Расширенная архитектура Performance Review + Task / Goal Review

Общая цель

Соединить **оценку эффективности по задачам (Goals, Tasks)** с **оценкой компетенций (Hard/Soft skills)**.
Теперь сотрудник получает не только ревью по навыкам через фиксированные интервалы (1м, 3м, 6м),
но и **оценку по каждой ключевой задаче**, которую он выполняет.

Структура данных

1. Таблица `Tasks`

Хранит основные рабочие задачи сотрудника.

Поле	Тип	Описание
`id_task`	UUID	Идентификатор задачи
`title`	string	Краткое описание задачи
`description`	text	Детальное описание цели
`id_employer`	UUID	Исполнитель (сотрудник)
`id_creator`	UUID	Автор задачи (руководитель / HR)
`start_date`	date	Дата начала
`end_date`	date	Плановая дата завершения
`status`	enum('active','completed','review')	Статус
`goal_id`	UUID	Ссылка на цель (если задача часть цели)

2. Таблица `Goals`

Хранит стратегические цели, из которых формируются задачи.

Поле	Тип	Описание
`id_goal`	UUID	ID цели
`title`	string	Название цели
`description`	text	Цель в терминах OKR или KPI
`id_employer`	UUID	Сотрудник, ответственный за достижение
`create_date`	date	Дата постановки цели
`deadline`	date	Плановая дата завершения
`status`	enum('active','review','done')	Статус цели

3. Таблица `Questions`

Хранит вопросы, по которым проводится оценка (и для компетенций, и для задач).

Поле	Тип	Описание
`id_question`	UUID	Идентификатор вопроса

Поле	Тип	Описание
id_role	UUID	Роль (для подбора релевантных вопросов)
context	enum('skill','task','goal')	Тип контекста
question_text	text	Формулировка вопроса
category	string	Категория оценки (например, ответственность, качество, планирование)

4. Таблица `Answers_TaskReview`

Фиксирует ответы на вопросы, связанные с выполнением задачи.

Поле	Тип	Описание
id_answer	UUID	ID записи
id_task	UUID	Ссылка на задачу
id_question	UUID	Вопрос
id_employer	UUID	Сотрудник, которого оценивают
id_respondent	UUID	Сотрудник, который оценивает
grade	int (0–10)	Оценка
update_time	timestamp	Дата обновления

Логика:

- если `id_employer = id_respondent` → это **самооценка**
- если разные → это **оценка коллег**

5. Таблица `Review_Schedule`

Определяет, **когда** нужно запускать тестирование по задачам и целям.

Поле	Тип	Описание
id_schedule	UUID	ID расписания
context	enum('goal','task','skill')	Тип ревью
related_id	UUID	ID задачи или цели
review_start	date	Дата старта опроса
review_end	date	Дата завершения
status	enum('pending','in_progress','completed')	Состояние цикла ревью

Логика работы

1. Создание задачи и цели

При создании записи в `Tasks` и `Goals` автоматически формируется **расписание оценки (`Review_Schedule`)**, где в `review_start` записывается дата `end_date` задачи.

2. Подготовка теста

Когда наступает дата `review_start`, система создаёт:

- в `Answers_TaskReview` — по одной записи на каждый вопрос, релевантный роли сотрудника (`grade = 0`);
- одноразовые ссылки для самооценки и для коллег (через API аналогичный API#1).

3. Прохождение опроса

Сотрудник и коллеги заполняют шкалу от **0 до 10** для каждого вопроса.
Вопросы подгружаются из `Questions`, где `context = 'task'` или `'goal'` и `id_role` совпадает с ролью респондента.

4. Завершение и расчёт результатов

После завершения периода (`review_end`):

- рассчитывается средний балл по каждому вопросу и категории;
- рассчитывается **дельта self / peer**;
- результаты записываются в агрегированную таблицу (аналог `Analytics` из предыдущего кейса);
- в дальнейшем эти данные включаются в общий `AdaptationIndex`.

Пример взаимосвязи

```
Goal → включает несколько Tasks
Task → запускает Review_Schedule
Review_Schedule → инициирует тестирование (вопросы + оценки)
Answers_TaskReview → хранит баллы 0–10 от self и peers
```

API #1 — Создание задач и целей (Task & Goal Creation)

Цель

Позволить сотруднику (или его руководителю) добавить в систему:

- **цель** (Goal) — стратегическое направление работы,
- **и/или задачу** (Task) — конкретное измеримое действие в рамках цели.

