпта:
    > “Запусти Cursor вручную перед мёржем.”

- **Автоматическая миграция схемы**
  Генерируйте миграции на основе изменений в моделях — экономит время и снижает риск рассинхронизации.
  - Пример хорошего промпта:
    > “По изменениям в модели Order добавь колонку `status:string` через миграцию.”
  - Пример плохого промпта:
    > “Сделай миграцию сам по интуиции AI.”

- **Удобные alias’ы в CLI**
  Создайте ярлыки для часто используемых команд Cursor — экономит время и поддерживает фокус.
  - Пример хорошего промпта:
    > “Добавь alias `cg` для `cursor generate` в .bashrc.”
  - Пример плохого промпта:
    > “Каждый раз вручную печатай cursor generate.”

- **Кэширование ответов AI**
  Сохраняйте удачные ответы локально — экономите ресурсы и ускоряете работу.
  - Пример хорошего промпта:
    > “Сохрани ответ на ‘validate email’ в `/prompts/cache/validate_email.md`.”
  - Пример плохого промпта:
    > “Не сохраняй, просто копируй из истории.”

5. Documentation & Knowledge Sharing

- **Автодокументирование (YARD)**
  Генерируйте YARD-комментарии по существующим методам — ускоряет поддержку и onboarding.
  - Пример хорошего промпта:
    > “Сгенерируй YARD-док для `class ShoppingCart; def add_item(item); … end; end`.”
  - Пример плохого промпта:
    > “Добавь комментарии, где нужно.”

- **Документирование промптов**
  Ведите README с описанием используемых Cursor-запросов и их результатов — облегчает обмен знаниями.
  - Пример хорошего промпта:
    > “Обнови `PROMPTS.md` раздел ‘Pagination’: добавить пример запросов.”
  - Пример плохого промпта:
    > “README не трогай.”

- **Регулярные ретроспективы**
  Анализируйте в команде, какие шаблоны и правила работают лучше, и обновляйте их.
  - Пример хорошего промпта:
    > “Собери статистику по использованию промптов за месяц и представь на ретроспективе.”
  - Пример плохого промпта:
    > “Ретроспективу проведём без данных.”

6. Performance & Metrics

- **Запросы на оптимизацию**
  Периодически просите Cursor находить «bottlenecks» в методах — помогает выявлять узкие места.
  - Пример хорошего промпта:
    > “Проанализируй метод `generate_report` на предмет медленных участков.”
  - Пример плохого промпта:
    > “Оптимизируй всё приложение.”

- **Метрики кода**
  Регулярно запрашивайте отчёты по LOC, сложности и другим показателям — отслеживаете технический долг.
  - Пример хорошего промпта:
    > “Сгенерируй отчёт по Cyclomatic Complexity для всех методов в `app/models`.”
  - Пример плохого промпта:
    > “Дай мне все метрики без фильтрации.”

- **Универсальные примеры**
  Включайте в промпты данные из реальных сценариев приложения — повышает применимость сгенерированного кода.
  - Пример хорошего промпта:
    > “Пример заказа: `{user_id:1, items:[{id:5,qty:2}], total:100.0}` для генерации JSON API-ответа.”
  - Пример плохого промпта:
    > “Используй любые данные.”

7. Workflow & Maintenance

- **Регулярное обновление контекста**
  При длительных сессиях перезагружайте историю запросов — избегаете устаревших подсказок и несогласованности.
  - Пример хорошего промпта:
    > “Очисти сессию Cursor и загрузи `init_prompts.md` заново.”
  - Пример плохого промпта:
    > “Продолжай работать с тысячей предыдущих сообщений.”

- **Избегайте чрезмерного доверия**
  При крупных изменениях сначала создавайте draft-ветку с AI-патчами — минимизируете риск поломки.
  - Пример хорошего промпта:
    > “Сгенерируй патч для `app/controllers` в ветке `feature/ai-refactor`.”
  - Пример плохого промпта:
    > “Вмерж в main сразу всё, что предложил AI.”

8. Security & Privacy

- **Конфиденциальность данных**
  Никогда не отправляйте в промптах реальные секреты или ключи — защищаете инфраструктуру.
  - Пример хорошего промпта:
    > “Используй placeholder `ENV['STRIPE_KEY']`, а не реальный ключ.”
  - Пример плохого промпта:
    > “Вставь сюда мой реальный секретный токен.”


THINKNETICA
онлайн-школа для разработчиков