User Stories для AI-агента проверки User Stories по INVEST

Основная User Story

Как системный аналитик/разработчик продукта, Я хочу отправлять текст User Story в Telegram-бота, Чтобы мгновенно получать детальный анализ соответствия критериям INVEST с конкретными рекомендациями по улучшению.

Дополнительные User Stories

US-1: Базовый анализ

Как продукт-оунер, Я хочу быстро проверять новые User Stories на соответствие INVEST, Чтобы обеспечивать качество бэклога продукта перед планированием спринта.

US-2: Обучение на примерах

Как начинающий аналитик, Я хочу видеть примеры правильно написанных User Stories с пояснениями, Чтобы обучаться лучшим практикам формулирования требований.

US-3: История анализа

Как член команды разработки, Я хочу просматривать историю предыдущих проверок, Чтобы отслеживать прогресс в улучшении качества User Stories.

US-4: Сравнительный анализ

Как тимлид, Я хочу сравнивать разные версии одной User Story, Чтобы оценивать эффективность рефакторинга требований.

US-5: Экспорт результатов

Как владелец продукта, Я хочу экспортировать отчеты анализа в удобном формате, Чтобы делиться результатами с стейкхолдерами.

Детализация основной User Story

Критерии приемки (Acceptance Criteria):

AC-1: Успешный запуск бота

Дано: Пользователь открывает Telegram-бота

Когда: Пользователь отправляет команду /start

Тогда: Бот отображает приветственное сообщение с инструкциями и кнопками основных действий

AC-2: Анализ User Story

Дано: Пользователь находится в основном меню бота

Когда: Пользователь отправляет текст User Story для анализа

Тогда: Бот возвращает структурированный отчет с:

Общей оценкой (Хорошо/Средне/Плохо)

Балльной оценкой 0-100

Детализацией по каждому критерию INVEST

Конкретными нарушениями с фрагментами текста

Рекомендациями по исправлению

AC-3: Невалидный ввод

Дано: Пользователь отправляет сообщение боту

Когда: Сообщение не содержит текст User Story или слишком короткое

Тогда: Бот возвращает понятное сообщение об ошибке с примером корректного формата

AC-4: Использование эталонных примеров

Дано: Бот анализирует User Story

Когда: Обнаружены нарушения критериев INVEST

Тогда: Бот предоставляет ссылки на похожие эталонные User Stories из базы данных

Use Cases для Telegram-бота анализа User Stories

UC-01: Анализ User Story

Заголовок	Анализ User Story по критериям INVEST
Акторы	системный аналитик, продукт-оунер
Предусловие	- Пользователь запустил бота командой /start
	- Бот доступен и функционирует
Ограничения	- Максимальная длина текста: 500 символов
	- Проверка только по критериям INVEST
	- Максимум 10 запросов в час на пользователя
Триггер	Пользователь отправляет текст User Story боту
Основной сценарий	1. Система проверяет валидность формата User Story
	2. Система отправляет запрос к LLM для анализа INVEST
	3. Для каждого нарушения система определяет:
	- Тип нарушения (критерий INVEST)
	- Фрагмент текста с ошибкой
	- Рекомендацию по исправлению
	4. Система обогащает ответ примерами из базы данных
	5. Система сохраняет результат проверки в истории
	6. Бот отображает структурированный отчет
	Критерии успеха: отчет сгенерирован за ≤30 сек, нарушения соответствуют INVEST
Альтернативный сценарий	1a. Если текст пустой или слишком короткий, система показывает ошибку: "Введите текст User Story для анализа"

Исключительный сценарий	2a. Если LLM недоступен, система возвращает ошибку: "Сервис временно не работает. Попробуйте позже"

UC-02: Просмотр истории проверок

Заголовок	Доступ к истории предыдущих анализов
Акторы	системный аналитик, продукт-оунер
Предусловие	- Пользователь авторизован в боте
	- Ранее выполнена хотя бы 1 проверка
Ограничения	- Отображаются только последние 20 проверок
	- Данные хранятся 90 дней
Триггер	Пользователь нажимает кнопку "История анализов"
Основной сценарий	1. Система загружает список проверок из базы данных в виде:
	- Дата и время анализа
	- Первые 50 символов текста User Story
	- Общая оценка (Хорошо/Средне/Плохо)
	- Количество нарушений
	2. При выборе записи система отображает полный отчет из истории
	Критерии успеха: история загружена за ≤3 сек
Альтернативный сценарий	2a. Пользователь выбирает конкретную запись для просмотра детального отчета с нарушениями и рекомендациями
Исключительный сценарий	1a. Если в истории нет проверок, система выводит: "У вас пока нет анализов. Проанализируйте первую User Story!"