Эти записи станут основой для последующего ревью и расчёта эффективности выполнения.

Endpoint

```
POST /api/review/task-goal/create
```

Входные данные

```
{
  "creator_id": "a0b91234-ff78-4bc2-a903-bfd321b91c67",
  "employer_id": "2e4aa83a-1cb2-4a5e-a433-8f10c4ac6c9b",
  "goal": {
    "title": "Повысить стабильность инфраструктуры",
    "description": "Снизить количество критических инцидентов до
2 в месяц",
    "deadline": "2025-12-01"
  },
  "tasks": [
    {
      "title": "Настроить систему мониторинга серверов",
      "description": "Внедрить Zabbix и Grafana на все
прод-сервера",
      "start_date": "2025-10-25",
      "end_date": "2025-11-15"
    },
    {
      "title": "Оптимизировать SLA по заявкам",
      "description": "Сократить среднее время обработки с 3
часов до 1.5",
      "start_date": "2025-11-16",
      "end_date": "2025-12-01"
    }
  ]
}
```

Основная логика выполнения

1. **Проверка полномочий**
   - Если `creator_id == employer_id` → сотрудник ставит себе задачу (самоинициатива)
   - Иначе — руководитель ставит задачу подчинённому.
2. **Создание цели**
   - Проверяем, есть ли активная цель с таким же `title` у сотрудника.
   - Если нет — создаём новую запись в `Goals`.

3. **Создание задач**
    - Для каждой задачи из массива создаём запись в `Tasks` с `goal_id` текущей цели.
    - Каждая задача получает `status = "active"` и уникальный `id_task`.
4. **Создание записи в `Review_Schedule`**
    - Для каждой задачи создаётся расписание ревью:
    - {
    - "context": "task",
    - "related_id": "{id_task}",
    - "review_start": "{end_date}",
    - "review_end": "{end_date + 7 days}",
    - "status": "pending"
    - }
    - То есть ревью по задаче начнётся в момент её завершения и будет активно неделю.

Выходные данные

```
{
  "status": "success",
  "goal_id": "cb27a0c7-6b2f-4c1b-9135-55d203f86e5d",
  "tasks_created": [
    {
      "task_id": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f",
      "title": "Настроить систему мониторинга серверов",
      "review_start": "2025-11-15",
      "review_end": "2025-11-22"
    },
    {
      "task_id": "a48ddc8c-c6d1-4bfa-b8c9-72b2408b8e21",
      "title": "Оптимизировать SLA по заявкам",
      "review_start": "2025-12-01",
      "review_end": "2025-12-08"
    }
  ]
}
```

Проверки и валидации

- `deadline` цели не может быть раньше, чем `end_date` последней задачи.
- Каждая задача должна иметь уникальное название в рамках одной цели.
- Если не указана цель (`goal`), система создаёт “техническую цель” с типом `"Без категории"`.

API #2 — Автоматический запуск ревью после завершения задачи

Цель

Когда наступает или проходит `end_date` задачи,
API автоматически:

1. проверяет, какие задачи готовы к ревью,
2. создаёт для них шаблон ответов (`grade = 0`),
3. формирует одноразовые ссылки для самооценки и коллег,
4. обновляет статус задачи и расписания.

Endpoint

```
POST /api/review/task-review/start
```

Входные параметры

Можно вызывать двумя способами:

1. Автоматически (cron / планировщик) — без тела.
2. Ручной запуск (для конкретной задачи):

```
{
  "task_id": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f"
}
```

Основная логика

1. Поиск задач для ревью

Система ищет задачи:

```
SELECT * FROM Tasks
WHERE end_date <= CURRENT_DATE
  AND status = 'active';
```

2. Создание записей в `Answers_TaskReview`

Для каждой найденной задачи:

- получаем `id_employer` (кого оцениваем),
- выбираем коллег из `Team_Relations` (кто оценивает),
- определяем роль сотрудника (`id_role`),
- подбираем релевантные вопросы из `Questions`, где `context = 'task'` и `id_role` совпадает.

