



Junior



Middle



Senior



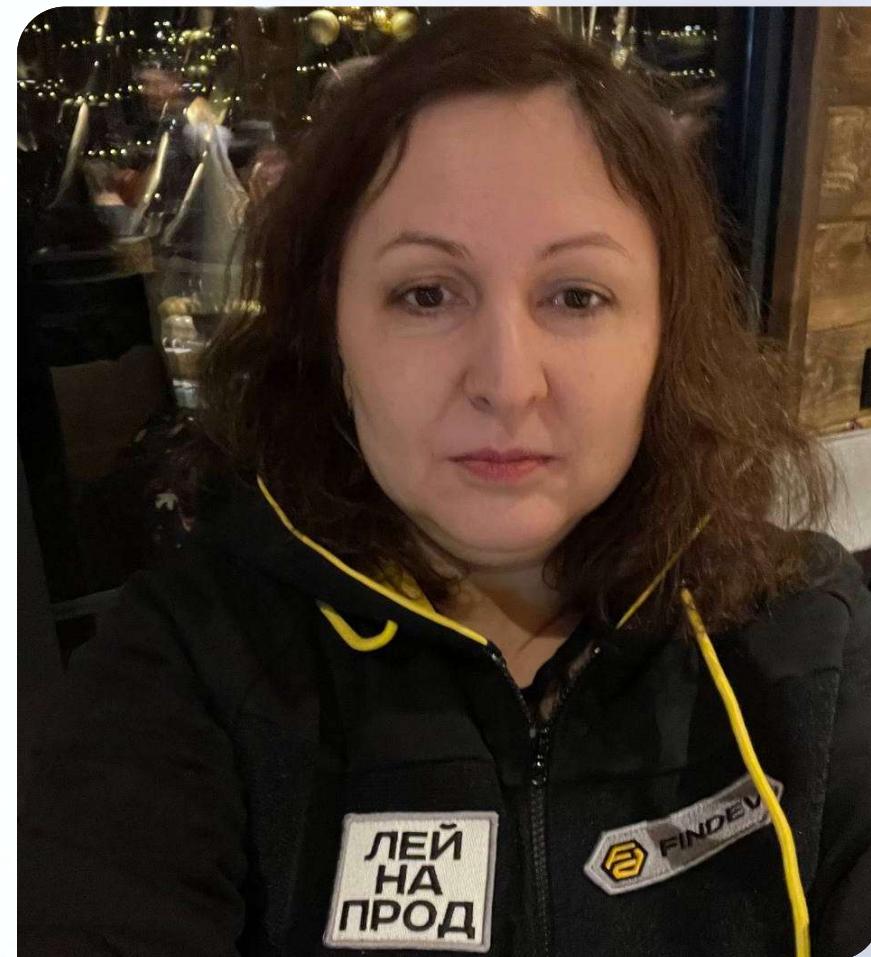
CODE REVIEW: ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ДЕЛЕ



Светлана Мелешкина



Светлана Мелешкина
Ведущий разработчик .NET
НЛМК ИТ



ЗАЧЕМ НАМ CODE REVIEW?