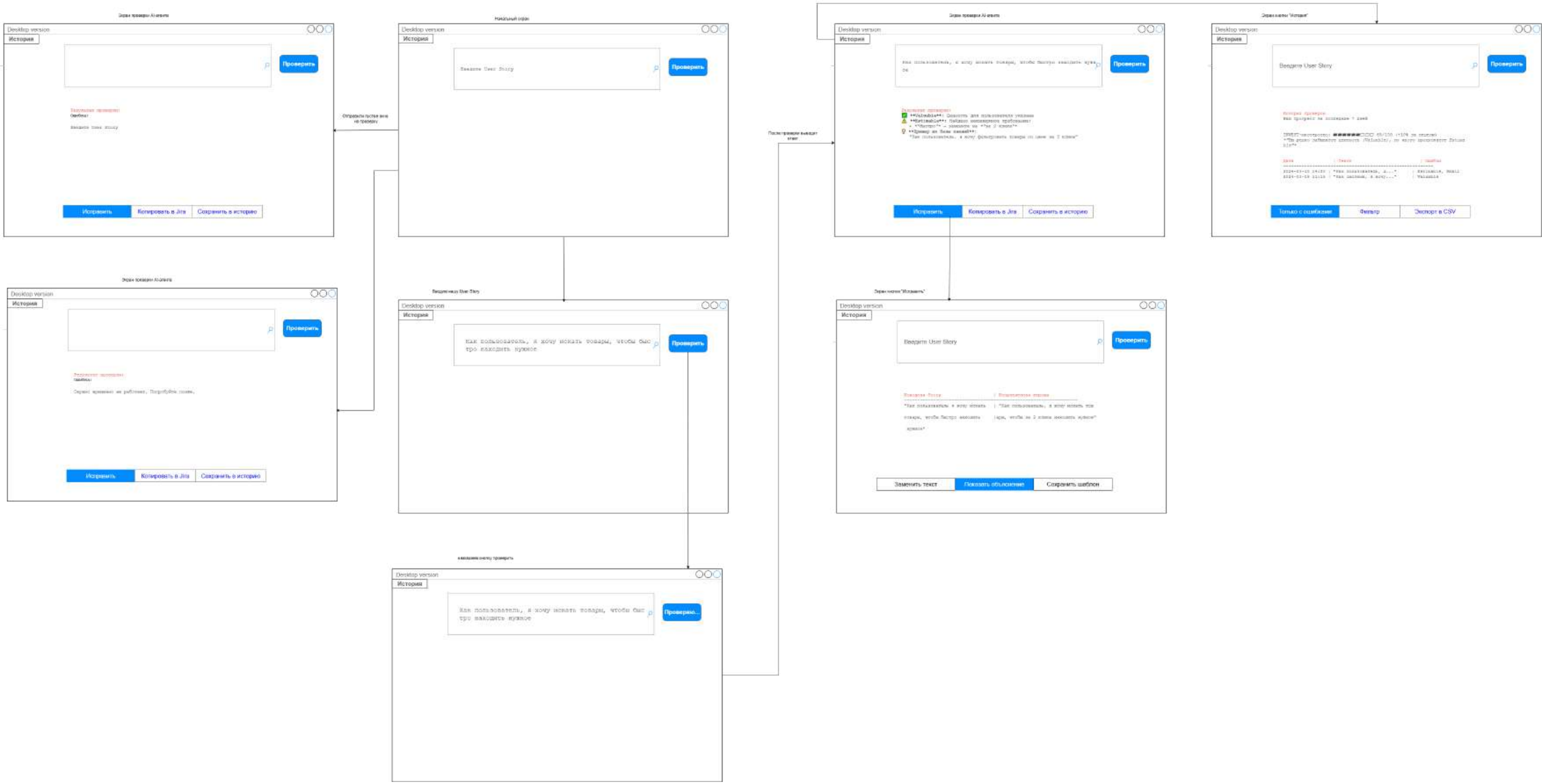
UC-03: Просмотр примеров User Stories

Заголовок	Получение эталонных примеров User Stories
Акторы	системный аналитик, начинающий аналитик
Предусловие	- Пользователь запустил бота командой /start
Ограничения	- Показывается до 5 примеров за раз
	- Примеры разделены по категориям качества
Триггер	Пользователь нажимает кнопку "Примеры User Stories"
Основной сценарий	1. Система показывает категории примеров:
	- Отличные примеры (90-100 баллов)
	- Хорошие примеры (70-89 баллов)
	- Примеры с типичными ошибками
	2. Пользователь выбирает категорию
	3. Система отображает 3-5 примеров с:
	- Текстом User Story
	- Оценкой анализа
	- Ключевыми достоинствами/недостатками
	Критерии успеха: примеры загружены за ≤2 сек
Альтернативный сценарий	3а. Пользователь может запросить анализ конкретного примера для обучения
Исключительный сценарий	2а. Если база примеров недоступна, система использует встроенные базовые примеры

