

SE-01-TS13

Техническое задание

Backend-доработки MoodBoss: Ассистент, карточка события, калории и примерочная эмоций

Параметр	Значение
Версия	v1.4 - полная редакция без сметы
Дата	30.04.2026
Статус	Итоговая техническая постановка для backend-реализации первого коммерческого релиза. Смета вынесена в отдельный документ.
Связанные документы	SE-01-TS11 - СУК MoodBoss; SE-01-SM13 - смета TS13; Матрица зависимостей СУК-МП v2.1

1. Предмет работ

Настоящее техническое задание фиксирует полный объём backend-доработок MoodBoss для первого коммерческого релиза по следующим направлениям:

- гибридная модель Ассистента: быстрые кнопки и запуск тех же сценариев из обычного текстового запроса;
- карточка события как базовая сквозная сущность продукта;
- калорийный сценарий внутри Ассистента с сохранением результата в карточку события и отображением в разделе Здоровье;
- примерочная эмоций как сценарий работы с карточкой события внутри Ассистента;
- темы Ассистента как отдельные чаты / cases без верхнеуровневой иерархии проектов;
- техническая готовность backend к последующей интеграции с СУК без реализации полноценного СУК в рамках настоящего ТЗ

Документ не содержит часов, стоимости, платежных условий и коммерческого графика. Эти сведения фиксируются только в отдельной смете SE-01-SM13.

2. Цель работ

Цель работ - превратить Ассистента из простого текстового чата и набора отдельных кнопок в рабочий сценарный контур MoodBoss, который помогает пользователю перейти от запроса к сохранённому структурированному объекту продукта

Backend должен обеспечить:

- запуск сценария через быструю кнопку на экране Ассистента;
- запуск того же сценария обычным текстовым запросом в чате;
- определение намерения пользователя по ключевым сценариям;
- уточняющие вопросы при недостатке данных;
- structured actions для frontend;
- создание draft / pending action;
- preview результата до сохранения;
- сохранение карточки события, калорийного результата или результата примерочной только после подтверждения пользователя;
- хранение результата как структурированного объекта backend, а не только как истории чата.

3. Архитектурная формула релиза

Ассистент -> кнопка или текстовый запрос -> понимание намерения -> уточняющие вопросы -> actions для frontend -> draft -> preview -> подтверждение пользователя -> EventCard -> Health / Calendar.

Эта формула является базовым правилом реализации. Никакой структурированный объект не должен сохраняться молча без подтверждения пользователя, если речь идёт о карточке события, калориях или результате примерочной эмоций.

4. Базовые принципы реализации

- Кнопка и текстовый запрос ведут в один сценарный контур backend, а не в разные независимые реализации.

- Backend должен возвращать frontend не только текстовый ответ, но и структурированные блоки: quick replies, action buttons, preview, status block.
- EventCard является центральной сущностью, к которой привязываются сценарии, если их результат должен стать фактом дня пользователя.
- Health отображает агрегаты и данные по состоянию, включая баланс калорий, но сам калорийный сценарий запускается через Ассистента.
- Calendar является основным контуром жизненного цикла карточки события.
- Примерочная эмоций не является отдельным верхнеуровневым разделом первого релиза, а работает как сценарий Ассистента и как связанный результат EventCard.
- Темы Ассистента трактуются как отдельные чаты / cases. Верхнеуровневые Projects в scope первого этапа не входят.
- СУК, витрины, агрегаты и управленческая аналитика реализуются отдельным T3 SE-01-TS11. В TS13 включается только техническая готовность backend к передаче событий и связей.

5. Ассистент: гибридная модель

5.1. Общая концепция

В первом коммерческом релизе подтверждается гибридная модель Ассистента. Быстрые кнопки сохраняются как простой вход в сценарии, но не являются единственным способом запуска. Те же сценарии должны запускаться из обычного текстового запроса пользователя в чате.

Способ запуска	Целевая backend-логика	Комментарий
Кнопка на экране Ассистента	Frontend передаёт известный scenario_type. Backend запускает общий сценарный контур: уточнение -> draft -> preview -> confirm -> save.	Кнопка не должна создавать отдельную backend-ветку.
Текстовый запрос в чате	Backend / Ассистент определяет intent_type, задаёт уточняющие вопросы и возвращает structured actions для frontend.	Текстовые запросы должны приводить к тем же действиям, что и кнопки.
Вложение: фото / материал	Backend связывает вложение с сообщением, использует его в сценарии и при необходимости создаёт черновик результата.	Актуально для калорий и примерочной.

