Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

12 правил code-review с AI: наш регламент для команды 42 PM

Зоны допуска AI, pre-commit с semgrep и bandit, CODEOWNERS, лимиты агентов и стенд ai-sandbox

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Команда на AI-помощниках (Claude Code, Copilot, Cursor) генерирует до половины строк в PR: SQL через f-string, голые except, вызовы несуществующих API и опасные автоген-миграции уезжают в main, если процесс ревью не перестроен. Мы внедряем регламент из 12 правил: зоны допуска AI, блокирующий pre-commit-контур, правила веток и изолированный стенд — до того, как баг доедет до пр...

Почему это важно бизнесу

- Одна SQL-инъекция в endpoint с клиентскими данными — утечка ПДн по 152-ФЗ: штрафы, уведомление РКН, остановка продаж
- Молча проглоченная ошибка в платёжном обработчике — недополученная выручка, которую находят через недели по актам сверки
- Без регламента ревью AI-кода дорожает: каждый нестандартный паттерн проверяют вручную, PR застревают, доставка фич тормозит
- Баг, пойманный pre-commit-хуком, стоит минуты; тот же баг на проде — часы разработчика, QA и дежурного плюс репутация

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

v1.172.0

Пин версии хука semgrep в .pre-commit-config.yaml — поведение правил меняется между релизами

по докам semgrep/pre-commit

3 зоны

Допуск AI: сам (CRUD/тесты/docstrings), pair-режим (логика/SQL), запрет (auth/платежи/ПДн)

наш стандарт

>70%

Порог покрытия рукописными юнит-тестами для PR с AI-кодом — тесты от AI не засчитываем

наш стандарт

50 строк

Порог длины SQL, после которого запрос обязан пройти ручной EXPLAIN ANALYZE на прод-объёме

наш стандарт

8–15 мин

Бюджет CI-гейта: полный прогон pytest на стенде ai-sandbox в чистом контейнере на каждый PR

наш стандарт

2 аппрува

Required approvals в ruleset ветки main + Require review from Code Owners для путей auth, bill…

по докам GitHub Rulesets

РЕАЛИЗАЦИЯ 1 ИЗ 3

Pre-commit-контур: semgrep + bandit до коммита

Что настраиваем

Все Python/JS-репозитории клиента; хуки локально у 12 разработчиков, контрольный прогон — в CI на чистом раннере

Как мы это делаем

1. Ставим framework pre-commit 4.x: pre-commit install в каждом клоне, конфиг .pre-commit-config.yaml в корне, версии всех хуков пиним по rev
2. Подключаем хук semgrep (repo semgrep/pre-commit, rev v1.172.0) с args --config .semgrep/ai-rules.yaml --error --skip-unknown-extensions, files на .py/.sql/.js/.ts
3. Пишем кастомные правила: no-fstring-sql (pattern $DB.execute(f"...")), no-bare-except, no-eval-exec — severity ERROR, нарушение блокирует коммит
4. Добавляем bandit (PyCQA 1.9.4, ставим bandit[toml], конфиг в pyproject.toml): B301 pickle, B602 shell=True, B608 SQL-конкатенация — HIGH валит хук
5. CI перепрогоняет pre-commit run --all-files на чистом раннере: локальный обход через git commit --no-verify не проходит мимо пайплайна

РЕЗУЛЬТАТ

40–60% типовых ошибок AI-кода (динамический SQL, голые except, eval/exec, pickle недоверенных данных) отсекаются до код-ревью. Ревьюер смотрит логику, а не ищет шаблонные дыры; время ревью прогнозируемо, а прод не видит инъекций из сгенерированного кода.