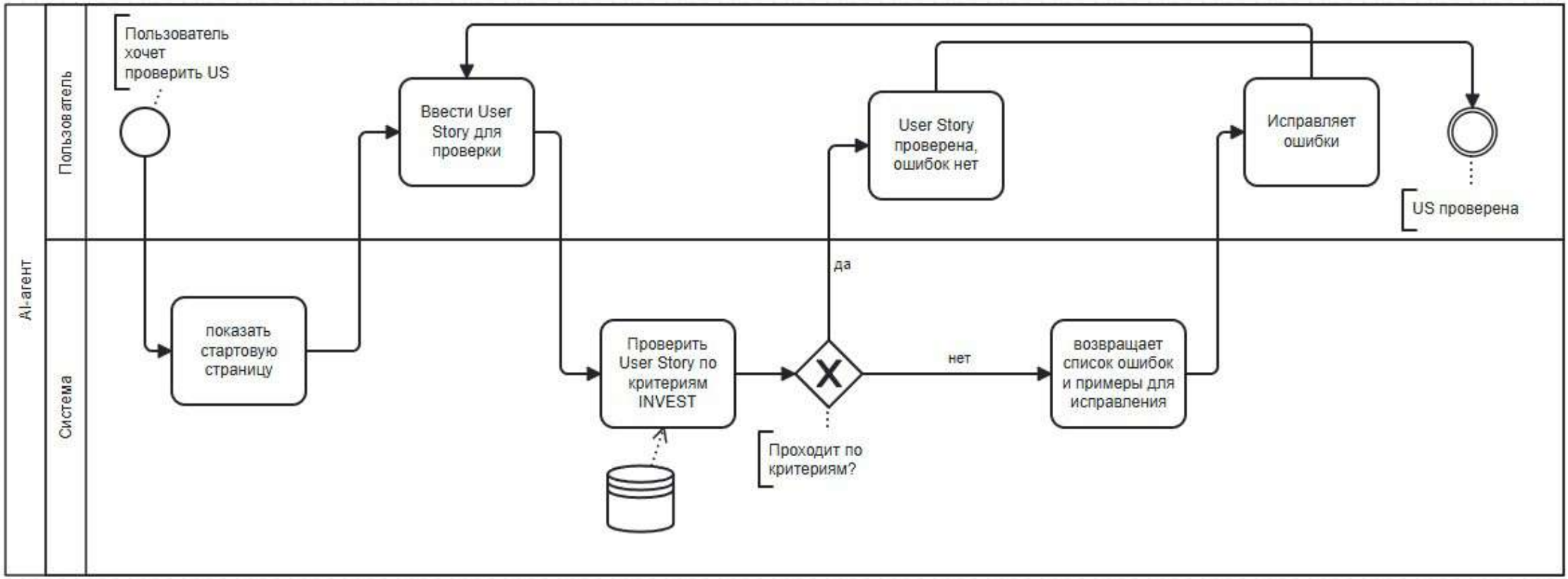
UC-04: Поиск похожих User Stories

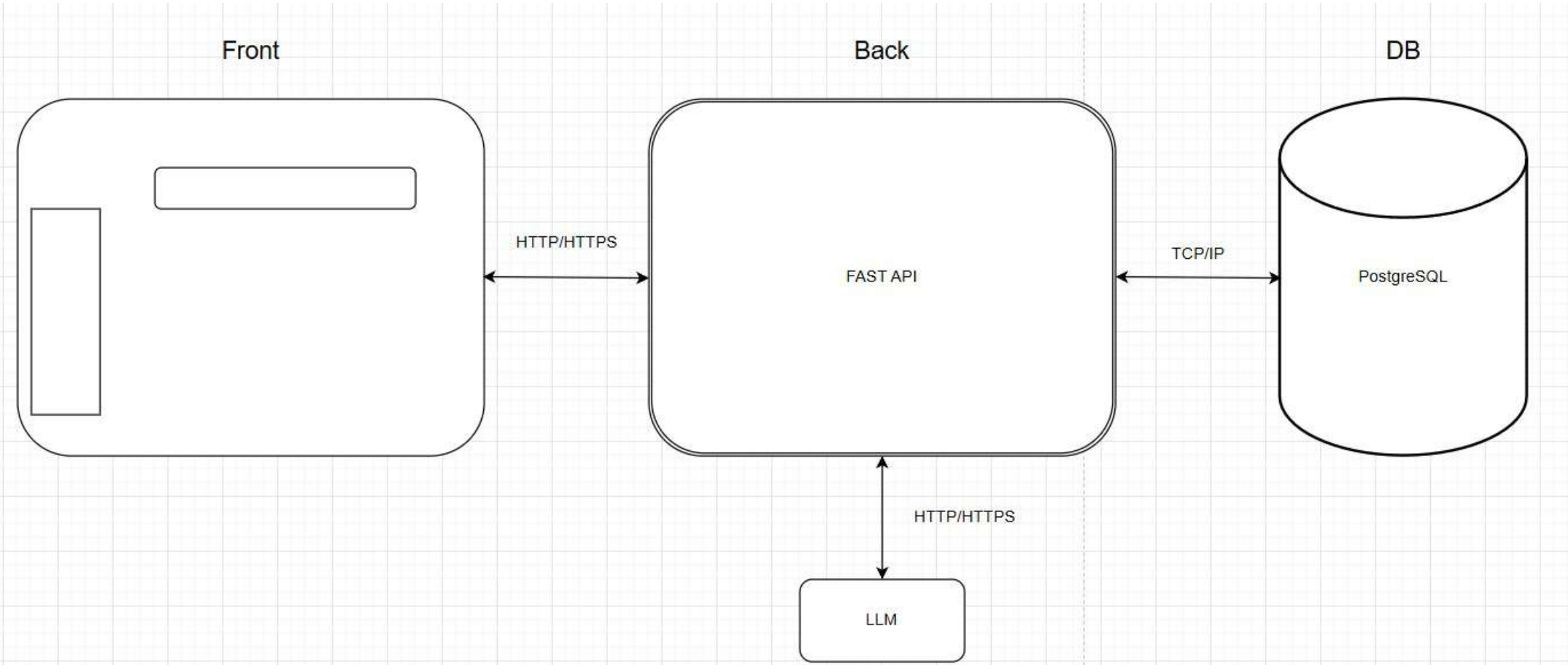
Заголовок	Поиск аналогичных User Stories в базе
Акторы	системный аналитик, продукт-оунер
Предусловие	- Выполнен анализ User Story

Предусловие	- Найдены нарушения критериев
Ограничения	- Показывается до 3 похожих примеров
	- Порог схожести: 75%
Триггер	Система автоматически после анализа User Story
Основной сценарий	1. Система ищет в базе данных User Stories с похожей тематикой
	2. Для найденных примеров показывает:
	- Текст эталонной User Story
	- Коэффициент схожести
	- Результат анализа эталона
	3. Пользователь может изучить примеры для улучшения своей User Story
	Критерии успеха: похожие примеры найдены за ≤5 сек
Альтернативный сценарий	1а. Пользователь может самостоятельно запросить поиск похожих Stories через отдельную команду
Исключительный сценарий	1а. Если похожих примеров не найдено, система сообщает: "Похожих примеров не найдено. Создайте первый!"



Начальный экран
Desktop version
История
Введите User Story
Проверить
Отправили пустое окно на проверку
Desktop version
История
Проверить
Исправить
Копировать в Jira
Сохранить в историю
Desktop version
История
Проверить
Исправить
Копировать в Jira
Сохранить в историю
Вводим нашу User Story
Desktop version
История
Как пользователь, я хочу искать товары, чтобы быстро находить нужное
Проверить
Desktop version
История
Как пользователь, я хочу искать товары, чтобы быстро находить нужное
Проверяю...
После проверки выводит ответ
Desktop version
История
Проверить
Исправить
Копировать в Jira
Сохранить в историю
Desktop version
История
Введите User Story
Проверить
Только с ошибками
Фильтр
Экспорт в CSV
Desktop version
История
Введите User Story
Проверить
Заменить текст
Показать объяснение
Сохранить шаблон


AI-агент
Пользователь
Система
Пользователь хочет проверить US
Ввести User Story для проверки
User Story проверена, ошибок нет
Исправляет ошибки
US проверена
показать стартовую страницу
Проверить User Story по критериям INVEST
да
нет
возвращает список ошибок и примеры для исправления
Проходит по критериям?


Front
Back
DB
HTTP/HTTPS
FAST API
TCP/IP
PostgreSQL
HTTP/HTTPS
LLM

Модель данных AI-агента

users

Атрибут	Формат	Описание	Доп. информация
user_id	UUID	Уникальный идентификатор пользователя	PK
telegram_id	BIGINT	ID пользователя в Telegram	Уникальный, NOT NULL
username	VARCHAR(100)	Имя пользователя в Telegram	Опционально
first_name	VARCHAR(100)	Имя пользователя	
last_name	VARCHAR(100)	Фамилия пользователя	
created_at	TIMESTAMP	Дата регистрации	DEFAULT CURRENT_TIMESTAMP
last_activity	TIMESTAMP	Последняя активность	DEFAULT CURRENT_TIMESTAMP
is_active	BOOLEAN	Статус активности	DEFAULT true

user_stories

Атрибут	Формат	Описание	Доп. информация
story_id	UUID	Уникальный идентификатор User Story	PK
user_id	UUID	Ссылка на пользователя	FK → users.user_id
original_text	TEXT	Текст анализируемой User Story	NOT NULL
created_at	TIMESTAMP	Дата анализа	DEFAULT CURRENT_TIMESTAMP
overall_score	INTEGER	Общая оценка качества	0-100
overall_grade	VARCHAR(10)	Общая оценка текстом	'excellent', 'good', 'poor'