5.2. Быстрые кнопки на экране Ассистента

- создать событие;
- запустить калорийный сценарий;
- запустить примерочную эмоций / фото-видео студию в доступном для первого релиза объёме;
- продолжить обычный чат с Ассистентом

Нажатие кнопки должно передавать известный сценарный тип в единый сценарный контур Ассистента. Ответ backend должен быть совместим с тем же механизмом draft / preview / confirm, который используется при запуске из текста.

5.3. Запуск из обычного чата

Запрос пользователя	Намерение	Ожидаемое действие
Завтра у меня сложная встреча, я переживаю	event_creation + possible emotion try_on	Создать черновик события, предложить примерочную эмоций, показать preview.
Я съел омлет, хлеб и кофе	calorie_tracking	Запустить калорийный сценарий, уточнить порцию, показать preview результата.
Хочу примерить спокойную уверенность перед разговором	emotion_try_on	Связать примерочную с событием или предложить создать событие
Что со мной сегодня происходит?	general_assistant / health_question	Дать структурированный ответ без обязательного создания EventCard, если нет действия

5.4. Structured response contract

Backend должен поддерживать ответ Ассистента не только в виде текста, но и в виде структурированных блоков для frontend.

- quick_replies - быстрые варианты ответа пользователя;
- action_buttons - действия: сохранить, изменить, отменить, уточнить;
- event_card_preview - preview карточки события;
- calorie_result_preview - preview результата калорийного сценария;
- emotion_try_on_preview - preview результата примерочной эмоций;
- status_block - статус сценария: draft_created, waiting_confirmation, saved, cancelled, failed.

Поле	Назначение
message	Текстовый ответ Ассистента для пользователя.
blocks	Массив структурированных блоков для frontend.
metadata.scenario_type	Тип сценария: event_creation, calorie_tracking, emotion_try_on, general_assistant.
metadata.source	Источник запуска: assistant_button, assistant_chat_text, attachment.
metadata.status	Текущий статус сценария.
draft_id / pending_action_id	Идентификатор черновика / ожидающего действия.

5.5. Draft / PendingAction

Ассистент не должен сохранять структурированные объекты без подтверждения пользователя. Для этого вводится сценарный черновик / ожидающее действие.

1. Пользователь нажимает кнопку или пишет текстовый запрос.
2. Backend определяет сценарий и недостающие данные.
3. Ассистент задаёт уточняющие вопросы или формирует предварительный результат.
4. Backend создаёт draft / pending_action.
5. Frontend показывает preview.
6. Пользователь подтверждает, редактирует или отменяет.
7. Backend сохраняет EventCard и связанные данные только после подтверждения.

Статус	Описание
draft_created	Черновик создан, но ещё не показан пользователю как финальный результат.
waiting_clarification	Backend ждёт уточнения от пользователя.
preview_shown	Frontend показал пользователю preview.
confirmed	Пользователь подтвердил сохранение.
saved	Backend успешно сохранил структурированный объект.
cancelled	Пользователь отменил действие.
failed	Сценарий завершился ошибкой, структурированный объект не должен быть частично сохранён без статуса.

6. Карточка события

6.1. Концепция

Карточка события является базовой сквозной сущностью MoodBoss. Сценарии Ассистента, примерочная эмоций и калорийный сценарий не должны жить отдельно от карточки события, если их результат имеет значение как факт дня пользователя.

6.2. Источники создания

- календарь.
- быстрая кнопка на экране Ассистента;
- текстовый запрос в чате Ассистента;
- примерочная эмоций;
- калорийный сценарий;
- Health-контур, если событие связано с состоянием или активностью пользователя.

6.3. Минимальный состав карточки события

Поле	Назначение	Комментарий
event_card_id	Идентификатор карточки	Обязателен для связи с Calendar, Health и будущим СУК.
user_id / profile_id	Пользователь и профиль	Нужны для персонального контекста.
title	Название события	Может быть предложено Ассистентом и изменено пользователем.
description	Описание	Краткое содержание события или результата сценария.
local_date / time	Дата и время	С учётом timezone пользователя.
source	Источник создания	assistant_button, assistant_chat_text, calendar, health, emotion_try_on
scenano_type	Тип сценария	event_creation, calorie_tracking, emotion_try_on
status	Статус	draft, preview_shown, confirmed, saved, cancelled, archived.
assistant_chat_id	Связь с темой / чатом	Если событие создано из Ассистента.
assistant_message_id	Связь с сообщением	Для трассировки результата.
health_entry_id	Связь с Health	Если событие связано со здоровьем или калориями.
calorie_entry_id	Связь с калориями	Если событие является приёмом пищи.
emotion_try_on_result_id	Связь с примерочной	Если сохранён результат примерочной.

