Eval - Тестирование

Схема работы системы Eval

Полная схема флоу с блокировками и таблицами

```
graph TB
    Start([Пользователь запускает тест]) --> StartRun[Eval_StartRun]

    StartRun --> CheckRun{Проверка run_status}
    CheckRun -->|pending| CreateReport[Создать report в eval_reports]
    CheckRun -->|!pending| Error1[Ошибка: run уже запущен]

    CreateReport --> UpdateRun[Обновить eval_runs:<br/>status=running, started_at]
    UpdateRun --> InitLeads[Запустить Eval_InitLead<br/>для каждого run_lead_id]

    InitLeads --> SetBatchTest[Установить batchTest=1]
    SetBatchTest --> SaveContext[Сохранить eval_context:<br/>run_id, suite_id, report_id]
    SaveContext --> ProcessQuestion[Eval_ProcessQuestion]

    ProcessQuestion --> CheckContext{Есть текущая серия<br/>в контексте?}

    CheckContext -->|Да| GetNextQuestion[Взять следующий вопрос<br/>из серии]
    CheckContext -->|Нет| GetUnclaimed[getUnclaimedSeries<br/>report_id, suite_id]

    GetUnclaimed --> CheckTable1[Проверить eval_answers:<br/>найти свободные серии/вопросы]
```

```
    CheckTable1 --> AvailableList{Есть доступные<br/>элементы?}

    AvailableList -->|Нет| AggregateReport[Eval_AggregateReport]
    AvailableList -->|Да| TryLock[Попытка захвата блокировки]

    TryLock --> LockType{Тип элемента?}
    LockType -->|Серия| LockSeries["eval_series_runId_seriesIndex<br/>TTL=60s"]
    LockType -->|Standalone| LockQuestion["eval_q_runId_questionId<br/>TTL=5s"]

    LockSeries --> CheckLock{Блокировка<br/>получена?}
    LockQuestion --> CheckLock

    CheckLock -->|Нет| ReleaseNotNeeded[continue<br/>блокировка не была получена]
    CheckLock -->|Да| DoubleCheck[Двойная проверка:<br/>проверить eval_answers]

    ReleaseNotNeeded --> TryLock

    DoubleCheck --> AlreadyClaimed{Элемент уже<br/>занят?}
    AlreadyClaimed -->|Да| ReleaseLock[bot.releaseLockForBot<br/>освободить блокировку]
    AlreadyClaimed -->|Нет| CreateAnswers[Создать записи в
eval_answers<br/>status=PROCESSING<br/>для всех вопросов серии]

    ReleaseLock --> TryLock

    CreateAnswers --> SaveSeriesContext[Сохранить серию в контекст:<br/>current_series,
answerIds]

    GetNextQuestion --> CheckFirst{Первый вопрос<br/>серии?}
    SaveSeriesContext --> CheckFirst

    CheckFirst -->|Да| ClearHistory[Очистить историю mainAgent]
    CheckFirst -->|Нет| SetQuery[Установить user_query]
    ClearHistory --> SetQuery

    SetQuery --> MARouter[MA_Router<br/>обработка вопроса ботом]

    MARouter --> CheckBatchTest{batchTest==1?}
    CheckBatchTest -->|Нет| NormalFlow[Обычный flow]
    CheckBatchTest -->|Да| SaveAnswer[Eval_SaveAnswer]
```

```
    SaveAnswer --> GetSession[Получить session_id<br/>из LLMTracer]
    GetSession --> SaveActual[Сохранить actual_answer<br/>в eval_answers]
    SaveActual --> GetMetrics[getTraceMetrics<br/>из llm_tracer]
    GetMetrics --> SaveMetrics[Сохранить метрики:<br/>latency_ms, tokens, etc.]

    SaveMetrics --> LockCounter["eval_run_runId_counter<br/>TTL=10s, wait=30s"]
    LockCounter --> IncrementCounter[incrementProcessedQuestions<br/>в eval_reports]
    IncrementCounter --> JudgeAnswer[Eval_JudgeAnswer]

    JudgeAnswer --> CallJudge[Вызов LLM-accecсора]
    CallJudge --> ParseJSON{Успешно<br/>распарсить JSON?}

    ParseJSON -->|Да| SaveRatings[Сохранить ratings:<br/>overall, accuracy,
etc.<br/>status=COMPLETED]
    ParseJSON -->|Нет| SaveDefault[Сохранить дефолтные<br/>ratings=50<br/>status=COMPLETED]

    SaveRatings --> UpdateSeriesIdx[Увеличить current_series_question_idx]
    SaveDefault --> UpdateSeriesIdx
    UpdateSeriesIdx --> ProcessQuestion

    MARouter -->|Ошибка| ErrorFallback[RAG_ErrorFallback]
    ErrorFallback --> CheckBatchTest2{batchTest==1?}
    CheckBatchTest2 -->|Да| HandleError[Eval_HandleError]
    CheckBatchTest2 -->|Нет| NormalError[Обычная обработка ошибки]

    HandleError --> MarkFailed[markAnswerFailed<br/>status=FAILED<br/>в eval_answers]
    MarkFailed --> IncrementFailed[incrementFailedQuestions<br/>в eval_reports]
    IncrementFailed --> ProcessQuestion

    AggregateReport --> CheckAllProcessed[areAllQuestionsProcessed<br/>report_id, suite_id]

    CheckAllProcessed --> CheckStatuses[Проверить статусы через table.find:<br/>PENDING, PROCESSING<br/>используя IN условие]
    CheckStatuses --> CheckSeries[Проверить целостность серий:<br/>все вопросы в финальных статусах<br/>COMPLETED или FAILED]

    CheckSeries --> AllDone{Все обработано?}
    AllDone -->|Нет| SendStatusMsg[Отправить статус главному лиду:<br/>обработка продолжается]
```

```
    AllDone -->|Да| FinalizeLock["eval_run_runId_finalize<br/>TTL=60s"]

    SendStatusMsg --> ExitFlow

    FinalizeLock --> LockAcquired{Блокировка<br/>получена?}
    LockAcquired -->|Нет| OtherLeadFinalizes[Другой лид финализирует<br/>и отправит уведомление]
    LockAcquired -->|Да| FinalizeReport[finalizeReport:<br/>calculateReportStats<br/>status=COMPLETED]

    OtherLeadFinalizes --> ExitFlow
    FinalizeReport --> FinalizeRun[finalizeRun:<br/>status=COMPLETED<br/>finished_at]

    FinalizeRun --> SendNotify[Eval_Notify<br/>отправить уведомление]

    SendNotify --> UpdateNotify[Обновить notification_sent=1]
    UpdateNotify --> ExitFlow

    Note1[Блокировка финализации гарантирует<br/>единственность отправки уведомления]

    ExitFlow --> End([Завершение])

    style Start fill:#e1f5ff
    style End fill:#e1f5ff
    style Error1 fill:#ffcccc
    style HandleError fill:#ffcccc
    style MarkFailed fill:#ffcccc
    style LockSeries fill:#fff4cc
    style LockQuestion fill:#fff4cc
    style LockCounter fill:#fff4cc
    style FinalizeLock fill:#fff4cc
    style NotifyLock fill:#fff4cc
    style CreateReport fill:#ccffcc
    style CreateAnswers fill:#ccffcc
    style SaveRatings fill:#ccffcc
    style FinalizeReport fill:#ccffcc
```