analysis_summary	TEXT	Сводка анализа от LLM	

violations

Атрибут	Формат	Описание	Доп. информация
violation_id	UUID	Уникальный идентификатор нарушения	PK
story_id	UUID	Ссылка на User Story	FK → user_stories.story_id
criteria	VARCHAR(1)	Критерий INVEST	I, N, V, E, S, T
fragment_text	TEXT	Проблемный фрагмент текста	
suggestion	TEXT	Рекомендация по исправлению	NOT NULL
severity	VARCHAR(10)	Уровень серьезности	low, medium, high

reference_examples

Атрибут	Формат	Описание	Доп. информация
example_id	UUID	Уникальный ID эталонного примера	PK
title	VARCHAR(200)	Название примера	
example_text	TEXT	Текст эталонной User Story	NOT NULL
category	VARCHAR(20)	Категория качества	'excellent', 'good', 'with_errors'
domain	VARCHAR(50)	Предметная область	'ecommerce', 'saas', 'mobile'
score	INTEGER	Оценка качества	0-100
invest_scores	JSON	Оценки по критериям INVEST	

explanation	TEXT	Пояснение к примеру	
created_at	TIMESTAMP	Дата добавления	DEFAULT CURRENT_TIMESTAMP

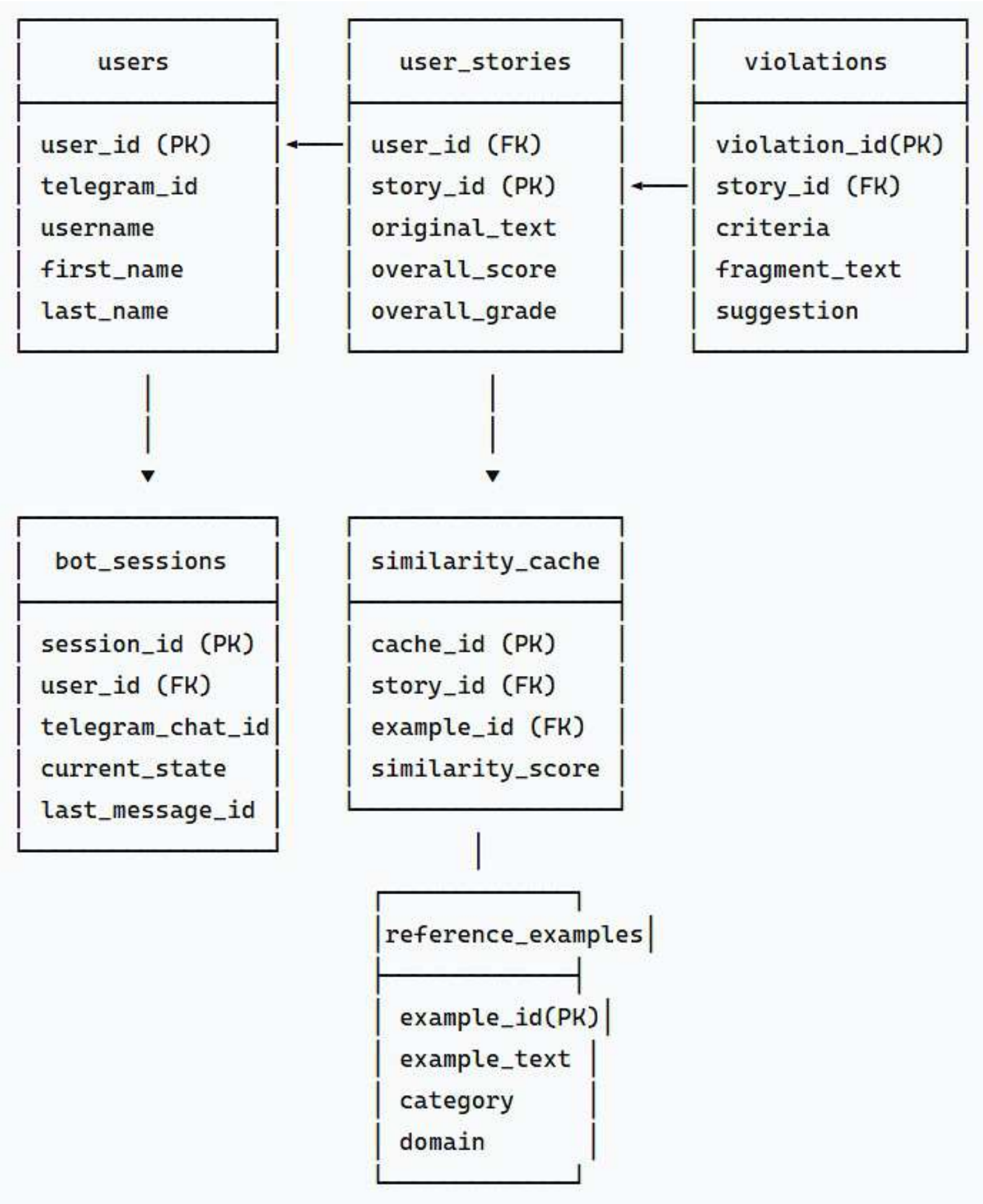
bot_sessions

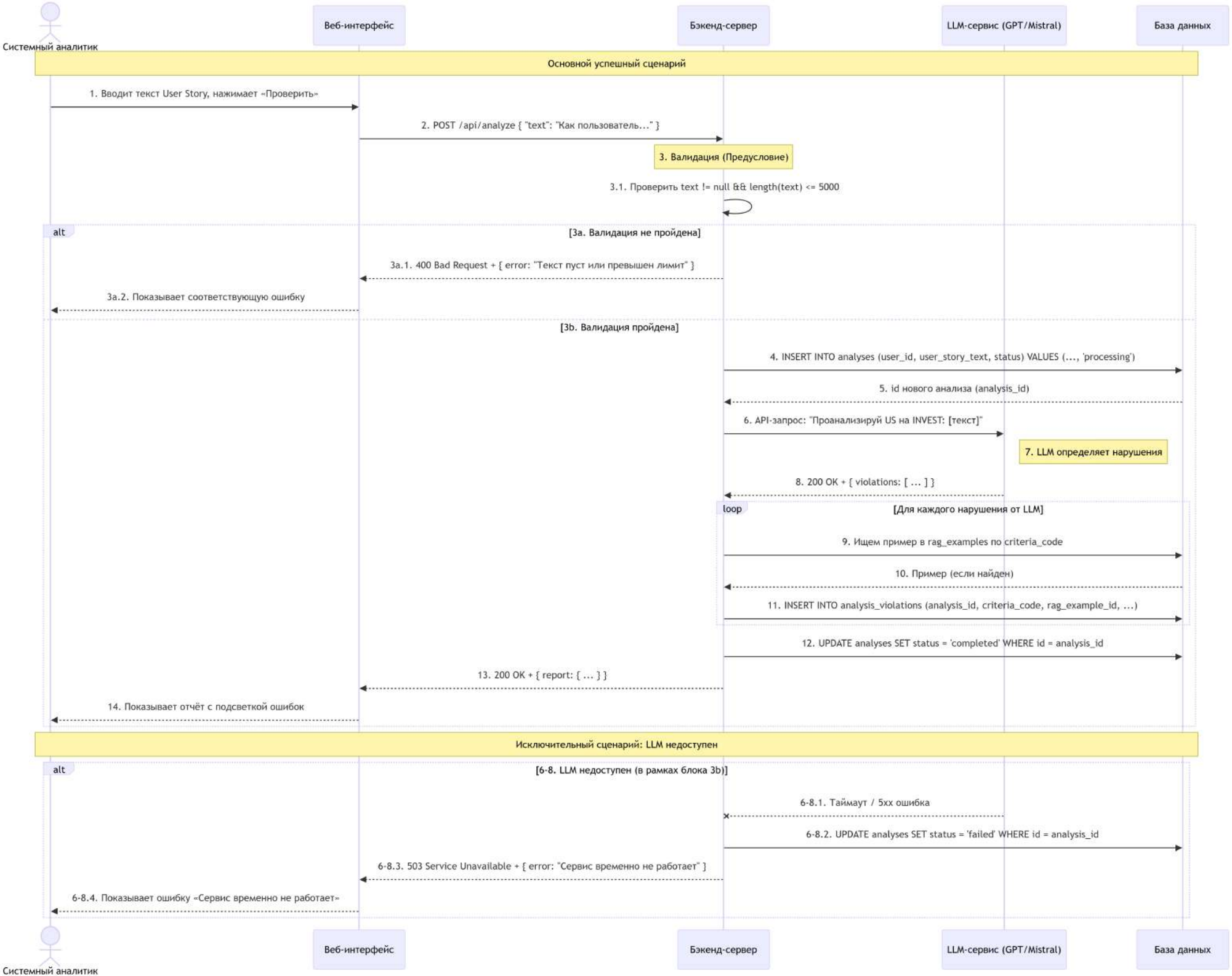
Атрибут	Формат	Описание	Доп. информация
session_id	UUID	Уникальный идентификатор сессии	PK
user_id	UUID	Ссылка на пользователя	FK → users.user_id
telegram_chat_id	BIGINT	ID чата в Telegram	NOT NULL
current_state	VARCHAR(50)	Текущее состояние бота	'start', 'waiting_story', 'showing_results'
last_message_id	INTEGER	ID последнего сообщения бота	Для редактирования
context_data	JSONB	Данные контекста диалога	
created_at	TIMESTAMP	Дата создания сессии	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	Дата обновления	DEFAULT CURRENT_TIMESTAMP

similarity_cache

Атрибут	Формат	Описание	Доп. информация
cache_id	UUID	Уникальный идентификатор кэша	PK
story_id	UUID	Ссылка на User Story	FK → user_stories.story_id
example_id	UUID	Ссылка на эталонный пример	FK → reference_examples.example_id

similarity_score	DECIMAL(4,3)	Коэффициент схожести	0.000 - 1.000
match_reason	TEXT	Причина совпадения	
created_at	TIMESTAMP	Дата создания записи	DEFAULT CURRENT_TIMESTAMP


users
user_id (PK)
telegram_id
username
first_name
last_name
user_stories
user_id (FK)
story_id (PK)
original_text
overall_score
overall_grade
violations
violation_id(PK)
story_id (FK)
criteria
fragment_text