Качество кода

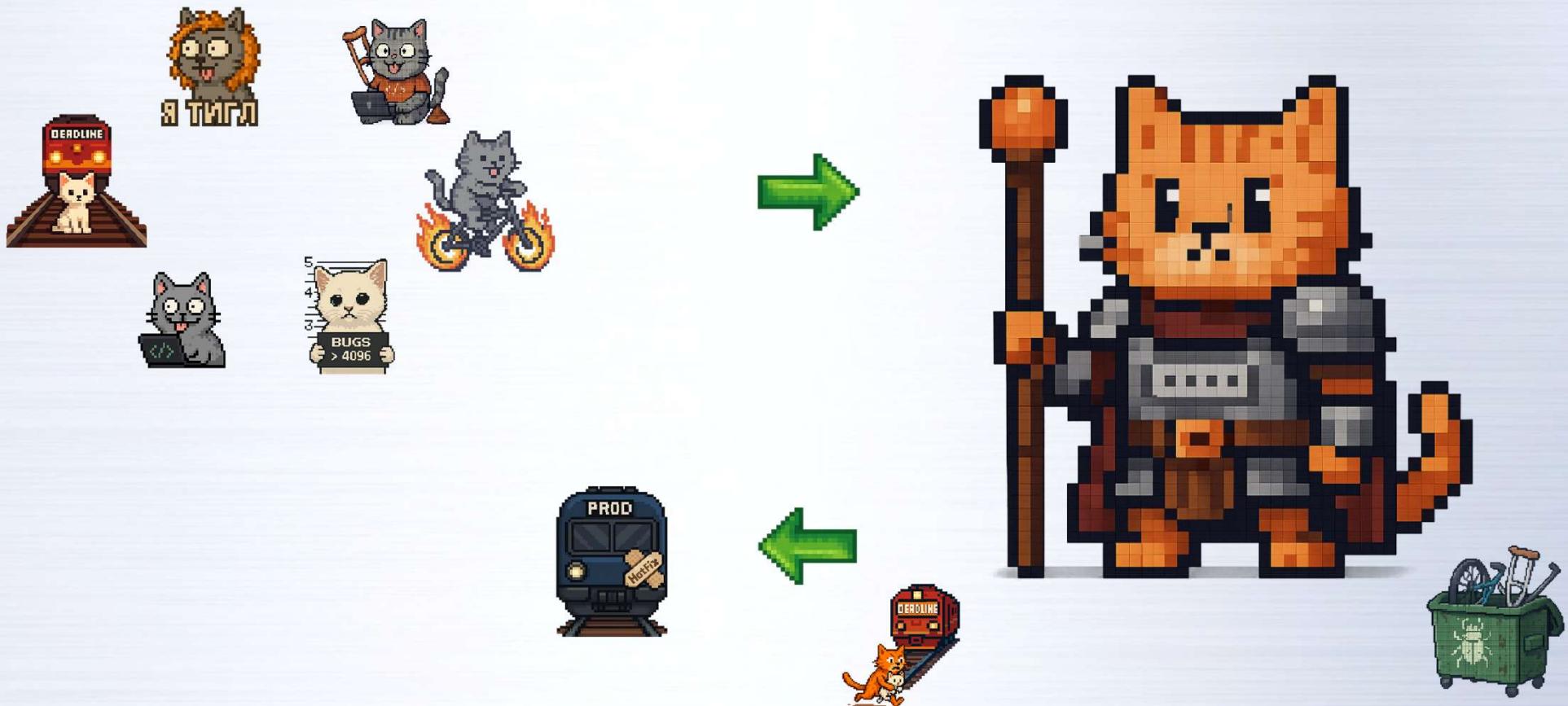
Снижение количества багов

Обучение команды

Снижение bus-factor



А МИНУСЫ БУДУТ?



СТРАСТИ ПО ОТСТУПАМ: АВТОМАТИЗИРУЕМ

- ✖ Пишем все замечания по отступам и стилю
- ✓ Автоматизируем проверки форматирования и стиля



сложно уловить суть: ДЕКОМПОЗИЦИЯ

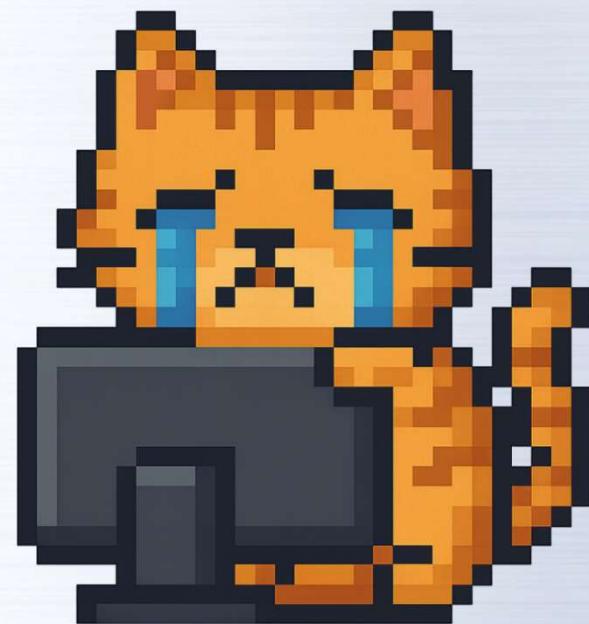
- ✗ На большом MR сразу пишем про каждую мелочь
- ✓ Вначале декомпозиция и крупные проблемы



ТОН РЕШАЕТ ВСЕ: ФОРМУЛИРОВКИ

- ✖ Ты не обработал ошибку.
- ✓ Здесь нужна обработка ошибки.

- ✖ Переименуй переменную.
- ✓ Предлагаю сделать имя переменной более осмысленным.