Схема таблиц и их связи

```
erDiagram
    eval_runs ||--o{ eval_reports : "has"
    eval_suites ||--o{ eval_questions : "contains"
    eval_suites ||--o{ eval_runs : "used_in"
    eval_reports ||--o{ eval_answers : "contains"
    eval_questions ||--o{ eval_answers : "answered_by"

    eval_runs {
        int id PK
        int suite_id FK
        text run_lead_ids "lead1,lead2,lead3"
        text run_status "pending|running|completed|failed"
        datetime started_at
        datetime finished_at
    }

    eval_suites {
        int id PK
        text name
        int bot_id
    }

    eval_questions {
        int id PK
        int suite_id FK
        text question_text
        text optimal_answer
        int series_index "null для standalone"
        int order_in_series "1,2,3..."
    }

    eval_reports {
        int id PK
        int run_id FK
        datetime started_at
        datetime finished_at
        int total_questions
        int processed_questions
        int failed_questions
```

```
        decimal average_rating
        text report_status "running| completed| failed"
        int notification_sent
    }

    eval_answers {
        int id PK
        int report_id FK
        int question_id FK
        text lead_id
        text session_id
        text question_text
        text optimal_answer
        text actual_answer
        text answer_status "pending| processing| completed| failed"
        decimal rating_overall
        decimal rating_accuracy
        int latency_ms
        int input_tokens
        int output_tokens
        text error_message
    }

    llm_tracer {
        text session_id PK
        decimal outclient_time
        int input_tokens
        int output_tokens
    }
```