suggestion
bot_sessions
session_id (PK)
user_id (FK)
telegram_chat_id
current_state
last_message_id
similarity_cache
cache_id (PK)
story_id (FK)
example_id (FK)
similarity_score
reference_examples
example_id(PK)
example_text
category
domain


Системный аналитик
Веб-интерфейс
Бэкенд-сервер
LLM-сервис (GPT/Mistral)
База данных
Основной успешный сценарий
1. Вводит текст User Story, нажимает «Проверить»
2. POST /api/analyze { "text": "Как пользователь..." }
3. Валидация (Предусловие)
3.1. Проверить text != null && length(text) <= 5000
alt
[3a. Валидация не пройдена]
3a.1. 400 Bad Request + { error: "Текст пуст или превышен лимит" }
3a.2. Показывает соответствующую ошибку
[3b. Валидация пройдена]
4. INSERT INTO analyses (user_id, user_story_text, status) VALUES (..., 'processing')
5. id нового анализа (analysis_id)
6. API-запрос: "Проанализируй US на INVEST: [текст]"
7. LLM определяет нарушения
8. 200 OK + { violations: [...] }
loop
[Для каждого нарушения от LLM]
9. Ищем пример в rag_examples по criteria_code
10. Пример (если найден)
11. INSERT INTO analysis_violations (analysis_id, criteria_code, rag_example_id, ...)
12. UPDATE analyses SET status = 'completed' WHERE id = analysis_id
13. 200 OK + { report: { ... } }
14. Показывает отчёт с подсветкой ошибок
Исключительный сценарий: LLM недоступен
alt
[6-8. LLM недоступен (в рамках блока 3b)]
6-8.1. Таймаут / 5xx ошибка
6-8.2. UPDATE analyses SET status = 'failed' WHERE id = analysis_id
6-8.3. 503 Service Unavailable + { error: "Сервис временно не работает" }
6-8.4. Показывает ошибку «Сервис временно не работает»
Системный аналитик
Веб-интерфейс
Бэкенд-сервер
LLM-сервис (GPT/Mistral)
База данных

Пояснения к Sequence Diagram

Основной успешный сценарий анализа

Триггер: Пользователь отправляет текст User Story в Telegram-бота.

Шаг 1-2: Бот принимает сообщение и передает его на бэкенд через API-запрос. Запрос содержит текст User Story и идентификатор пользователя.

Шаг 3: Критически важный этап валидации текста. Система проверяет наличие текста, его длину (не менее 10 и не более 500 символов), а также базовую структуру на наличие ключевых фраз. Эта проверка защищает систему от некорректных данных и предотвращает ненужные вызовы внешних сервисов.