КЛЮЧЕВОЙ НЮАНС

Semgrep-правила пишем под паттерны именно AI-кода, а не берём только реестр: registry-пак не знает наших ORM-обёрток. Пин по rev обязателен — набор и поведение правил меняются между релизами.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Зоны допуска: CODEOWNERS + ограничения Claude Code

Что настраиваем

GitHub-репозитории клиента: main под ruleset, критичные пути (auth, billing, migrations) — за ревьюерами-людьми

Как мы это делаем

1. Делим кодовую базу на 3 зоны: AI сам (CRUD, тесты по шаблону, docstrings), pair-режим (бизнес-логика, SQL, интеграции), запрет (auth, платежи, ПДн, миграции)
2. В .github/CODEOWNERS закрепляем /auth/, /billing/, /migrations/ за senior-ревьюерами; ruleset на main: required status checks, 2 approvals и Require review from Cod…
3. В .claude/settings.json: permissions.deny на Edit для auth/** и migrations/**; PreToolUse-хук отдаёт permissionDecision: deny — держит даже bypass-режим
4. Pair-коммиты помечаем тегом [pair-with-claude] в commit message; CI-бот по тегу требует расширенный чек-лист в PR
5. Миграции — только с меткой @migration-by-human: alembic-файл без неё блокируется джобой CI до ручного ревью схемы

РЕЗУЛЬТАТ

AI ускоряет рутину (CRUD, массовые рефакторинги, docstrings), но физически не может править аутентификацию, платёжные интеграции и схему БД: запрет зашит в permissions агента и в правила ветки, а не в устные договорённости команды.

КЛЮЧЕВОЙ НЮАНС

Deny-правила проверяются раньше allow, а PreToolUse-хуки срабатывают даже при --dangerously-skip-permissions — по докам Claude Code это единственный локальный барьер, который агент не обойдёт.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Стенд ai-sandbox: прогон AI-кода на обезличенной копии

Что настраиваем

Отдельная VM 16 vCPU / 32 ГБ / NVMe; Docker Compose-копия прод-стека: Postgres 16, Redis 7, Python 3.12, Node 20, RabbitMQ 3.13,...

Как мы это делаем

1. Разворачиваем compose-стек, идентичный проду по мажорным версиям; секреты только из .env стенда — ни один боевой ключ на VM не попадает
2. Обезличиваем дамп скриптом на Faker: email/телефоны/ФИО — синтетика, суммы умножаем на случайный коэффициент 0,2–5, сохраняя порядок величин
3. Правило PR: фича с AI-кодом прогоняется docker compose up + pytest локально, логи прикладываются; CI повторяет прогон в чистом контейнере за 8–15 минут
4. Для подписок и периодики подменяем время (freezegun): «прокручиваем» 7 дней сценария за секунды и ловим ошибки расписаний
5. Раз в квартал обновляем обезличенный дамп и минорные версии Postgres/Redis, чтобы стенд не расходился с продом

РЕЗУЛЬТАТ

Баги AI-кода, которые всплыли бы только на стейдже или проде (N+1 на реальных объёмах, падения периодических задач, необратимые миграции), ловятся до мержа. Данные клиентов не покидают контур: на стенде нет ни одной живой записи ПДн.

КЛЮЧЕВОЙ НЮАНС

EXPLAIN на 1 000 тестовых строк ничего не доказывает — план запроса меняется на прод-объёме. Обезличенный дамп держим в масштабе прода, а SQL длиннее 50 строк гоняем EXPLAIN ANALYZE руками.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Галлюцинации API**
  AI вызывает метод, которого нет в установленной версии. Обходим: муру в pre-commit, пин зависимостей в lock-файле и grep новых вызовов API по diff пе…
- ✕ **Тесты, подогнанные под код**
  AI генерит тесты под своё же поведение. Шаблон pytest и ассерты фиксирует человек, AI лишь размножает кейсы; порог >70% — только рукописными тестами
- ✕ **Голый except Exception**
  Ошибка молча глотается и всплывает через недели. Правило semgrep no-bare-except (ERROR) плюс обязательный logger.exception в каждом обработчике
- ✕ **DROP COLUMN в автогене миграций**
  alembic autogenerate предлагает удаление колонок при расхождении моделей. Метка @migration-by-human обязательна — без неё CI-джоба блокирует мерж
- ✕ **Хук обойдён через --no-verify**
  Локальный pre-commit — не гарантия: CI обязан перепрогонять pre-commit run --all-files, а semgrep стоит required status check в ruleset ветки
- ✕ **requests.get().json() без таймаутов**
  AI пишет happy-path интеграции. Скелет — от AI, но timeout, retry с backoff и обработку 5xx добавляет человек; semgrep-правилом ловим requests без ti…
- ✕ **AI на проде во время инцидента**
  Правдоподобные, но ложные гипотезы съедают время дежурного. На инциденте — stack trace, логи, метрики; AI подключаем только на пост-мортеме
- ✕ **pickle.load на файлах пользователей**
  Прямой RCE-вектор, который AI предлагает «для простоты». Bandit B301 в блокирующем режиме; приём файлов — только json/строгие схемы и песочница