Схема блокировок и их жизненный цикл

```
sequenceDiagram
    participant L1 as Lead 1
    participant L2 as Lead 2
```

```
participant DB as eval_answers
participant Lock as Lock System

Note over L1,L2: Параллельная обработка вопросов

L1->>DB: getUnclaimedSeries(report_id, suite_id)
DB-->>L1: [Серия 1, Серия 2, Вопрос 3]

L2->>DB: getUnclaimedSeries(report_id, suite_id)
DB-->>L2: [Серия 1, Серия 2, Вопрос 3]

L1->>Lock: waitForBotLock("eval_series_{runId}_1")
Lock-->>L1: true (получена)

L2->>Lock: waitForBotLock("eval_series_{runId}_1")
Lock-->>L2: false (занята L1)

L1->>DB: Двойная проверка: серия свободна?
DB-->>L1: Да, свободна

L1->>DB: Создать записи status=PROCESSING
DB-->>L1: answerIds: [101, 102, 103]

L2->>Lock: waitForBotLock("eval_series_{runId}_2")
Lock-->>L2: true (получена)

L2->>DB: Двойная проверка: серия свободна?
DB-->>L2: Да, свободна

L2->>DB: Создать записи status=PROCESSING
DB-->>L2: answerIds: [104, 105]

Note over L1: Обработка вопросов серии 1
L1->>DB: Обновить answer_id=101: actual_answer, status=COMPLETED

Note over L2: Обработка вопросов серии 2
L2->>DB: Обновить answer_id=104: actual_answer, status=COMPLETED

Note over L1,L2: Блокировки автоматически освобождаются по TTL
```

```
    Note over L1,L2: или при ошибке через releaseLockForBot()
```

Схема обработки ошибок

```
flowchart TD
    Start[Обработка вопроса] --> Process[MA_Router обрабатывает]

    Process --> Success{Успешно?}
    Process --> Timeout{Таймаут?}
    Process --> Error{Ошибка API?}

    Success -->|Да| SaveAnswer[Eval_SaveAnswer]
    SaveAnswer --> Judge[Eval_JudgeAnswer]

    Judge --> JudgeSuccess{Accecсор<br/>ответил?}
    JudgeSuccess -->|Да| SaveRatings[Сохранить ratings<br/>status=COMPLETED]
    JudgeSuccess -->|Нет| SaveDefault[Сохранить дефолт<br/>status=COMPLETED]

    SaveRatings --> NextQuestion[Следующий вопрос]
    SaveDefault --> NextQuestion

    Timeout -->|Да| ErrorFallback[RAG_ErrorFallback]
    Error -->|Да| ErrorFallback

    ErrorFallback --> CheckBatch{batchTest==1?}
    CheckBatch -->|Нет| UserError[Показать ошибку<br/>пользователю]
    CheckBatch -->|Да| HandleError[Eval_HandleError]

    HandleError --> MarkFailed[markAnswerFailed<br/>status=FAILED<br/>error_message]
    MarkFailed --> IncrementFailed[incrementFailedQuestions]
    IncrementFailed --> NextQuestion

    NextQuestion --> CheckMore{Есть ещё<br/>вопросы?}
    CheckMore -->|Да| Process
    CheckMore -->|Нет| Aggregate[Eval_AggregateReport]

    Aggregate --> CheckAll{Все вопросы<br/>обработаны?}
```

```
    CheckAll -->|Нет|  Wait[Ждать завершения<br/>других лидов]
    CheckAll -->|Да|  Finalize[Финализация report]

    style Error fill:#ffcccc
    style Timeout fill:#ffcccc
    style HandleError fill:#ffcccc
    style MarkFailed fill:#ffcccc
    style Success fill:#ccffcc
    style SaveRatings fill:#ccffcc
    style Finalize fill:#ccffcc
```