Шаг 4-5: Если текст валиден, система сохраняет факт начала анализа в базу данных. Запись создается сразу со статусом "processing", что позволяет отслеживать состояние операции. Система получает уникальный идентификатор записи для связи всех последующих данных.

Шаг 6-7: Вызов внешнего LLM-сервиса — ключевое действие системы. Это основная бизнес-логика приложения. LLM анализирует текст по критериям INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable) и определяет нарушения.

Шаг 8: Получение результата от LLM в структурированном виде. Ответ содержит массив нарушений с указанием критерия, проблемного фрагмента текста и рекомендаций по исправлению, а также общую оценку качества.

Шаг 9: Для каждого выявленного нарушения система ищет подходящие эталонные примеры из базы данных. Это помогает пользователю лучше понять природу ошибки и способы ее исправления.

Шаг 10-11: Сохранение результатов анализа в базу данных. Каждое нарушение фиксируется в отдельной записи с привязкой к основному анализу. Это позволяет в дальнейшем строить статистику и аналитику.

Шаг 12: Фиксация успешного завершения операции путем обновления статуса анализа на "completed". Это важный шаг для мониторинга состояния системы.

Шаг 13-14: Возврат сформированного отчета на фронтенд и показ финального результата пользователю. Отчет включает общую оценку, детализацию нарушений, рекомендации и обучающие примеры.

Альтернативный сценарий (ошибки валидации)

Шаг 3a: Единый блок обработки любой ошибки валидации. Обрабатывается до любых обращений к базе данных и внешним сервисам, что предотвращает лишнюю работу и обеспечивает эффективность системы. Система мгновенно возвращает понятную ошибку пользователю с пояснением требований к формату ввода.

Исключительный сценарий (ошибки LLM)

Шаг 6-8: Обработка ошибки внешней зависимости (LLM). Важно, что система не просто прекращает работу, но выполняет следующие действия:

6-8.1: Фиксирует факт ошибки (таймаут или код ответа 5xx) в системных логах.

6-8.2: Меняет статус проверки на "failed" в базе данных, что позволяет в дальнейшем анализировать статистику сбоев и надежность системы.

6-8.3: Корректно возвращает понятную ошибку на уровень API, сохраняя структуру ответа.

6-8.4: Показывает пользователю информативное сообщение об ошибке без технических деталей, предлагая повторить попытку позже.

Дополнительные сценарии

Просмотр истории анализов: Пользователь запрашивает историю предыдущих проверок. Система загружает из базы данных последние 20 анализов с краткой информацией (дата, фрагмент текста, оценка) и отображает в удобном формате. Если история отсутствует, система показывает соответствующее сообщение.

Просмотр обучающих примеров: Пользователь запрашивает эталонные примеры User Stories. Система сначала показывает доступные категории примеров (отличные, хорошие, с типичными ошибками), затем после выбора категории загружает 3-5 конкретных примеров с пояснениями и анализом.

Критические точки процесса

Валидация на раннем этапе предотвращает нагрузку на дорогостоящие ресурсы (LLM)

Статусная модель (processing → completed/failed) обеспечивает отслеживаемость операций

Изоляция ошибок LLM не прерывает работу всего приложения

Поиск похожих примеров происходит после анализа, что обеспечивает релевантность рекомендаций

Лимитирование истории (20 последних записей) защищает от перегрузки интерфейса

REST API для AI-агента анализа User Stories

GET /ping

Проверка доступности сервиса

Параметр	Тип	Описание	Обязательность
-	-	-	-

Response 200

Параметр	Тип	Описание	Обязательность
ok	boolean	Признак доступности сервиса	да

POST /analyze

Анализ User Story по критериям INVEST

Request

Параметр	Тип	Описание	Обязательность
text	string	Текст User Story для анализа	да
user_id	integer	ID пользователя Telegram	да
chat_id	integer	ID чата Telegram	да

Response 200

Параметр	Тип	Описание	Обязательность
analysis_id	string	UUID анализа	да

overall_verdict	string	Общая оценка: "excellent", "good", "poor"	да
overall_score	integer	Оценка 0-100	да
violations	array[object]	Список нарушений	да
violations[].criteria	string	Критерий: "I", "N", "V", "E", "S", "T"	да
violations[].fragment	string	Проблемный фрагмент текста	да
violations[].suggestion	string	Рекомендация по исправлению	да
violations[].severity	string	Уровень: "low", "medium", "high"	да
similar_examples	array[object]	Похожие эталонные примеры	да
similar_examples[].example_id	string	ID примера	да
similar_examples[].text	string	Текст примера	да
similar_examples[].score	integer	Оценка примера	да
analysis_time	number	Время анализа в секундах	да

Response 400

Параметр	Тип	Описание	Обязательность
detail	string	"Текст User Story не предоставлен"	да

Response 422

Параметр	Тип	Описание	Обязательность
detail	array[object]	Детали ошибки валидации	да

Response 500

Параметр	Тип	Описание	Обязательность
detail	string	"Внутренняя ошибка сервера"	да

Response 503

Параметр	Тип	Описание	Обязательность
detail	string	"Сервис временно недоступен"	да

GET /history

Получение истории анализов пользователя

Request

Параметр	Тип	Описание	Обязательность
user_id	integer	ID пользователя Telegram	да
limit	integer	Лимит записей (по умолчанию 20)	нет

Response 200

Параметр	Тип	Описание	Обязательность

analyses	array[object]	Список анализов	да
analyses[].analysis_id	string	UUID анализа	да
analyses[].text_preview	string	Первые 50 символов текста	да
analyses[].overall_score	integer	Оценка 0-100	да
analyses[].overall_verdict	string	Оценка текстом	да
analyses[].violations_count	integer	Количество нарушений	да
analyses[].created_at	string	Дата создания (ISO)	да

Response 404

Параметр	Тип	Описание	Обязательность
detail	string	"История не найдена"	да

GET /examples

Получение эталонных примеров

Request

Параметр	Тип	Описание	Обязательность
category	string	Категория: "excellent", "good", "with_errors"	нет
domain	string	Домен: "ecommerce", "saas", "mobile"	нет
limit	integer	Лимит (по умолчанию 5)	нет

Response 200

Параметр	Тип	Описание	Обязательность
examples	array[object]	Список примеров	да
examples[].example_id	string	UUID примера	да
examples[].title	string	Название примера	да
examples[].text	string	Текст User Story	да
examples[].category	string	Категория качества	да
examples[].domain	string	Предметная область	да
examples[].score	integer	Оценка 0-100	да
examples[].invest_scores	object	Оценки по INVEST	да
examples[].explanation	string	Пояснение	да

