

Beadloom

Держите архитектуру актуальной и достоверной
— от одного репозитория до ландшафта микросервисов

Презентация для команды · 30 минут · июнь 2026

Зачем мы здесь

Цель встречи: понять, что такое Beadloom, какие проблемы он решает для нас и как им пользоваться в ежедневной работе.

План (30 мин):

Блок	Время
Проблема и позиционирование	~5 мин
Как устроен Beadloom	~7 мин
Живая демонстрация	~10 мин
Федерация и agentic-стек	~5 мин
Roadmap и вопросы	~3 мин

Проблема: расхождение «задумано» и «реально»

Архитектура, документация и код **со временем расходятся** — и за этим никто не следит системно.

Боль	Что происходит
● Контракты между сервисами	Клиент использует поле GraphQL, которое бэкенд убрал — сбой в проде
● Знание в головах	«Как устроено» знают 2 человека; уходят — уходит экспертиза
● Документация врёт	Спека устарела, разработка идёт по неверным данным
● Агент тратит час на ориентацию	grep → чтение → семантические догадки в каждой сессии

IDE ищет код. Beadloom управляет знанием.

Что такое Beadloom

Локальный инструмент (CLI + SQLite + MCP), который превращает архитектуру в **запрашиваемое, проверяемое знание**.

Три столпа внутри одного репозитория:

1. **Context Oracle** — граф в YAML, детерминированный контекст < 20 мс
2. **Doc Sync Engine** — связь код ↔ доки, ловит устаревание на каждом коммите
3. **Architecture Rules** — границы в YAML, `beadloom lint` блокирует в CI

Плюс **федерация** — сверка контрактов между репозиториями (AMQP, GraphQL).

Без Docker, без облака. Один CLI, один файл `.beadloom/beadloom.db`.

Beadloom vs семантический поиск IDE

	IDE (Cursor, Copilot)	Beadloom
Вопрос	«Где этот класс?»	«Что это за фича и как вписана?»
Метод	Эмбединги + LLM	Явный граф + обход
Результат	Вероятностный	Детерминированный
Документация	Не следит	Ловит устаревание
Границы	Не проверяет	Соблюдает, блокирует в CI
Знание	Умирает с сессией	Живёт в Git

Не замена IDE — инфраструктурный слой: данные + правила + честные гейты.

Архитектура: домены

services/ (CLI, MCP, TUI)

↓

application/ (reindex, doctor, gate, site)

↓

context_oracle	doc_sync	graph	onboarding
----------------	----------	-------	------------

↓

infrastructure/ (SQLite, метрики)

6 DDD-доменов + слой use-case'ов + два интерфейса (CLI / MCP / TUI).

Направление зависимостей **принудительно** — правило `architecture-layers` в `rules.yml`.

Как данные текут через систему

```
.beadloom/_graph/*.yaml (граф + правила)
  ↓
beadloom reindex
  ↓
SQLite (.beadloom/beadloom.db)
  ↓
beadloom ctx / prime / search / why
  ↓
Человек или AI-агент (CLI / MCP / TUI)
```

Три источника в индексе:

- YAML-граф (узлы, рёбра, контракты)
- Markdown-документация (привязана к узлам)
- Код (tree-sitter, 12 языков, аннотации `# beadloom:domain=...`)

Состояние проекта прямо сейчас

Метрики с `beadloom status` на этом репозитории:

Метрика	Значение
Версия	1.10.0
Узлы / рёбра	26 / 85
Документы	30
Символы кода	853
Покрытие доками	96% (25/26)
Устаревшие доки	0
Debt Score	10 / 100 (low)
Покрытие тестами	91.6%+

 Портал: zoologov.github.io/beadloom

Architecture as Code: правила в YAML

Правила — не «на ревью, если заметят», а **проверка при сборке**:

```
rules:
  - name: architecture-layers
    layers:
      - { name: services, tag: layer-service }
      - { name: domains, tag: layer-domain }
    enforce: top-down

  - name: tui-no-direct-infra
    forbid_import:
      from: "src/beadloom/tui/**"
      to: "src/beadloom/infrastructure/**"
```

7 типов правил: `require`, `deny`, `forbid`, `layers`, `forbid_cycles`, `forbid_import`, `check`.

```
beadloom lint --strict # exit 1 → CI падает
```

Единый гейт CI: `beadloom ci`

Одна команда — один exit code для пайплайна:

```
reindex → lint → sync-check → config-check → doctor → [federate --fail-on]
```

- **GitHub Action** — готовый composite action
- **GitLab CI** — шаблон
- **Честность по конструкции** — портал и дашборд строятся из тех же путей кода, что и гейты

Агент может предложить что угодно. **Истина — детерминированный гейт.**



Живая демонстрация

~10 минут · терминал + портал

Сценарий — в файле `beadloom-team-30min-notes.md`

Демо 1: Обзор проекта

```
beadloom status          # узлы, покрытие, здоровье
beadloom status --debt-report # оценка архитектурного долга 0-100
beadloom graph           # Mermaid-диаграмма
```

Что показать: 26 узлов, 0 stale docs, debt score 10/100.