Схема статусов вопросов

```
stateDiagram-v2
    [*] --> PENDING: Создан в claimSeries

    PENDING --> PROCESSING: Захвачен лидом

    PROCESSING --> COMPLETED: Успешно обработан<br/>и оценён
    PROCESSING --> FAILED: Ошибка при обработке

    COMPLETED --> [*]
    FAILED --> [*]

    note right of PROCESSING
        Блокировка активна
        TTL: 5s (вопрос) или 60s (серия)
    end note

    note right of COMPLETED
        Финальный статус
        Включает ratings и метрики
    end note
```

Схема параллельной обработки (Run Leads Pool)

```
graph LR
    subgraph "Run Leads Pool"
        L1[Lead 1]
        L2[Lead 2]
        L3[Lead 3]
    end

    subgraph "Общий Report"
        R[(eval_reports<br/>report_id=1)]
    end

    subgraph "Вопросы Suite"
        Q1[Серия 1: Q1, Q2, Q3]
        Q2[Серия 2: Q4, Q5]
        Q3[Вопрос 6]
        Q4[Вопрос 7]
    end

    subgraph "Блокировки (run_id=5)"
        B1[eval_series_5_1]
        B2[eval_series_5_2]
        B3[eval_q_5_6]
        B4[eval_q_5_7]
    end

    L1 -->|claimSeries| B1
    L2 -->|claimSeries| B2
    L3 -->|claimSeries| B3

    B1 --> Q1
    B2 --> Q2
    B3 --> Q3
```

```
    Q1 --> R
    Q2 --> R
    Q3 --> R

    L1 -.->|Параллельно| L2
    L2 -.->|Параллельно| L3

    style R fill:#ccffcc
    style B1 fill:#fff4cc
    style B2 fill:#fff4cc
    style B3 fill:#fff4cc
    style B4 fill:#fff4cc
```

Ключевые моменты

Блокировки

- Все блокировки привязаны к `run_id` для изоляции параллельных запусков
- Формат: `{prefix}_{runId}_{identifier}`
- TTL: 5s для вопросов, 60s для серий, 60s для финализации
- Освобождаются автоматически по TTL или через `bot.releaseLockForBot()`

Таблицы

- `eval_runs` - запуски тестов
- `eval_reports` - отчёты (один на run)
- `eval_answers` - ответы на вопросы (много на report)
- `eval_questions` - вопросы из suite
- `llm_tracer` - метрики производительности

Обработка ошибок

- Ошибки обрабатываются через `Eval_HandleError`
- Вопрос помечается как `FAILED` с `error_message`
- Тест продолжается со следующим вопросом
- Не влияет на другие лиды в пуле
- FAILED вопросы можно переобработать (не считаются занятыми)

Оптимизация запросов

- Используется `table.find` с условием `IN` для проверки статусов
- Заменены `filter/map` на прямые запросы к БД где возможно
- Проверка активных статусов: `["answer_status", "IN", ["pending", "processing", "completed"]]`
- Проверка финальных статусов: `["answer_status", "IN", ["completed", "failed"]]`

Серии vs Standalone

- Серии захватываются целиком (все вопросы сразу)
- Standalone вопросы захватываются по одному
- История очищается перед началом новой серии
- Целостность серий проверяется при финализации

Версия #1
Павел Борисов создал 25 December 2025 21:35:32
Павел Борисов обновил 25 December 2025 21:36:56