GET /examples/categories

Получение списка категорий примеров

Request

Параметр	Тип	Описание	Обязательность
-	-	-	-

Response 200

Параметр	Тип	Описание	Обязательность
categories	array[string]	Список категорий	да

```yaml
openapi: 3.0.3
info:
  title: AI Agent for User Story Analysis API
  description: API для анализа User Story по критериям INVEST с использованием AI
  version: 1.0.0
  contact:
    name: INVEST_bot


servers:
  - url: https://api.telegram.org/bot{BOT_TOKEN}
    description: Telegram Bot API

tags:
  - name: Health
    description: Проверка работоспособности сервиса
  - name: Analysis
    description: Анализ User Story по критериям INVEST
  - name: Database
    description: Работа с базой данных примеров

paths:
```

```
/ping:
  get:
    tags:
      - Health
    summary: Проверка доступности сервиса
    description: Проверяет, что сервис работает и доступен
    security: []  # Без аутентификации
    responses:
      '200':
        description: Сервис доступен
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/HealthResponse'
      '500':
        description: Внутренняя ошибка сервера
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'

/analyze:
```

```
post:
  tags:
    - Analysis
  summary: Анализ User Story по критериям INVEST
  description: Анализирует User Story с использованием AI и возвращает оценку по критериям INVEST
  security:
    - ApiKeyAuth: []  # Требует API-ключ
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/AnalysisRequest'
  responses:
    '200':
      description: Успешный анализ User Story
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AnalysisResponse'
    '400':
      description: Неверный запрос
```

```
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/ErrorResponse'
'422':
  description: Ошибка валидации данных
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/ValidationError'
'500':
  description: Внутренняя ошибка сервера
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/ErrorResponse'
'503':
  description: Сервис временно недоступен
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/ErrorResponse'
```

```
/examples:
  get:
    tags:
      - Database
    summary: Получить примеры User Story из базы данных
    description: Возвращает список примеров User Story с их анализом
    security:
      - ApiKeyAuth: []  # Требует API-ключ
    parameters:
      - name: limit
        in: query
        schema:
          type: integer
          default: 10
        description: Количество возвращаемых примеров
      - name: offset
        in: query
        schema:
          type: integer
          default: 0
        description: Смещение для пагинации
```

```
    responses:
      '200':
        description: Список примеров успешно получен
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ExamplesResponse'
      '500':
        description: Внутренняя ошибка сервера
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'

/examples/search:
  get:
    tags:
      - Database
    summary: Поиск похожих User Story в базе данных
    description: Ищет похожие User Story в базе данных по тексту запроса
    security:
      - ApiKeyAuth: []  # Требует API-ключ
```

```
parameters:
  - name: query
    in: query
    required: true
    schema:
      type: string
    description: Текст User Story для поиска похожих примеров
  - name: threshold
    in: query
    schema:
      type: number
      format: float
      minimum: 0
      maximum: 1
      default: 0.75
    description: Порог схожести (от 0 до 1)
responses:
  '200':
    description: Результаты поиска
    content:
      application/json:
        schema:
```

```
              $ref: '#/components/schemas/SearchResponse'
        '400':
          description: Неверный запрос
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'

components:
  schemas:
    HealthResponse:
      type: object
      properties:
        ok:
          type: boolean
          example: true
          description: Признак доступности сервиса
        timestamp:
          type: string
          format: date-time
          example: "2024-01-15T10:30:00Z"
          description: Время проверки
```

```
    version:
      type: string
      example: "1.0.0"
      description: Версия API

  AnalysisRequest:
    type: object
    required:
      - user_story
    properties:
      user_story:
        type: string
        example: "Как пользователь, я хочу зарегистрироваться в системе, чтобы получить доступ к персонализированному контенту"
        description: Текст User Story для анализа в формате "Как <роль>, я хочу <действие>, чтобы <цель>"
      use_cache:
        type: boolean
        default: true
        description: Использовать кэш и поиск в базе данных

  AnalysisResponse:
    type: object
```

```
properties:
  overall_verdict:
    type: string
    enum: [Хорошо, Средне, Плохо]
    example: "Хорошо"
    description: Общая оценка User Story
  overall_score:
    type: integer
    minimum: 0
    maximum: 100
    example: 85
    description: Оценка в баллах (0-100)
  criteria:
    type: object
    properties:
      Independent:
        type: boolean
        example: true
        description: Соответствие критерию Independent (независимость)
      Negotiable:
        type: boolean
        example: true
```

```
      description: Соответствие критерию Negotiable (обсуждаемость)
    Valuable:
      type: boolean
      example: true
      description: Соответствие критерию Valuable (ценность)
    Estimable:
      type: boolean
      example: true
      description: Соответствие критерию Estimable (оцениваемость)
    Small:
      type: boolean
      example: true
      description: Соответствие критерию Small (малый размер)
    Testable:
      type: boolean
      example: true
      description: Соответствие критерию Testable (тестируемость)
recommendations:
  type: array
  items:
    type: string
```

```
        example: ["Уточнить, какой именно персонализированный контент будет доступен", "Добавить критерии приемки для тестирования"]
        description: Список рекомендаций по улучшению User Story
      similar_stories_found:
        type: boolean
        example: false
        description: Найдены ли похожие истории в базе данных
      used_cached_result:
        type: boolean
        example: false
        description: Использован ли результат из кэша/базы данных
      analysis_id:
        type: string
        format: uuid
        example: "123e4567-e89b-12d3-a456-426614174000"
        description: Уникальный идентификатор анализа

  ExamplesResponse:
    type: object
    properties:
      examples:
        type: array
```

```
        items:
          $ref: '#/components/schemas/UserStoryExample'
      total_count:
        type: integer
        example: 150
        description: Общее количество примеров в базе
      has_more:
        type: boolean
        example: true
        description: Есть ли еще примеры для загрузки

  UserStoryExample:
    type: object
    properties:
      id:
        type: integer
        example: 1
        description: Уникальный идентификатор примера
      query:
        type: string
        example: "Как пользователь, я хочу зарегистрироваться через электронную почту, чтобы иметь возможность войти позже"
```

```
        description: Оригинальный текст User Story
      answer:
        type: string
        example: "Хорошо: соответствует критериям INVEST"
        description: Результат анализа
      normalized_query:
        type: string
        example: "как пользователь я хочу зарегистрироваться через электронную почту чтобы иметь возможность войти позже"
        description: Нормализованный текст для поиска

    SearchResponse:
      type: object
      properties:
        similar_stories:
          type: array
          items:
            $ref: '#/components/schemas/SimilarStory'
        query:
          type: string
          example: "Как пользователь, я хочу зарегистрироваться"
          description: Исходный запрос для поиска
```

```
        threshold:
          type: number
          format: float
          example: 0.75
          description: Использованный порог схожести
        total_found:
          type: integer
          example: 3
          description: Количество найденных похожих историй

    SimilarStory:
      type: object
      properties:
        story:
          type: string
          example: "Как пользователь, я хочу зарегистрироваться через электронную почту, чтобы иметь возможность войти позже"
          description: Текст похожей User Story
        analysis:
          type: string
          example: "Хорошо: соответствует критериям INVEST"
          description: Результат анализа похожей истории
```

```
    similarity_score:
      type: number
      format: float
      minimum: 0
      maximum: 1
      example: 0.89
      description: Коэффициент схожести (0-1)

ErrorResponse:
  type: object
  properties:
    error:
      type: string
      example: "Invalid request"
      description: Тип ошибки
    message:
      type: string
      example: "User story text is required"
      description: Сообщение об ошибке
    timestamp:
      type: string
      format: date-time
```

```
      example: "2024-01-15T10:30:00Z"
      description: Время возникновения ошибки

ValidationError:
  type: object
  properties:
    detail:
      type: array
      items:
        type: object
        properties:
          loc:
            type: array
            items:
              type: string
            example: ["body", "user_story"]
          msg:
            type: string
            example: "field required"
          type:
            type: string
            example: "value_error.missing"
```

```
securitySchemes:
  ApiKeyAuth:
    type: apiKey
    in: header
    name: X-API-Key
    description: Секретный API-ключ для доступа к защищенным endpoint'ам
```