6.4. Основные операции

- создание черновика карточки;
- получение preview карточки;
- редактирование черновика до сохранения;
- подтверждение сохранения;
- получение полной карточки по идентификатору;
- базовое обновление сохранённой карточки;
- архивирование / удаление карточки по отдельному действию пользователя

6.5. Правило создания после подтверждения

Событие event_card_created как факт создания сохранённой карточки должно фиксироваться только после подтверждения пользователем. До подтверждения используется draft_id / pending_action_id или EventCard со статусом draft, если это требуется backend-моделью. Для СУК и приёмки сохранённой карточкой считается только объект со статусом confirmed / saved.

7. Калорийный сценарий

7.1. Общая концепция

Калорийный сценарий является сценарием внутри Ассистента. Результат должен сохраняться в карточку события и отображаться в разделе Здоровье. Ручной ввод калорий как отдельная форма первого уровня в score первого релиза не входит.

7.2. Способы запуска

- нажатие кнопки Калории на экране Ассистента;
- текстовое описание еды или напитка в чате;
- фото блюда или напитка;
- фото с текстовой подписью;
- уточняющий диалог по порции, составу или времени приёма пищи

7.3. Целевой сценарий

1. Пользователь запускает сценарий кнопкой или текстовым запросом.
2. Ассистент определяет `calorie_tracking` или получает `scenario_type` от кнопки.
3. При недостатке данных Ассистент задаёт уточняющие вопросы.
4. Backend выполняет расчёт калорийности прихода.
5. Frontend получает `calorie_result_preview`.
6. Пользователь подтверждает, уточняет или отменяет результат.
7. После подтверждения создаётся EventCard типа `meal / food` и связанная запись `Health / calories`.
8. Раздел Здоровье получает обновлённые данные баланса.

7.4. Scope первого релиза по калориям

- ИИ-сценарий расчёта калорийности по фото, тексту и их комбинации;
- диалоговое уточнение порции / состава;
- preview результата до сохранения;
- сохранение прихода калорий после подтверждения;
- использование существующих `activity`-данных как источника расхода;
- отображение текущего баланса на момент обращения;
- минимально необходимый API для экрана баланса калорий внутри Здоровья.

7.5. Не входит в первый релиз

- внешние `food database`;
- `barcode scanning`;
- сложная нутриентная аналитика;
- расширенные рекомендации по рациону;
- отдельный раздел первого уровня для калорий;
- ручной ввод калорий как самостоятельная форма первого уровня.

8. Примерочная эмоций

8.1. Общая концепция

Примерочная эмоций является сценарием работы с карточкой события внутри Ассистента. Она не должна жить отдельно от карточки события, если результат сценария сохраняется как факт или подготовка к событию.

8.2. Способы запуска

- кнопка на экране Ассистента;
- текстовый запрос пользователя в чате;
- предложение Ассистента внутри диалога по уже созданному событию;
- переход из `preview` карточки события.

8.3. Целевой сценарий

1. Пользователь создаёт или выбирает событие.
2. Ассистент предлагает запустить примерочную эмоций, если это релевантно.
3. Пользователь выбирает эмоциональный режим или формулирует его текстом.
4. Backend формирует результат примерочной.
5. Frontend показывает emotion_try_on_preview.
6. Пользователь подтверждает сохранение результата.
7. Результат сохраняется в EventCard и связанном контексте дня пользователя.

8.4. Score первого релиза по примерочной

- запуск сценария внутри Ассистента;
- получение базового результата;
- preview результата;
- сохранение результата как части EventCard;
- связь результата с календарным контуром MoodBoss.

8.5. Не входит в первый релиз

- отдельный самостоятельный продуктовый раздел первого уровня;
- расширенный набор нестандартных режимов;
- сложная аналитика поведения пользователя по сценарию примерки;
- сложная генеративная студия beyond score первого релиза, если она не описана отдельным ТЗ.

9. Темы Ассистента

В первом этапе тема трактуется как отдельный чат / case внутри Ассистента. Верхнеуровневые проекты, объединяющие несколько чатов, в первый этап не входят.