УВАЖАЕМ СОБЕСЕДНИКА: ОБЪЯСНЯЕМ

- ✖ Сделай так
- ✖ Потому, что я так думаю

- ✓ Объясняем
- ✓ Обосновываем
- ✓ Ссылаемся



REVIEW — КОМАНДНАЯ ИГРА

Проверь свой код

Пиши читаемо и понятно

Комментируй неочевидные места

Конструктивный фидбек <> придирки



ПОДКЛЮЧАЕМ АІ



РИСКИ ПЕРЕДАЧИ КОДА В LLM

- Нарушение законов о персональных данных
- Утечка конфиденциальных данных
- Несоответствие внутренним политикам безопасности
- Обучение модели на вашем коде



ОБУЧЕНИЕ МОДЕЛИ НА ВАШИХ ДАННЫХ: ПРОВЕРЯЕМ ДОГОВОР

- Корпоративный токен
- Ваш личный токен



УТЕЧКА СЕКРЕТОВ: РАЗДЕЛЯЕМ КОД И ДАННЫЕ

Не храним секреты в коде



НЕ ОТДАЕМ КОД НАРУЖУ: LLM НА ТВОЕМ НоУТЕ

- ✓ **OpenAI** GPT OSS 20b
- ✓ **Qwen** 3 Coder 30b
- ✗ **Meta** Code Llama 13b
- ✗ **Mistral** Devstral



НЕ ОТДАЕМ КОД НАРУЖУ: LLM В ИНФРАСТРУКТУРЕ КОМПАНИИ

- ✓ **OpenAI** GPT OSS: 120b
- ✓ **Qwen** 3 Coder: 480b
- ✗ **Deepseek** v3.1: 671b
- ✗ **Zhipu AI** GLM 4.6



AI REVIEW TOOLS

AI-помощники в IDE

Code hosting review

Отдельные AI-ревьюеры

Стат. анализ + AI

PR automation + AI



CODE RABBIT

- RAG
- Использует свои модели
- Встраивается в pipeline
- VSC CLI версия – бесплатная
- Создает промт для исправления проблемы



CLINE

- Запускается локально
- Сам получает diff
- Использует указанную вами LLM



ЧАТ С LLM

- Нужно копировать диффы
- Контекст не дополняется автоматически
- Комментарии в MR надо заполнять вручную
- + Доступно без каких настроек
- + Можно проработать вопросы в том же диалоге



AI ПОМОЩНИК:
ЧЕМ ПОЛЕЗЕН



AI ПОМОЩНИК:

РАСШИФРОВКА ЧТО ДЕЛАЕТ КОД

```
if (x == 0)
{
    print(
        "Hello, world!");
}
```



Этот код выводит фразу Привет Мир, если параметр равен нулю.



AI ПОМОЩНИК: ПРОВЕРКА СООТВЕТСТВИЯ ЗАДАЧЕ

```
if (x == 0)
{
    print(
        "Hello, world!");
}
```

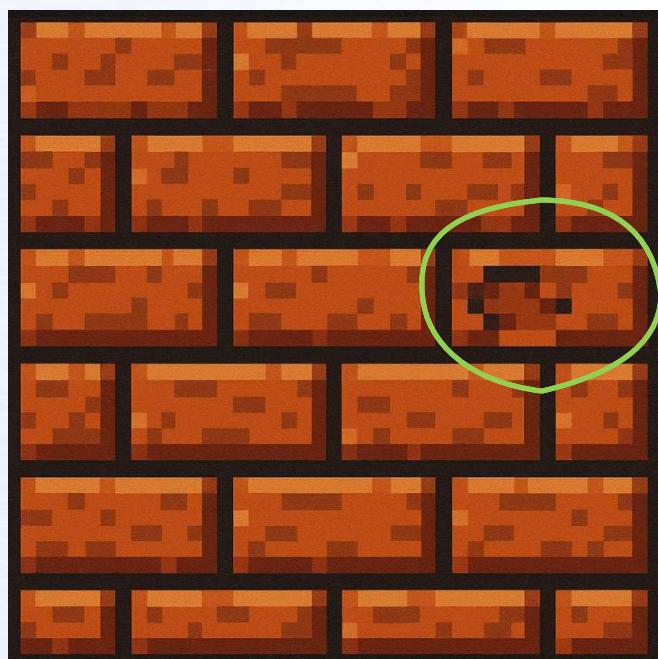


TASK-0001: быть
вежливым всегда.

AI: Реализовано только
при некоторых
значениях параметра.