Сказать: «Всё это — из одного `reindex`, без ручного ввода метрик».

Демо 2: Контекст фичи

```
beadloom ctx context-oracle --json | head -80  
beadloom why context-oracle  
beadloom search "federation"
```

Что показать: пакет контекста — подграф + доки + символы + **активные правила** для узла.

Сказать: «Агент получает не случайные файлы, а структурированный ответ за < 20 мс».

Демо 3: Синхронность и границы

```
beadloom sync-check      # устаревшие доки → exit 2
beadloom lint            # нарушения границ
beadloom doctor          # целостность графа
```

Что показать: зелёный `sync-check`, чистый `lint`.

Сказать: «На каждом коммите pre-commit hook и CI не дадут уехать в красное».

Демо 4: Контекст для агента

```
beadloom prime          # < 2K токенов — старт сессии агента
beadloom setup-mcp      # настройка MCP для IDE
```

MCP: 18 инструментов — 14 graph/read-write + 4 process-tools (BDL-048):

`task_init` · `bead_context` · `complete_bead` · `checkpoint`

```
{ "mcpServers": { "beadloom": { "command": "beadloom", "args": ["mcp-serve"] } } }
```

Демо 5: Портал документации

Открыть: <https://zoologov.github.io/beadloom/>

1. **Dashboard** — метрики, тренды, рекомендации (ECharts)
2. **Architecture** — интерактивные C4 / Mermaid
3. **Landscape** — карта контрактов с вердиктами

Собирается: `beadloom docs site` → VitePress build.

Федерация: контракты между сервисами

Каждый сервис → `beadloom export` → артефакт с SHA коммита.

Хаб → `beadloom federate` → единый ландшафт.

Вердикт	Значение
<code>CONFIRMED</code>	Поставщик и потребитель совместимы
<code>BREAKING</code>	Потребитель ссылается на то, чего нет в схеме
<code>ORPHANED_CONSUMER</code>	Потребляет, но никто не производит
<code>UNDECLARED_PRODUCER</code>	Производит, но никто не потребляет
<code>EXTERNAL</code>	Внешняя зависимость — без ложных тревог

Dogfood: реальный GraphQL `BREAKING` пойман до релиза.

Agentic-стек: что уже работает (PO)

BDL-047 — AI tech-writer в CI

`sync-check` → Goose-агент чинит **только** устаревшие доки → `beadloom ci` → PR на ревью

BDL-048 — Agentic flow packaging

`beadloom setup-agentic-flow` — воспроизводимый multi-agent flow в любой репо:

`dev → test → review → tech-writer` + coordinator + Beads DAG

BDL-049 — в работе

Trunk-based + AI tech-writer на PR (не на каждый push) → main всегда зелёный

Claude Code + Beadloom + Beads + `beadloom ci`

Кому и как начать

Роль	Что даёт Beadloom
Разработчик	<code>beadloom ctx <фича></code> вместо часа на ориентацию
Тимлид / архитектор	Явный граф в Git, границы в CI
Platform / DevEx	Готовые гейты + MCP для агентов
Работа с ИИ	<code>prime</code> + правила в контексте узла

```
pipx install beadloom
beadloom init --bootstrap
beadloom reindex
beadloom setup-rules && beadloom setup-mcp
```

Roadmap: куда идём

P0  Agentic cluster — AI tech-writer + setup-agentic-flow

P1 Integration map (команда микросервисов):

- Интерактивный landscape (Cytoscape/D3) с pop-up карточками контрактов
- Field-level данные контрактов (GraphQL SDL, AsyncAPI)
- Cross-repo `ctx` — агент на сервисе A видит контракт с B
- `unverified` lifecycle для bootstrap-графа

P2 REST/OpenAPI, PR-bot, Federation-MCP, ownership из CODEOWNERS

Приоритет: **честность > полнота**. Опубликованная ложь хуже отсутствующей фичи.

Ключевые выводы

1. **Beadloom = честная архитектурная правда** — внутри репо и между сервисами
2. **Детерминированный контекст** для людей и агентов — не семантические догадки
3. **CI — единственная точка истинного enforcement** — `beadloom ci`
4. **Уже dogfood'им на себе** — v1.10.0, портал, agentic flow, AI tech-writer
5. **Следующий шаг для команды** — попробовать на своём сервисе: `init --bootstrap` →
PR

Вопросы?

Репозиторий: github.com/zoologov/beadloom

Портал: zoologov.github.io/beadloom

Документация: [docs/getting-started.md](https://github.com/zoologov/beadloom/blob/master/docs/getting-started.md)

Спасибо!