- создать тему;
- продолжить тему;
- переименовать тему;
- удалить тему;
- видеть список тем.

9.1. Правило сохранности данных

Удаление темы не должно автоматически удалять сохраненные карточки событий, данные Здоровья, калорийные записи, результаты примерочной или долгосрочный профиль пользователя. Рекомендуемая backend-модель для первого релиза - soft delete / архивирование темы вместо физического удаления, если тема связана с сохраненными объектами продукта

10. Backend-контракты и OpenAPI

В рамках реализации необходимо обновить OpenAPI и backend-контракты по следующим группам endpoints / операций:

- создание / список / переименование / удаление тем как отдельных чатов / cases;
- отправка сообщения Ассистенту с поддержкой structured response;
- запуск сценария из кнопки;
- создание draft / pending action;
- получение preview;
- редактирование draft;
- confirm / cancel draft;
- создание и получение EventCard;
- сохранение calorie result в EventCard и Health;
- сохранение emotion try-on result в EventCard;
- получение данных для экрана баланса калорий внутри Здоровья.

Группа API	Ожидаемые операции
Assistant messages	send message, attach material, return structured blocks, return metadata
Scenario router	start by button, start by intent, continue scenario, fail gracefully
Draft / PendingAction	create, get, update, confirm, cancel, expire
EventCard	create draft, preview, confirm, get, update, archive
Calories	start, clarify, preview, confirm, save, get balance
Emotion try-on	start, preview, confirm, save result
Topics	list, create, continue, rename, soft delete

11. Детальная модель данных

Настоящий раздел является частью основного ТЗ, а не приложением. Он фиксирует минимальную целевую модель данных для реализации TS13. Конкретные названия таблиц, сериализаторов и классов могут быть адаптированы под текущую backend-архитектуру MoodBoss, но поля, связи и статусы должны быть сохранены по смыслу.

11.1. EventCard

EventCard - базовая сквозная сущность MoodBoss, связывающая Ассистент, Calendar, Health, калорийный сценарий и примерочную эмоций.

Поле	Тип / формат	Обязательность	Описание
event_card_id	uuid / string	обязательно	Уникальный идентификатор карточки события.
user_id	uuid / string	обязательно	Пользователь-владелец карточки.
profile_id	uuid / string	обязательно	Профиль пользователя, к которому относится событие.
title	string	обязательно после preview	Название события, предложенное Ассистентом или заданное пользователем.
description	string / nullable	желательно	Описание события или результата сценария.
event_type	enum	обязательно	meeting, meal, emotion_try_on, health, custom, other.
scenario_type	enum	обязательно при создании из Ассистента	event_creation, calorie_tracking, emotion_try_on, general_assistant.
source	enum	обязательно	assistant_button, assistant_chat_text, calendar, health, emotion_try_on.
status	enum	обязательно	draft, preview_shown, confirmed, saved, cancelled, archived, failed.
local_date	date	обязательно	Локальная дата пользователя.
time	time / nullable	опционально	Локальное время события, если применимо.
timezone	string	обязательно	Timezone пользователя.
assistant_chat_id / topic_id	uuid / string / nullable	если создано из Ассистента	Связь с темой Ассистента.
assistant_message_id	uuid / string / nullable	если создано из сообщения	Связь с сообщением Ассистента.
draft_id / pending_action_id	uuid / string / nullable	до сохранения	Связь с черновиком.
health_entry_id	uuid / string / nullable	если есть Health-связь	Связь с Health.
calorie_entry_id	uuid / string / nullable	если это meal / food	Связь с калорийной записью.
emotion_try_on_result_id	uuid / string / nullable	если есть результат примерочной	Связь с результатом примерочной.
created_at / updated_at	datetime UTC	обязательно	Служебные временные метки.
confirmed_at / cancelled_at	datetime UTC / nullable	по статусу	Метки подтверждения или отмены.

Связи EventCard: один пользователь может иметь много EventCard; одна EventCard может быть связана с одним Draft / PendingAction до сохранения; одна EventCard может иметь одну CalorieEntry или один EmotionTryOnResult в рамках первого релиза: удаление темы Ассистента не удаляет EventCard

11.2. Draft / PendingAction

Draft / PendingAction фиксирует промежуточный результат сценария до подтверждения пользователем. Он нужен для preview, редактирования, cancel и защиты от молчаливого сохранения.