AI-агент для проверки User Stories по INVEST

1. Функциональные требования

1.1 Критерии приемки

Функциональность: Анализ User Story в Telegram-боте

Сценарий 1: Успешный анализ User Story

Дано: Пользователь открывает Telegram-бота командой /start

Когда: Пользователь отправляет валидный текст User Story

Тогда: Бот возвращает структурированный отчет с нарушениями критериев INVEST и рекомендациями по исправлению

Сценарий 2: Обнаружение неизмеримых критериев

Дано: Пользователь отправил User Story со словами "быстро", "удобно"

Когда: Бот анализирует текст

Тогда: Бот отмечает нарушение критерия "Estimable" и предлагает заменить на конкретные метрики

Сценарий 3: Проверка ценности для пользователя

Дано: Пользователь отправил User Story с фразой "Как система, я хочу..."

Когда: Бот анализирует текст

Тогда: Бот отмечает нарушение критерия "Valuable" и предлагает переформулировать с точки зрения пользователя

Сценарий 4: Проверка размера Story

Дано: Пользователь отправил User Story длиннее 500 символов или короче 10 символов

Когда: Бот анализирует текст

Тогда: Бот предупреждает о нарушении критерия "Small" и ограничениях длины

Сценарий 5: Обработка невалидного ввода

Дано: Пользователь отправил пустое сообщение или текст не в формате User Story

Когда: Бот проверяет ввод

Тогда: Бот возвращает понятное сообщение об ошибке с примером корректного формата

1.2 User Story

Как системный аналитик, я хочу отправлять текст User Story в Telegram-бота, чтобы получать мгновенный отчет о нарушениях критериев INVEST с рекомендациями по улучшению.

2. Нефункциональные требования

2.1 Производительность

Время отклика бота: < 3 секунд

Время анализа через LLM: < 30 секунд

Время обработки запроса: < 5 секунд (исключая LLM)

Поддержка пользователей: до 10 одновременных сессий

2.2 Надежность

Доступность системы: 99.5% в месяц

Время восстановления: < 15 минут

Успешность запросов к LLM: ≥ 95%

Retry механизм: до 3 попыток при временных ошибках

2.3 Безопасность

Шифрование данных: TLS 1.2+ для всех коммуникаций

Валидация входных данных: защита от XSS и SQL-инъекций

Rate limiting: не более 10 запросов в час на пользователя

Хранение ключей: в environment variables

2.4 Масштабируемость

Горизонтальное масштабирование: поддержка до 5 инстансов

Балансировка нагрузки: автоматическое распределение запросов

Масштабирование БД: read replicas для запросов истории

2.5 Удобство использования

Время обучения: < 3 минут для освоения функций

Количество шагов: 1 шаг для основного сценария

Ясность сообщений: понятные формулировки на русском языке

Обратная связь: индикаторы прогресса операций

2.6 Совместимость

Telegram: поддержка всех версий Telegram Messenger

Мобильные устройства: адаптивный интерфейс бота

API: REST API спецификация

Форматы данных: JSON для API, Markdown для отчетов

3. Архитектура системы

3.1 Компоненты

Telegram Bot → Backend API (Python) → LLM → Database (SQLite)

3.2 База данных (актуальная схема)

users - пользователи Telegram

user_stories - анализируемые User Stories

violations - выявленные нарушения INVEST

reference_examples - эталонные примеры

bot_sessions - сессии и состояния бота

3.3 API Эндпоинты

POST /analyze - анализ User Story

GET /history - получение истории анализов

GET /examples - получение обучающих примеров

4. Процесс анализа

Основной сценарий:

Валидация входных данных

Сохранение в БД со статусом "processing"

Вызов LLM для анализа INVEST критериев

Обогащение результатов через поиск похожих примеров

Сохранение нарушений и рекомендаций

Возврат структурированного отчета

Ответ включает:

Общую оценку ("Хорошо", "Средне", "Плохо")

Балльную оценку (0-100)

Детализацию по каждому критерию INVEST

Конкретные нарушения с фрагментами текста

Рекомендации по исправлению

Похожие эталонные примеры

5. Метрики успеха

Производительность: < 3 сек время отклика для 95% запросов

Надежность: 99.5% доступность, MTTR < 15 мин

Качество: точность анализа INVEST ≥ 90%

Безопасность: 0 критических уязвимостей

Удобство: 90% пользователей осваивают за 3 минуты

6. Мониторинг и логирование

Метрики: Prometheus + Grafana

Логи: структурированное логирование с контекстом

Алерты: автоматические уведомления о проблемах

Аналитика: отслеживание использования функций