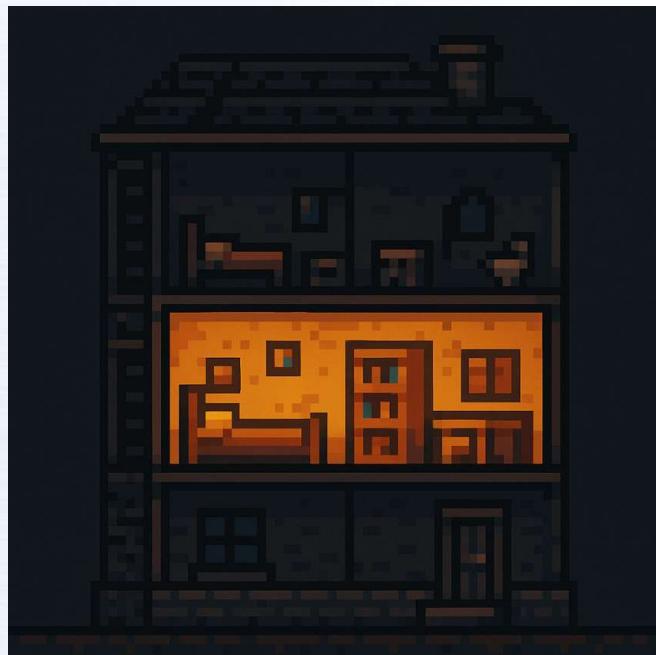
AI ПОМОЩНИК: НАХОДИТ МНОГО МЕЛОЧЕЙ



Дефект



AI ПОМОЩНИК:
ПОДСВЕЧИВАЕТ КУДА СМОТРЕТЬ



**Почему нет
несущей стены?**



AI ПОМОЩНИК:
ОПИШЕТ ПРОБЛЕМУ КОРРЕКТНО



Использовано временное
решение, которое
желательно заменить на
более устойчивое.



AI КЕЙС:

ПРОВЕРЯЕМ СВОЙ КОММИТ



```
git show <commitHash>> diff.txt
```

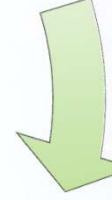
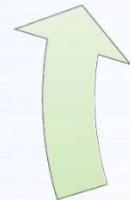




Commit



Review



Доработка



Проверка



MR



PROMT: Проверь мой код на наличие ошибок, потенциальных багов и неэффективных решений. Мне нужна конструктивная критика. Кратко опиши список проблем. Если проблем много сосредоточься на наиболее вероятных.

Задачи решаемые этим кодом:

- Добавить округление значений колонок в соответствие с точностью заданной в справочнике.
- Кешировать этот справочник, и обеспечить обновления кеша при обновление справочника.



PROMT: Формат ответа:

Краткое резюме:

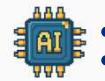
- Итог: [OK/требует доработок/критично]
- Главные риски: ...

Замечания:

1) [severity: blocker/major/minor/nit] Кратко суть проблемы

- Почему это важно: ...
- Доказательство: файл:строка/фрагмент
- Рекомендация: конкретное действие





:Реализация кеширования правил округления

Проблема: Кеш реализован через Lazy<Task>, но обновляется сбросом значения. Возможно состояние **гонки**: если одновременно несколько потоков вызовут ReloadAsync, может получиться несколько запусков инициализации кеша, а Lazy устареет, или, наоборот, можно получить последствия из-за одновременного сброса кеша и его чтения (Lazy не thread-safe по сбросу).

Рекомендация: использовать thread-safe механизм обновления (например, AsyncLazy).





:LazyAsync<T> – это паттерн, позволяющий лениво (по требованию, только один раз) и асинхронно инициализировать некоторый объект.

При первом вызове асинхронной функции (например, `await lazyAsync.Value`) стартует задача инициализации, результат которой кешируется для следующих вызовов. Это аналог Lazy<T>, но для асинхронных операций.

Зачем: Чтобы не инициализировать данные заранее, не делать несколько параллельных запросов на инициализацию и не городить собственные блокировки.



КАКАЯ ПОЛЬЗА



Убираем часть замечаний и багов

Узнаем новые паттерны

Укорачиваем цикл review



AI КЕЙС: ПРОВЕРЯЕМ MR



GitLab web:

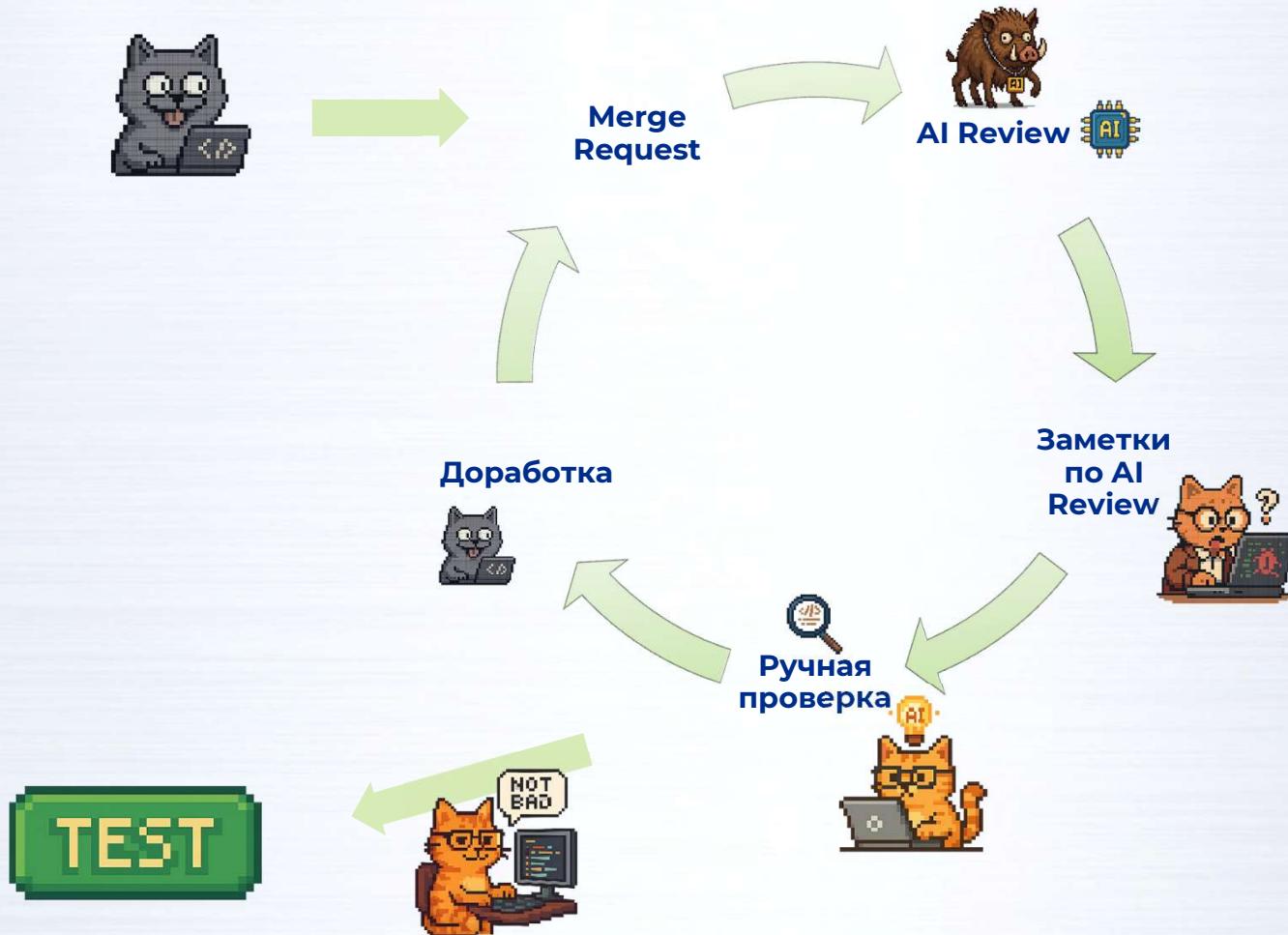
`... merge_requests/1234.diff`

GitLab git:

`git fetch origin merge-requests/1234/head:mr-1234`

`git checkout mr-1234`





1. Просим описание изменений человеческим языком

 : Опиши изменения diff ...

 : Добавлен функционал и API для обнаружения различий между двумя наборами данных.

Реализована **логика сопоставления элементов** по их идентификаторам в двух независимых хранилищах, с возвратом структуры, описывающей расхождения.

Система **асинхронно** перебирает различные подвыборки данных, используя настраиваемый уровень **параллелизма**, и записывает результаты по мере их получения в поток для дальнейшей обработки.



2. Просим провести код ревью

: Проведи **code review**. Обрати особое внимание на следующие аспекты:

Качество и стиль кода: Читаемость, осмысленность имен переменных, методов и классов. Следование рекомендациям C# (*naming conventions, code style guidelines*).

Сложность и эффективность алгоритмов: Оцени временную и пространственную сложность основных алгоритмов если это применимо.

Параллелизм и многопоточность: Если используется параллельная или многопоточная обработка проверь на потенциальные проблемы.

Валидация входных данных и обработка ошибок: Корректна ли обработка исключений. Достаточно ли валидации пользовательского ввода или внешних данных.



2. Задаем формат ответа

 : Формат ответа:

Краткое резюме:

- Итог: [OK/требует доработок/критично]
- Главные риски: ...

Замечания:

1) [severity: blocker/major/minor/nit]

Кратко суть проблемы

- Почему это важно: ...
- Доказательство: фрагмент
- Рекомендация: конкретное действие



3. Review от LLM

 : Контроллер: MismatchController.cs Замечания по API интерфейсу

- Нет ограничения на диапазон дат. Вызовы с широкими диапазонами могут привести к `Timeouts / OutOfMemory`.
- Эндпоинт `'DownloadMismatched'` выставляет `content-type` как `'text/plain; charset=utf-8'`, а должен быть `'text/csv'`



3. Review от LLM

Review:

✗ Опасная конструкция: внутри StreamMismatchedAsync – используется канал без ограничения размера Channel.CreateUnbounded – **теоретически** можно захлебнуться в памяти.