Поле	Тип / формат	Обязательность	Описание
draft_id / pending_action_id	uuid / string	обязательно	Уникальный идентификатор черновика.
user_id	uuid / string	обязательно	Владелец черновика.
profile_id	uuid / string	обязательно	Профиль, к которому относится действие.
assistant_chat_id / topic_id	uuid / string / nullable	если создано в чате	Тема Ассистента.
assistant_message_id	uuid / string / nullable	если создано из сообщения	Исходное сообщение пользователя.
scenario_type	enum	обязательно	event_creation, calorie_tracking, emotion_try_on.
intent_type	enum / nullable	если определен intent	Распознанное намерение пользователя.
source	enum	обязательно	assistant_button, assistant_chat_text, attachment.
status	enum	обязательно	draft_created, waiting_clarification, preview_shown, confirmed, cancelled, expired, failed.
payload	json	обязательно	Структурированные данные черновика.
preview_payload	json / nullable	после preview	Данные, показанные пользователю.
idempotency_key	string	обязательно для confirm	Ключ защиты от повторного сохранения.
expires_at	datetime UTC / nullable	желательно	Срок жизни черновика.
created_at / updated_at	datetime UTC	обязательно	Служебные временные метки.

11.3. CalorieEntry

CalorieEntry фиксирует результат калорийного сценария после подтверждения пользователем. До подтверждения результат существует только как preview внутри Draft / PendingAction.

Поле	Тип / формат	Обязательность	Описание
calorie_entry_id	uuid / string	обязательно после save	Уникальный идентификатор записи калорий.
event_card_id	uuid / string	обязательно	Связь с EventCard типа meal / food.
health_entry_id	uuid / string / nullable	если создается Health-запись	Связь с Health.
user_id / profile_id	uuid / string	обязательно	Владелец и профиль.
input_type	enum	обязательно	text, photo, text_photo.
source_text	string / nullable	если был текст	Описание еды пользователем.
attachment_id	uuid / string / nullable	если было фото	Связь с загруженным материалом.
meal_type	enum / nullable	желательно	breakfast, lunch, dinner, snack, drink, unknown.
estimated_kcal	number	обязательно после расчёта	Оценка калорийности прихода.
confidence	number / nullable	желательно	Условная уверенность расчёта.
clarifications	json / nullable	если были уточнения	Порция, состав, время приёма пищи.
status	enum	обязательно	preview_shown, confirmed, saved, cancelled, failed.
created_at / confirmed_at	datetime UTC	обязательно после save	Метки создания и подтверждения.

11.4. EmotionTryOnResult

EmotionTryOnResult фиксирует результат примерочной эмоций, если пользователь подтвердил сохранение результата в карточку события.

Поле	Тип / формат	Обязательность	Описание
emotion_try_on_result_id	uuid / string	обязательно после save	Уникальный идентификатор результата примерочной.
event_card_id	uuid / string	обязательно	Связь с EventCard.
user_id / profile_id	uuid / string	обязательно	Владелец и профиль.
emotion_mode	string / enum	обязательно	Выбранный эмоциональный режим, например calm_confidence.
input_text	string / nullable	если был текст	Формулировка пользователя.
attachment_id	uuid / string / nullable	если был материал	Связь с фото / видео / материалом.
result_summary	string	обязательно	Краткое описание результата для пользователя.
result_payload	json / nullable	если есть structured result	Структурированный результат примерочной.
status	enum	обязательно	preview_shown, confirmed, saved, cancelled, failed.
created_at / confirmed_at	datetime UTC	обязательно после save	Метки создания и подтверждения.

11.5. AssistantTopic

AssistantTopic - тема / чат / case внутри Ассистента. В первом релизе не является верхнеуровневым проектом и не объединяет несколько проектов.

Поле	Тип / формат	Обязательность	Описание
assistant_chat_id / topic_id	uuid / string	обязательно	Уникальный идентификатор темы.
user_id / profile_id	uuid / string	обязательно	Владелец и профиль.
title	string	обязательно	Название темы, заданное пользователем или предложенное Ассистентом.
status	enum	обязательно	active, archived, deleted_soft.
last_message_id	uuid / string / nullable	желательно	Последнее сообщение в теме.
created_at / updated_at	datetime UTC	обязательно	Служебные временные метки.
deleted_at	datetime UTC / nullable	если удалена	Метка soft delete.

Удаление AssistantTopic должно выполняться как soft delete / архивирование, если тема связана с EventCard, CalorieEntry, EmotionTryOnResult или Health-записями.