Для каждого вопроса создаём строки:

```
{
  "id_task": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f",
  "id_question": "Q123",
  "id_employer": "2e4aa83a-1cb2-4a5e-a433-8f10c4ac6c9b",
  "id_respondent": "2e4aa83a-1cb2-4a5e-a433-8f10c4ac6c9b",
  "grade": 0,
  "update_time": null
}
```

(такая запись создаётся и для self, и для каждого peer)

3. Создание одноразовых ссылок

Для каждого респондента формируется ссылка:

`https://review.company.app/form?token={uuid4}`

Она хранится в таблице `Review_Logs`:

Поле	Значение
`token`	UUID
`id_task`	ссылка на задачу
`id_respondent`	кто оценивает
`expires_at`	`review_start + 1 day`
`status`	`"pending"`

4. Рассылка уведомлений

На e-mail респондентов (и самого сотрудника) отправляется письмо с темой:

“Оцените выполнение задачи: Настроить систему мониторинга серверов”
и ссылкой для прохождения.

5. Обновление статусов

После рассылки:

- `Tasks.status = 'review'`
- `Review_Schedule.status = 'in_progress'`

Выходные данные

```
{
  "status": "success",
  "tasks_initiated": [
    {
      "task_id": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f",
      "review_links": [
        {
          "respondent": "2e4aa83a-1cb2-4a5e-a433-8f10c4ac6c9b",
          "type": "self",
          "link":
"https://review.company.app/form?token=7c51a18e-39b4-4d0a-b2e2-6
5d23d5b6302"
        },
        {
          "respondent": "b5a3ef13-918c-48a9-b16d-6e0b92a9d5ee",
          "type": "peer",
```

```
          "link":
"https://review.company.app/form?token=1b5a12d7-714f-4934-99cd-9
b6cfa07b601"
        }
      ]
    }
  ]
}
```

Особенности логики

Условие	Поведение
Сотрудник уволен (`dismissal_date < today`)	Пропускается
Нет коллег в `Team_Relations`	Создаётся только self-test
Если уже есть `Answers_TaskReview` с `grade=0`	Повтор не создаётся
Если задача в статусе `review` или `done`	Не дублируется

Автоматизация

API вызывается ежедневно (например, в 06:00) и формирует ревью по всем завершённым задачам за последние 24 часа.
Это обеспечивает **постоянный цикл обратной связи** — сотрудник завершает задачу → система инициирует её оценку.

API #3 — Получение формы ревью по задаче (Task Review Form)

Цель

Когда сотрудник или коллега переходит по одноразовой ссылке из письма,
API #3:

- проверяет, действительна ли ссылка (токен),
- определяет, кто оценивает и какую задачу,
- подбирает вопросы из соответствующих категорий,
- возвращает структуру формы для фронтенда (вопросы, описание задачи, мета-информация).

Endpoint

`GET /api/review/task/form?token={uuid}`

Входные данные

```
{
  "token": "1b5a12d7-714f-4934-99cd-9b6cfa07b601"
}
```

Основная логика

1. Проверка токена

Поиск токена в таблице `Review_Logs`.

- Проверяем срок действия (`expires_at > now()`);
- Проверяем статус (`pending`).
  Если токен просрочен или уже использован → возвращаем:

```
{ "status": "error", "message": "Link expired or already used" }
```

2. Определение контекста

По токену подтягиваем:

- `id_task` — идентификатор задачи,
- `id_respondent` — кто оценивает,
- `id_employer` — кого оценивают,
- `id_role` — роль оцениваемого (для подбора вопросов).

3. Загрузка информации о задаче

Из таблицы `Tasks`:

```
SELECT title, description, start_date, end_date, status
FROM Tasks WHERE id_task = {id_task};
```

4. Подбор вопросов

Из таблицы `Questions`:

```
SELECT id_question, question_text, category
FROM Questions
WHERE context = 'task'
  AND (id_role = {id_role} OR id_role IS NULL);
```

Если в таблице `Answers_TaskReview` уже есть заполненные ответы (`grade > 0`), они подтягиваются для редактирования/просмотра.