! Итоговый поток через channel разбивается по всем датам параллельно – при большом диапазоне можно получить тысячи параллельных тасков.



4. Модель фантазирует, не все должно попасть в review

 : Как ограничить количество тасков в нашем случае?

 : В вашем коде для параллельной обработки используется `Parallel.ForEachAsync` с настроенным `MaxDegreeOfParallelism`. Количество тасков не будет превышать `MaxDegreeOfParallelism`.

 : Ну и в чем тогда проблема?

 : Ограничение количества задач у вас сделано правильно, ... и дальше LLM начинает выкручиваться в стиле если бы, да бы кабы, то во рту бы росли грибы.



4. Дорабатываем в том же диалоге (у модели есть контекст)



: Можно ли упростить имя `findmismatched`.

 : В REST-сервисах редко используют глаголы в endpoint-ах. Обычно используются существительные. Если метод GET, подразумевается, что вы что-то ищете. Слово "find" – вряд ли имеет ценность для имени эндпоинта.

Лучшие альтернативы: GET /`mismatched?date=YYYY-MM-DD`



5. Досматриваем Review от LLM

Сравнение двух списков

! Для каждого элемента из первой коллекции выполняется поиск соответствия во второй коллекции через перебор элементов, что приводит к сложности $O(N*M)$. Аналогично в обратную сторону. Эффективнее грузить оба списка в структуры с быстрым поиском (например, HashSet) и использовать методы сравнения коллекций (Except, Intersect, Join). При большом объёме данных текущий подход будет неэффективен.



КАКАЯ ПОЛЬЗА



Улучшаем читаемость

Прокачиваем разработчиков

Контролируем архитектуру

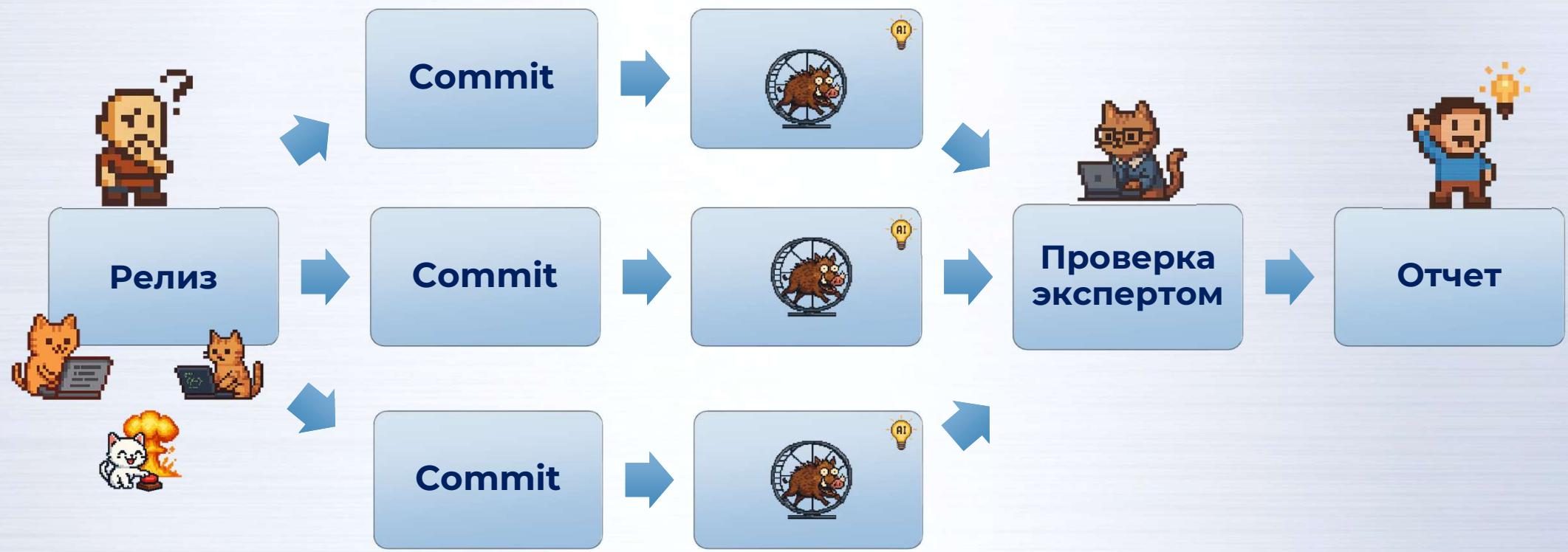
Отлавливаем часть потенциальных
багов



AI КЕЙС: ПРОВЕРКА РЕЛИЗА

diff с прошлым релизом





1. Промт для описания коммита

Кратко опиши изменения внесенные этим коммитом: {diff}



2. Промт для ревью

Проведи code review коммита по задаче.