12. JSON-контракты backend-ответов

Настоящий раздел является частью основного ТЗ. JSON-примеры фиксируют целевой формат ответов backend для frontend. Поля могут быть расширены, но не должны терять базовую структуру message / blocks / metadata / status / ids.

12.1. EventCard preview

```
{
  "message": "Я подготовил карточку события. Проверьте данные перед сохранением.",
  "blocks": [
    {
      "type": "event_card_preview",
      "draft_id": "draft_123",
      "event_card": {
        "title": "Сложная встреча",
        "description": "Пользователь переживает перед встречей",

```

```

    "local_date": "2026-05-01",
    "time": "10 00",
    "event_type": "meeting",
    "status": "preview_shown"
  },
  "actions": [
    "confirm",
    "edit",
    "cancel"
  ]
},
"metadata": {
  "scenario_type": "event_creation",
  "source": "assistant_chat_text",
  "intent_type": "event_creation",
  "status": "preview_shown"
}
}

```

12.2. Calorie preview

```
{
  "message": "Я оценил калорийность. Подтвердите или уточните порцию.",
  "blocks": [
    {
      "type": "calorie_result_preview",
      "draft_id": "draft_456",
      "calorie_result": {
        "input_type": "text_photo",
        "estimated_kcal": 420,
        "meal_type": "breakfast",
        "items": [
          {
            "name": "омлет",
            "estimated_kcal": 250
          },
          {
            "name": "хлеб",
            "estimated_kcal": 120
          },
          {
            "name": "кофе",
            "estimated_kcal": 50
          }
        ]
      },
      "status": "preview_shown"
    },
    {
      "actions": [
        "confirm",
        "clarify",
        "cancel"
      ]
    }
  ],
  "metadata": {
    "scenario_type": "calorie_tracking",
    "source": "assistant_chat_text",
  }
}
```

```
    "status": "preview_shown"  
  }  
}
```

12.3. Emotion try-on preview

```
{  
  "message": "Я подготовил результат примерочной эмоций. Можно сохранить его в карточку события.",  
  "blocks": [  
    {  
      "type": "emotion_try_on_preview",  
      "draft_id": "draft_789",  
      "emotion_try_on": {  
        "emotion_mode": "calm_confidence",  
        "result_summary": "Спокойная уверенность перед разговором",  
        "status": "preview_shown"  
      },  
      "actions": [  
        "confirm",  
        "edit",  
        "cancel"  
      ]  
    },  
  ],  
  "metadata": {  
    "scenario_type": "emotion_try_on",  
    "source": "assistant_chat_text",  
    "status": "preview_shown"  
  }  
}
```

12.4. Confirm response

```
{  
  "message": "Готово. Результат сохранён.",  
  "blocks": [  
    {  
      "type": "status_block",  
      "status": "saved"  
    },  
    {  
      "type": "event_card_preview",  
      "event_card_id": "event_123",  
      "actions": [  
        "open",  
        "edit",  
        "archive"  
      ]  
    }  
  ],  
  "metadata": {  
    "draft_id": "draft_123",  
    "event_card_id": "event_123",  
    "calorie_entry_id": null,  
    "emotion_try_on_result_id": null,  
    "status": "saved",  
    "idempotency_key": "confirm_abc"  
  }  
}
```


12.5. Cancel response

```
{
  "message": "Действие отменено. Я ничего не сохранил.",
  "blocks": [
    {
      "type": "status_block",
      "status": "cancelled"
    }
  ],
  "metadata": {
    "draft_id": "draft_123",
    "status": "cancelled"
  }
}
```

13. Ошибки и статусы API

Backend должен возвращать ошибки в едином формате, чтобы frontend мог показывать понятный status_block и не создавать частично сохранённые объекты.

HTTP code	error_code	Когда возникает	Ожидаемое поведение
400	invalid_payload	Некорректный request body, отсутствуют обязательные поля, неверный формат enum.	Frontend показывает ошибку ввода и не переводит сценарий в saved.
403	profile_mismatch	Переданный draft_id, event_card_id или profile_id не принадлежит текущему пользователю / профилю.	Операция блокируется, объект не меняется.
404	draft_not_found	Черновик не найден или был удалён.	Frontend предлагает начать сценарий заново.
409	draft_expired	Истёк срок действия draft / pending action	Frontend показывает статус expired и предлагает создать новый draft.
409	already_confirmed	Повторный confirm для уже подтверждённого draft.	Backend возвращает ранее созданные ids без создания дублей.
422	missing_clarification	Для расчёта или EventCard не хватает данных: дата, порция, режим змощии и т. п.	Backend возвращает quick_replies / уточняющий вопрос.
500	scenario_failed	Внутренняя ошибка сценария или ошибка внешнего AI-сервиса.	Backend не создаёт частично сохранённый объект без статуса failed.