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- pre-commit с semgrep (пин по rev) + bandit во всех репо, режим блокирующий
- PULL_REQUEST_TEMPLATE.md с чек-листом и полем «писал ли AI» в каждом репозитории
- Запретный список для AI: auth, платёжные интеграции, ПДн — только человек

НОРМАЛЬНО

- CODEOWNERS + ruleset main: required status checks, 2 approvals и Require review from…
- permissions.deny в .claude/settings.json на auth/** и migrations/**
- Тег [pair-with-claude] в коммитах и расширенное ревью по нему
- SQL длиннее 50 строк — ручной EXPLAIN ANALYZE на объёме, близком к проду

ХОРОШО

- ai-sandbox: compose-копия прод-стека с обезличенным дампом (Faker)
- CI-гейт 8–15 мин: pytest на стенде обязателен для каждого PR с AI-кодом
- PreToolUse-хуки с permissionDecision: deny — барьер даже в bypass-режиме
- Квартальная ревизия semgrep-правил по свежим ошибкам из ретроспектив

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Pre-commit с semgrep и bandit стоит во всех репозиториях и валит коммит при severity ERROR?
- [ ] CI перепрогоняет те же проверки на чистом раннере — обход через --no-verify бесполезен?
- [ ] В PR-шаблоне есть поле «использовался ли AI и какой» и оно реально заполняется?
- [ ] Пути auth, платежей и миграций закрыты CODEOWNERS и запрещены агенту в .claude/settings.json?
- [ ] Юнит-тесты к AI-коду написаны человеком, а не сгенерированы под то же самое поведение?
- [ ] Каждая миграция БД помечена @migration-by-human и прошла ручное ревью схемы?
- [ ] Есть стенд с обезличенными данными прод-масштаба для прогона AI-кода до мержа?
- [ ] Секреты берутся из env/секрет-менеджера — grep по репозиторию не находит ключей?
- [ ] Правило «на прод-инциденте — без AI» зафиксировано в runbook дежурного?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Внедряем регламент AI-разработки под ключ: зоны допуска, CODEOWNERS, PR-шаблоны, правила веток — за 2–3 недели
- Пишем кастомные semgrep-правила под ваш стек и ORM, настраиваем pre-commit и CI-гейты
- Разворачиваем ai-sandbox: compose-копия прод-стека, скрипт обезличивания на Faker, квартальное обновление
- Настраиваем ограничения Claude Code/Cursor: permissions, hooks, запретные пути — политикой, а не договорённостью
- Обучаем ревьюеров: чек-лист PR, разбор типовых ошибок AI-кода, сопровождение первые 2 месяца

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Semgrep Docs: Pre-commit — раздел Extensions (docs.semgrep.dev — 2026)

02 semgrep/pre-commit — официальные хуки semgrep и semgrep-ci (github.com — v1.172.0)

03 Claude Code Docs: Settings, Permissions, Hooks (PreToolUse) (code.claude.com — 2026)

04 GitHub Docs: About code owners; Available rules for rulesets (docs.github.com — 2026)

05 Bandit (PyCQA): конфигурация и тесты B301/B602/B608 (bandit.readthedocs.io — 1.9.4)

06 pre-commit framework: установка и .pre-commit-config.yaml (pre-commit.com — 4.x)

07 Наш шаблон: ai-rules.yaml + PULL_REQUEST_TEMPLATE.md (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.