Описание задачи: {ОПИСАНИЕ ЗАДАЧИ}

Проверь есть ли в изменениях серьезные ошибки или уязвимости, критические баги, возможная потеря или неправильно обработка данных, уязвимости безопасности, серьезные проблемы с производительностью или с многопоточностью в случае их использования.

Соответствуют ли изменения поставленной задаче/описанию.

Не указывай мелкие замечания и стилистические вопросы. Не придумывай несуществующих проблем.

Если серьёзных проблем нет и задача полностью выполнена – просто напиши “OK”.

Формат ответа:

Если есть серьёзные проблемы – кратко и чётко опиши их списком.

Если всё OK – просто напиши “OK”.



Задача: TASK-1234 Некорректное отображение рисунков в конечном письме.

 **описание:** Изменена логика отображения рисунков в письме – теперь к тегу img добавлен CSS-класс для управления отступами, чтобы картинки корректно располагались рядом друг с другом, вместо наложения и неправильного отображения.

 **review:** Добавлен class="margin-left: 10px;", но это некорректно: class используется для CSS-классов, а вам нужен style="margin-left:10px;". Это не работает без дополнительного CSS-класса в стилях. Текущая реализация может не решить проблему визуализации, потому что margin не применится к тегу img из-за ошибки выше. Нужна проверка.



Задача: TASK-1235 При выставленном фильтре Цех/Участок отображаются дочерние отделы без признака, при этом недоступны для выбора.

Описание изменений: Доработана логика фильтрации дочерних подразделений: теперь в дереве отображаются только те отделы, которые либо доступны для выбора, либо содержат внутри себя доступные отделы, чтобы дерево не обрывалось.

AI Review: OK.



КАКАЯ ПОЛЬЗА



Видим изменения не по задачам

Подсвечиваем серьезные проблемы

Описание изменений всего релиза
человеческим языком



AI КЕЙС: ПРИНИМАЕМ ПРОЕКТ

Общая архитектура
Грубые ошибки
Зависимости
Конфигурирование
Логгирование



ОТКРЫВАЕМ ПРОЕКТ В РЕДАКТОРЕ

Собираемость

Зависимости

Конфигурация



С ЧЕГО НАЧАТЬ?

Архитектура и структура проектов

Модели данных

Входные точки (API)

Бизнес-логика (основные модули)



1. Архитектура проекта

 : Проанализируй структуру папок и файлов и сопоставь с описанием проекта.

Твоя задача:

- Кратко опиши, как проект разбит на логические блоки и какие из них требуют внимательного ревью (например, обрабатывают критичные данные, используют параллелизм, интеграции и т.п.).
- Итогом должен быть список блоков для дальнейшего ревью, с пометками (что критично, что можно ревьюить формально).



2. Модели данных и основные сущности

 : Проведи ревью моделей данных и основных сущностей (классы и/или схемы таблиц) из этого кода.

Укажи только существенные проблемы и несоответствия:

- неправильная структура
- отсутствие нужных связей
- несоответствие бизнес-требованию
- нарушение целостности данных
- потенциальные уязвимости
- критически неверная валидация

Если всё хорошо – напиши "OK".

Код/схемы:

{сюда – классы, структуры, фрагменты миграций}



3. API

 : Проанализируй спецификацию Swagger (OpenAPI) для проекта.

Определи, какие бизнес-процессы/модули отражены во внешнем API.

Дай краткое описание ключевых сущностей и процессов.



4. Бизнес логика, ищем самое важное



: Определи основные модули бизнес-логики.

Критерии:

- Это должны быть классы или сервисы, которые реализуют ключевые бизнес-процессы (например, обработка заказов, учет, биллинг, управление пользователями и т.п.).
- Не включай утилиты, репозитории, вспомогательные слои, настройки и инфраструктуру.

Составь список таких модулей с кратким пояснением по каждому (максимум 1–2 предложения).



ИЩЕМ КРИТИЧНОЕ: **REVIEW МОДУЛЕЙ**

Работа с памятью

Работа с многопоточностью

Проблемные алгоритмы

Недостаток логирования

Магические числа в коде





: Проведи ревью кода. Укажи только серьёзные ошибки и уязвимости – не пиши о стилистике и мелких недочётах. Если критических проблем нет – напиши "OK".