13.1. Единый формат ошибки

```
{
  "error": {
    "code": "missing_clarification",
    "message": "Нужно уточнить порцию блюда.",
    "status": "waiting_clarification",
    "details": {
      "missing_fields": [
        "portion_size"
      ]
    }
  },
  "draft_id": "draft_456"
},
"blocks": [
  {
```

```

    "type": "quick_replies",
    "items": [
      "маленькая порция",
      "средняя порция",
      "большая порция"
    ]
  },
  {
    "type": "status_block",
    "status": "waiting_clarification"
  }
]
}

```

13.2. Статусы сценариев

Статус	Смысл	Можно ли сохранять объект
draft_created	Черновик создан, пользователь ещё не подтвердил.	Нет.
waiting_clarification	Backend ждёт уточняющие данные.	Нет.
preview_shown	Preview показан пользователю.	Нет, пока нет confirm.
confirmed	Пользователь подтвердил действие.	Да, backend может выполнить save.
saved	Объект сохранён.	Уже сохранён.
cancelled	Пользователь отменил действие.	Нет.
expired	Черновик устарел.	Нет.
failed	Сценарий завершился ошибкой.	Нет без отдельного восстановления.

14. Правила идиempотентности и защиты от дублей

Повторный confirm не должен создавать две EventCard, две CalorieEntry или два EmotionTryOnResult. Backend должен обеспечивать идиempотентность операций подтверждения.

14.1. Общие правила

- Каждая операция confirm должна принимать или формировать idempotency_key
- Для одного draft_id допускается только один успешный переход в confirmed / saved.
- Если frontend повторно отправил confirm с тем же idempotency_key, backend возвращает ранее созданный результат.
- Если confirm пришёл повторно с другим idempotency_key, но draft уже saved, backend не создаёт дубль и возвращает already_confirmed / ранее созданные ids.
- Создание EventCard, CalorieEntry и EmotionTryOnResult должно выполняться в транзакции или в эквивалентном атомарном блоке.
- При сбое после частичного сохранения backend должен уметь вернуть консистентный статус или выполнить rollback.

14.2. Минимальная логика confirm

1. Проверить пользователя и профиль.
2. Найти draft / pending_action.
3. Проверить статус draft: draft_created, preview_shown или waiting_clarification могут продолжаться; cancelled / expired / failed не сохраняются.
4. Проверить idempotency_key.
5. Если результат уже сохранён, вернуть существующие event_card_id / calorie_entry_id / emotion_try_on_result_id.
6. Если результата ещё нет, выполнить сохранение в транзакции.
7. Обновить статусы draft и связанных объектов.
8. Вернуть confirm response с итоговыми идентификаторами.

14.3. Уникальные ограничения

Объект	Рекомендуемое ограничение	Зачем нужно
Draft / PendingAction	unique(draft_id, user_id, profile_id)	Исключить доступ к чужому draft
Confirm operation	unique(draft_id, idempotency_key)	Исключить повторную обработку одного confirm.
EventCard from draft	unique(draft_id) where status in confirmed/saved	Не создать две карточки из одного draft.
CalorieEntry from draft	unique(draft_id) for calorie_tracking	Не создать две калорийные записи из одного draft.
EmotionTryOnResult from draft	unique(draft_id) for emotion_try_on	Не создать два результата примерочной из одного draft.

15. Техническая готовность к СУК

Полноценный СУК по этим событиям остаётся отдельным ТЗ SE-01-TS11. В рамках TS13 backend должен обеспечить техническую готовность к последующей интеграции: идентификаторы объектов, источники создания, статусы сценариев, временные метки и связи между сущностями.

15.1. Поля, которые должны быть доступны в рамках TS13

- user_id;
- profile_id;
- assistant_chat_id / topic_id;
- assistant_message_id;
- scenario_type;
- scenario_source;
- intent_type;
- draft_id;
- event_card_id;
- health_entry_id;
- calorie_entry_id;
- emotion_try_on_result_id;
- source_screen;
- source_action;
- status;
- created_at / updated_at / confirmed_at / cancelled_at;
- local_date;
- timezone;
- language;
- platform;

- app_version.