Выходные данные

```
{
  "status": "success",
  "task": {
```

```
    "id": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f",
    "title": "Настроить систему мониторинга серверов",
    "description": "Внедрить Zabbix и Grafana на все
прод-сервера",
    "start_date": "2025-10-25",
    "end_date": "2025-11-15"
  },
  "review": {
    "type": "peer",
    "respondent_id": "b5a3ef13-918c-48a9-b16d-6e0b92a9d5ee",
    "employer_id": "2e4aa83a-1cb2-4a5e-a433-8f10c4ac6c9b"
  },
  "questions": [
    {
      "category": "Ответственность",
      "items": [
        { "id": "Q101", "text": "Сотрудник соблюдал сроки
выполнения задачи?", "grade": 0 },
        { "id": "Q102", "text": "Проявлял ли инициативу при
решении проблем?", "grade": 0 }
      ]
    },
    {
      "category": "Качество выполнения",
      "items": [
        { "id": "Q201", "text": "Насколько аккуратно и полно
выполнена задача?", "grade": 0 },
        { "id": "Q202", "text": "Были ли ошибки, требующие
доработки?", "grade": 0 }
      ]
    }
  ],
  "link_valid_until": "2025-10-23T12:00:00Z"
}
```

Сценарии поведения

Сценарий	Поведение
Self-review	`id_respondent = id_employer`, блок отображает “Самооценка по задаче”
Peer-review	разные ID, отображается “Оценка коллеги для задачи …“
Черновик	ранее сохранённые оценки подставляются
Просрочено	фронт отображает страницу «Срок действия ссылки истёк»
Уже завершено	API возвращает `"status": "completed"` и режим только чтения

API #4 — Сохранение ответов (Task Review Submit)

Цель

Сохранять оценки (0–10) по каждой задаче после того, как сотрудник или коллега прошли ревью через форму.
Фиксировать факт завершения опроса, обновлять время обновления и закрывать токен.

Endpoint

```
POST /api/review/task/submit
```

Входные данные

```
{
  "token": "1b5a12d7-714f-4934-99cd-9b6cfa07b601",
  "answers": [
    { "id_question": "Q101", "grade": 8 },
    { "id_question": "Q102", "grade": 9 },
    { "id_question": "Q201", "grade": 7 },
    { "id_question": "Q202", "grade": 10 }
  ]
}
```

Основная логика

1. Проверка токена

- Проверяем валидность токена в `Review_Logs`.
- Если токен просрочен → `error: "Link expired"`.
- Если статус = `completed` → `error: "Review already submitted"`.

2. Получение контекста

Из `Review_Logs` подтягиваем:

- `id_task`
- `id_respondent`
- `id_employer`

3. Сохранение ответов

Для каждой пары `(id_task, id_question)` обновляем `Answers_TaskReview`:

```
UPDATE Answers_TaskReview
SET grade = {grade}, update_time = NOW()
WHERE id_task = {id_task}
  AND id_question = {id_question}
  AND id_respondent = {id_respondent};
```

4. Проверка завершения

Если у данного `id_respondent` по задаче **все вопросы имеют `grade > 0`**,
то статус токена обновляется:

```
UPDATE Review_Logs SET status = 'completed' WHERE token =
{token};
```

5. Проверка завершения всей задачи

Если все участники (self и peers) завершили свои ревью:

```
SELECT COUNT(*) FROM Answers_TaskReview
WHERE id_task = {id_task} AND grade = 0;
```

Если `0` →

- `Tasks.status = 'done'`
- `Review_Schedule.status = 'completed'`

Выходные данные

```
{
  "status": "success",
  "task_id": "0b18a1e3-f57e-4c3b-9474-472a5b4d884f",
  "respondent_id": "b5a3ef13-918c-48a9-b16d-6e0b92a9d5ee",
  "answers_saved": 4,
  "review_completed": true
```

}

Дополнительные проверки

Условие	Поведение
Некорректный `id_question`	Ошибка `400: invalid question`
Оценка вне диапазона 0–10	Ошибка `400: invalid grade range`
Токен повторно используется	`status: error, message: review already submitted`

Архитектура Performance Review 360 + Task Review

Цель системы

Система Performance Review предназначена для:

- отслеживания адаптации и эффективности сотрудников по soft/hard skills;
- оценки динамики профессионального роста через регулярные ревью;
- анализа достижения целей и задач с использованием шкалы 0–10;
- совмещения самооценки и оценки коллег (360);
- выявления областей риска и зон для развития.