Проверь по пунктам:

Память и ресурсы: Утечки памяти, неосвобождённые ресурсы, избыточное удержание объектов, избыточное создание или хранение больших коллекций и объектов. Неосвобожденные файловые дескрипторы, гонки за файлами, утечка сетевых соединений.

Многопоточность: Race condition, deadlock, неправильная синхронизация. Использование небезопасных общих переменных или коллекций.

Алгоритмические проблемы: Неэффективные алгоритмы с высокой асимптотикой по времени или памяти.

Логгирование: Отсутствие логирования в важных для сопровождения местах.

Магические числа: Необъяснённые жёстко прописанные значения, влияющие на бизнес-логику.

Обработка ошибок и исключений: Пустые catch-блоки, "подавленные" ошибки, игнорирование падений.

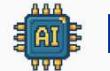
Работа с секретами и чувствительными данными: Захардкоженные ключи, пароли, токены в коде. Риск утечек приватных данных, отсутствие маскирования данных для логов.

Пользовательский ввод: Отсутствие валидации, риск инъекций, опасная сериализация.

Временный код: TODO, DEBUG, Console.WriteLine, закоммиченные заглушки.

Формат ответа: Кратко по номерам пунктов – список критических проблем с коротким описанием. Если критических проблем нет, напиши только: "OK".





RailwayDetector – классический god object: он одновременно занимается I/O-конфигурированием, IPC, обработкой CV, генерацией сигналов, управлением шаред-мемори, даже управляет временем и таймерами.

Нет явного использования ООП для выделения состояний детектора, делегирования поведения (например, стратегия трекинга/разных типов поездов).

Реализация методов слишком длинная (пример: `init` > 200 строк; `wagonLogic` – сравнимо).



 Уровень логирования в текущем файле НЕ соответствует сути ситуации.

Почти ВСЕ события выдаются через LOG(INFO), даже критические ошибки и предупреждения.

Для production-системы и быстрого мониторинга это большой минус: вы не отличите поток штатных событий от настоящих сбоев, не сможете отреагировать на ERROR/WARNING.





Управление памятью / ресурсоемкие конструкции

Без особых RAII-обёрток: нативные дескрипторы шаред памяти, мейки окон OpenCV, работают вручную. Ошибки приведут к ресурсным утечкам.



КАКАЯ ПОЛЬЗА



Больше документации

Улучшена читаемость

Улучшена поддержка

Исправили грубые ошибки



AI КЕЙС: ОЦЕНКА РАЗРАБОТЧИКА

Коммиты за период







: Ты – опытный технический ревьюер в команде разработки. Твоя задача – сформулировать краткое описание и оценить интересность каждого коммита для анализа профессиональных навыков разработчика.

Инструкция: Определи интересность коммита для оценки навыков разработчика: **Interesting**: interesting – если изменения значимые (новая логика, архитектура, сложные тесты, интеграции, нетривиальные багфиксы и т.д.). Interesting: not_interesting – если изменения рутинные, не вносят что-то новое в логику/архитектуру/тесты (стиль, текста, мелкие правки, обновления зависимостей и т.п.).

Требования к ответу: Выводи только в следующем формате одной строкой:

****Interesting:** <interesting/not_interesting>**





: Ты – опытный технический ревьюер в команде разработки. Твоя задача – сформулировать краткое описание каждого коммита для анализа профессиональных навыков разработчика.

Инструкция: Очень кратко опиши, что сделано в коммите (**Summary**, максимум 1–2 предложения). Укажи категорию изменений (**Category**), выбрав строго из вариантов:

feature – новая функциональность

refactor – архитектурные изменения без изменения логики

bugfix – исправление ошибок

test – тесты

doc – документация, комментарии

style – стилистика, форматирование, переименование

chore – обслуживание, инфраструктурные задачи, обновление зависимостей

infra – изменения в CI/CD, конфигурации, окружении

ui_tweak – незначительные UI-правки (верстка, цвета и т.д.)

Требования к ответу: Выводи только в следующем формате markdown (один коммит = один такой блок):

- ****Summary:**** <очень кратко, что сделано>
- ****Category:**** <feature/refactor/bugfix/test/doc/style/chore/infra/ui_tweak>



КАКАЯ ПОЛЬЗА



Быстрая классификация коммитов

Описания изменений к коммитам



МОИ ЛЮБИМЫЕ LLM

OpenAI ChatGPT 4.1

OpenAI ChatGPT 5

OpenAI OSS 120b

Anthropic Claude Sonnet 4.5



VIBE REVIEW НЕ ЗАМЕНЯЕТ ЭКСПЕРТИЗУ



TELEGRAM СТИКЕРЫ: КОТ REVIEW



@svetkis



ВАШИ
ВОПРОСЫ