Программирование: как документировать код в 2026 году, чтобы его поняли люди и ИИ-агенты

12.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

Зачем документировать код в 2026 году

Программирование — процесс создания и модификации компьютерных программ, и документация в коде остаётся его неотъемлемой частью. В 2026 году у неё появился второй адресат — ИИ-агенты. Они всё чаще читают репозиторий, чтобы предложить правку, сгенерировать тест или ответить на вопрос. Если комментарии и docstring написаны для людей, агент тоже поймёт контекст. Если это набор бессвязных пометок, ошибётся и человек, и модель.

Ниже — практический подход: какие артефакты документации нужны, как их писать и как проверить, что они работают.

Три уровня документации в коде

Разделяйте документацию по назначению. Смешивать уровни — главная причина, почему комментарии устаревают.

- **Комментарий** — пояснение к конкретному месту: почему сделано именно так, а не иначе. Не пересказывает код.
- **Docstring** (строка документации) — контракт функции, класса или модуля: что принимает, что возвращает, какие ошибки бросает, какие побочные эффекты.
- **README и ADR** (Architecture Decision Record, запись об архитектурном решении) — зачем модуль существует, какие ограничения и компромиссы приняты.

Правило простое: если ответ на вопрос «почему» не помещается в одну строку рядом с кодом — он belongs в ADR или README, а не в комментарий.

Как писать docstring, который поймёт и человек, и агент

ИИ-агенты опираются на сигнатуры и текст docstring. Чем формальнее структура, тем меньше догадок. Для Python в 2026 году де-факто стандарт — стиль Google или NumPy; для TypeScript и JavaScript — TSDoc; для Java — Javadoc. Выберите один на проект и зафиксируйте в линтере.

Минимальный набор в docstring:

1. Одно предложение — что делает функция.
2. Параметры: имя, тип, смысл, допустимые значения.
3. Возвращаемое значение и его тип.
4. Исключения и условия, при которых они возникают.
5. Побочные эффекты: запись в БД, сетевой вызов, изменение глобального состояния.
6. Пример вызова — короткий, рабочий.

Пример: вместо «функция считает скидку» напишите «возвращает итоговую цену с учётом скидки; бросает ValueError, если скидка меньше 0 или больше 100; не меняет переданный объект».

Такой docstring отвечает на вопросы, которые иначе агент станет выяснять экспериментально — а значит, с риском сломать поведение.

Комментарии: когда нужны, а когда вредны

Хороший комментарий объясняет причину, а не механику. Плохой дублирует код и устаревает при первом рефакторинге.

- **Нужны:** обход известного бага с номером тикета, неочевидный порядок вызовов, временное ограничение по нагрузке, ссылка на внешний контракт.
- **Не нужны:** пересказ имени переменной, очевидные циклы, «магия» без пояснения (лучше переписать код).

Если комментарий ссылается на тикет или внешний документ, указывайте идентификатор и дату. Так комментарий можно проверить и удалить, когда причина исчезнет.

Документация как часть процесса, а не отдельная задача

Документация в коде живёт, только если её проверяют автоматически. Практические шаги:

1. Включите линтер docstring (например, pydocstyle для Python или ESLint-правила для TSDoc) в CI. Сборка падает — документация обновляется.
2. Добавьте в шаблон pull request пункт: «изменены ли docstring и README при смене контракта».
3. Храните ADR рядом с кодом, в каталоге docs/adr, с датой и статусом (предложено, принято, заменено).
4. Генерируйте справочник API из docstring автоматически, чтобы он не отставал от кода.

Это не бюрократия: агент, который читает репозиторий, получает тот же контракт, что и человек. Чем меньше расхождений, тем реже он предлагает неверные правки.

Типичные ошибки

- **Документация «на будущее» —** описание функций, которых нет. Агент воспринимает это как факт и строит на этом выводы.
- **Устаревшие примеры —** код в docstring не компилируется. Проверяйте примеры в тестах, если это возможно.
- **Смешение языков —** половина комментариев на русском, половина на английском. Выберите основной язык для кода и придерживайтесь его; исключения — только для внешних терминов.
- **Секреты в комментариях —** пароли, токены и внутренние адреса не должны попадать в репозиторий даже в виде примера.

Чек-лист перед коммитом

1. Docstring описывает контракт, а не реализацию.
2. Комментарии объясняют «почему», а не «что».

3. Примеры в документации запускаются.
4. Ссылки на тикеты и ADR актуальны.
5. Линтер документации проходит без предупреждений.

Документация в коде — такой же инженерный артефакт, как тесты. В 2026 году она экономит время не только команде, но и ИИ-агентам, которые всё чаще работают с репозиторием. Начните с одного модуля и одного правила в линтере — этого достаточно, чтобы увидеть эффект.

Источник: https://cms.dolinais.ru/programmirovanie-kak-dokumentirovat-kod-v-2026-godu-chtoby-ego-ponyali-lyudi-i-ii-agenty