Общая структура данных

Ключевые таблицы:

Таблица	Назначение
`Employer`	Справочник сотрудников: ФИО, должность, e-mail, дата трудоустройства
`Review_Periods`	Периоды ревью (1, 3, 6, 12 месяцев)
`Hard_Skills`, `Soft_Skills`	Справочники категорий и вопросов (1–10 баллов)
`Goals`	Цели, поставленные сотрудником или руководителем
`Tasks`	Задачи, связанные с целями
`Questions`	Вопросы по ролям и категориям для ревью
`Answers_Review`	Ответы сотрудников и коллег (основное хранилище оценок)
`Answers_TaskReview`	Ответы по конкретным задачам и целям

Таблица	Назначение
`Review_Stats`	Агрегированные результаты и динамика (TPI, тренды, флаги)

Логика потоков данных (end-to-end flow)

Представь это как «цепочку автоматического цикла жизни ревью»:

```
[1] Создание задач и целей
    ↓
[2] Проверка наступления даты ревью
    ↓
[3] Генерация одноразовых ссылок
    ↓
[4] Прохождение ревью (self + peers)
    ↓
[5] Сохранение результатов
    ↓
[6] Аналитика и визуализация
```

API-Цепочка (внутренняя логика системы)

API #1 — Task & Goal Creation

Роль: исполнитель или руководитель.
Создаёт цели и задачи с описанием, сроком и связанной категорией.

`POST /api/review/task/create`
Функция: записывает цели в `Goals`, задачи в `Tasks`, с привязкой к `employer_id` и `period_id`.
→ Эта запись запускает "потенциал ревью" — в будущем к ней прикрепятся тесты.

API #2 — Review Trigger

Роль: системный планировщик (cron).
Каждый день проверяет, настала ли дата ревью для конкретного сотрудника.

`POST /api/review/trigger`
Функция:
Если дата совпадает с `employment_date + period_interval`, то система:

- создаёт шаблон записей с `grade=0` в `Answers_Review`;
- генерирует одноразовые ссылки;
- рассылает уведомления:
  - сотруднику — для self-review,
  - коллегам — для peer-review.

API #3 — Review Form Load

Роль: получатель ссылки (сотрудник или коллега).
При открытии формы система:

- проверяет валидность токена (24 часа);
- подгружает вопросы из `Questions`, фильтруя по `role_id` и `review_type` (self/peer);
- выводит диапазон шкалы 0–10;
- позволяет пройти анкету без авторизации.

```
GET /api/review/form?token={uuid}
```

API #4 — Save Review Answers

Роль: фронтенд при отправке формы.
Фиксирует ответы в `Answers_Review` или `Answers_TaskReview`.

```
POST /api/review/submit
```

Функция:
Обновляет записи с `grade > 0`.
Если все ответы сохранены → статус `completed = true`.

API #5 — Aggregation & Analytics

Роль: аналитика/HR.
Обрабатывает все накопленные данные.

```
GET /api/review/task/summary?employee_id={uuid}
```

Результат:

- средние баллы по категориям;
- разница Self vs Peers;
- динамика по месяцам;
- индексы адаптации (AIndex) и эффективности (TPI);
- флаги:
  - ⚠ `переоценка` (разница > 2),
  - 🟢 `рост`,
  - 🔴 `падение`.

Архитектурная визуализация (словесная блок-схема)

Метрики и визуальные показатели

Метрика	Источник	Интерпретация
SelfAvg	Answers_Review	Самооценка
PeerAvg	Answers_Review	Средняя оценка коллег
TPI	Aggregation	Индекс результативности
AdaptationIndex	Aggregation	Индекс адаптации
DeltaTrend	Aggregation	Изменение баллов за 3+ месяцев
ReviewCompliance	Log / Review	% сотрудников, прошедших ревью
GoalCompletion	Tasks / Goals	% выполненных целей

Метрика	Источник	Интерпретация
`TaskFocusScore`	Aggregation	Соответствие задач целям

🧩 Интеграции и расширения

- **Frontend** → SPA на React/Vue. Работает по одноразовым токенам, без auth.
- **Backend** → Node.js + PostgreSQL. Cron-процессы через `bull` или `Celery`.
- **Email Gateway** → SendGrid / Mailgun.
- **Analytics** → Grafana или встроенные SVG-графики (API #5).
- **Расширения**:
  - интеграция с Jira / Asana для автоматического импорта задач;
  - связка с HRM для учёта увольнений и отпусков;
  - экспорт результатов в CSV/Excel по API.