15.2. События для последующего ТЗ по СУК

- assistant_scenario_started;
- assistant_intent_detected;
- assistant_question_asked;
- assistant_action_shown;
- assistant_action_clicked;
- assistant_draft_created;
- assistant_preview_shown;
- assistant_draft_confirmed;
- assistant_draft_cancelled;
- event_card_created;
- event_card_created_from_assistant;
- calorie_scenario_started;
- calorie_result_preview_shown;
- calorie_result_saved;
- emotion_try_on_started;
- emotion_try_on_preview_shown;
- emotion_try_on_result_saved;
- assistant_scenario_completed;
- assistant_scenario_abandoned.

15.3. Не входит в TS13 по СУК

- витрины СУК;
- управленческие отчёты;
- агрегаты и регулярные расчёты;
- аналитические панели;
- расширенная матрица продуктовых метрик;
- бюджетная и рекламная аналитика по этим событиям.

16. Требования к безопасности и сохранности данных

- Ассистент не должен сохранять EventCard, calorie result или emotion try-on result без подтверждения пользователя;
- удаление темы / чата не должно удалять сохранённые структурированные объекты продукта;
- для операций сохранения, редактирования и удаления должны применяться проверки user_id / profile_id;
- вложения должны быть связаны с конкретным сообщением и сценарием;
- при ошибке сценария пользователь должен получить понятный status_block, а backend не должен создавать частично сохранённые объекты без явного статуса;
- все идентификаторы, возвращаемые frontend, должны проверяться на принадлежность пользователю и профилю.

17. Критичные сценарии сквозной приёмки

Сценарий	Шаги проверки	Критерий успеха
Событие из кнопки	Ассистент -> кнопка Создать событие -> уточнения -> draft -> preview -> confirm -> EventCard -> Calendar	Карточка сохранена и отображается в календарном контуре.
Событие из текста	Пользователь пишет Завтра сложная встреча -> intent -> уточнения -> preview -> confirm	EventCard создана из чата, связана с темой и сообщением.
Калории из чата	Пользователь пишет описание еды или прикладывает фото -> расчет -> preview -> confirm	Создана EventCard типа meal, данные отображаются в Health.
Примерочная эмоций	Пользователь создаёт / выбирает событие -> примерочная -> preview -> confirm	Результат примерочной сохранён в EventCard
Темы	Создать тему, продолжить, переименовать, удалить	Тема управляется как чат / case, сохранённые EventCard и Health не удаляются.
Техническая готовность	Проверить наличие ids, source, status,	Backend отдаёт необходимые

Сценарий	Шаги проверки	Критерий успеха
к СУК	timestamps, связей	поля для будущего ТЗ по СУК.
Идемпотентность confirm	Дважды отправить confirm по одному draft_id / idempotency_key	Вторая операция возвращает ранее созданные ids и не создаёт дубль.

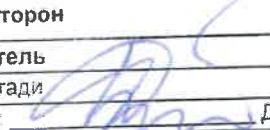
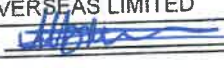
18. Ограничения scope первого релиза

- верхнеуровневые проекты, объединяющие несколько тем / чатов.
- полноценный СУК, витрины, отчёты и агрегаты.
- внешние food database и barcode scanning.
- сложные повторяющиеся события и расширенный lifecycle событий.
- расширенная нутриентная аналитика и рекомендации по рациону;
- отдельный продуктовый раздел первого уровня для примерочной эмоций или калорий.
- отдельные frontend/mobile работы, кроме согласования контрактов и поддержки интеграции.

19. Результат работ

- гибридную модель Ассистента с запуском сценариев из кнопок и из чата;
- понимание намерений пользователя по ключевым сценариям;
- уточняющие вопросы и structured actions для frontend;
- draft / preview / confirm flow;
- создание и сохранение EventCard;
- сохранение калорийного результата в EventCard и Health;
- сохранение результата примерочной в EventCard;
- темы как отдельные чаты / cases;
- детальную модель данных для EventCard, Draft / PendingAction, CalorieEntry, EmotionTryOnResult и AssistantTopic;
- JSON-контракты backend-ответов для preview, confirm и cancel;
- единый формат ошибок и статусов API;
- идемпотентность confirm и защиту от дублей;
- техническую готовность к последующему ТЗ по СУК.

Подписи сторон

Исполнитель	Заказчик
ООО Вистади	ONERY OVERSEAS LIMITED
Директор:  Дильдин В.С.	Директор:  Булициду А.
М.П.	М.П.