User Story для INVEST-Checker Bot

Основная User Story:

Как Product Owner/разработчик, я хочу автоматически анализировать User Stories по критериям INVEST, чтобы быстро оценивать их качество и получать конкретные рекомендации по улучшению.

Детализированные User Stories

US-001: Анализ User Story по INVEST

Как член команды разработки, я хочу отправлять текст User Story боту и получать оценку по 6 критериям INVEST, чтобы объективно оценить качество истории перед включением в бэклог.

Критерии приемки:

- бот принимает User Story в формате

"Как <роль>, я хочу <действие>, чтобы <цель>"

- возвращает оценку от 0 до 6 баллов

- показывает выполненные и невыполненные критерии INVEST

- дает конкретные рекомендации по улучшению

US-002: Улучшение User Story через ИИ

Как Product Owner,

Я хочу автоматически улучшать слабые User Stories с помощью AI, чтобы быстро доводить их до стандартов качества без ручной правки.

Критерии приемки:

- предлагает кнопку "Улучшить историю" после анализа

- улучшенная версия сохраняет оригинальную цель и ценность

- улучшенная история имеет более высокую оценку INVEST

- пользователь может выбрать между несколькими вариантами улучшений

US-003: Поиск похожих историй в базе

Как аналитик, я хочу находить похожие User Stories в существующей базе,

Чтобы избегать дублирования и использовать проверенные формулировки.

Критерии приемки:

- автоматически ищет похожие истории при анализе новой
- показывается процент схожести с существующими историями
- пользователь может выбрать готовую историю из базы
- приоритет отдается "золотым" историям с высокой оценкой

US-004: Экспорт результатов анализа

Как руководитель проекта, я хочу экспортировать результаты анализа в разные форматы, чтобы делиться отчетами с командой и сохранять их в документации.

Критерии приемки:

- экспорт в TXT формате с читабельной структурой
- экспорт в CSV для анализа в таблицах
- экспорт включает оригинальную историю и анализ
- файлы имеют понятные имена и временные метки

US-005: Управление базой знаний

Как архитектор знаний, я хочу добавлять качественные User Stories в базу знаний, чтобы постепенно накапливать библиотеку эталонных примеров.

Критерии приемки:

- автоматическое добавление улучшенных историй в "золотую" коллекцию
- ручное добавление историй с оценкой 5/6 и выше
- просмотр всей базы с пагинацией
- поиск по существующим историям

Приоритет 1 (MVP):

US-001 - Базовый анализ INVEST

US-002 - Улучшение через ИИ

US-003 - Поиск похожих историй

Приоритет 2:

US-004 - Экспорт результатов

US-005 - Управление базой знаний

Тестовый сценарий для начала:

1. Пользователь отправляет: "Как клиент, я хочу фильтровать товары по цене, чтобы найти подходящий вариант"

2. Бот анализирует и показывает оценку

3. Бот предлагает улучшить историю

4. После улучшения показывается версия с более четкими критериями приемки

5. Пользователь добавляет улучшенную версию в базу знаний

Критерии успеха проекта

Метрики успеха:

- 90% User Stories проходят анализ за менее чем 10 секунд
- улучшенные истории имеют на 1-2+ балла выше оригинальных
- база знаний содержит 20+ историй

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
	- Используется многоуровневое кэширование
	- Автоматический поиск похожих историй перед запросом к LLM
Триггер	Пользователь отправляет текст User Story боту
Основной сценарий	1. Система проверяет валидность формата User Story
	2. Система проверяет кэш анализа (исключает дублирующие запросы к LLM)
	3. Система ищет точные (100%) и похожие (≥85%) совпадения в базе
	4. При нахождении точного совпадения используется кэшированный анализ
	5. При нахождении похожих историй предлагается выбор вариантов
	6. Если совпадений нет - запрос к LLM для анализа INVEST
	7. Система отображает структурированный отчет с оценкой 0-6/6
	8. Предлагаются опции: улучшить, экспорт, добавить в базу, история улучшений
Альтернативный сценарий	1а. При нестандартной формулировке (но похожей на User Story) бот все равно анализирует
Исключительный сценарий	6а. При недоступности LLM используется расширенный кэш и похожие истории

UC-02: Улучшение User Story

Заголовок	Многошаговое улучшение User Story через ИИ
Акторы	системный аналитик, продукт-оунер
Предусловие	Выполнен анализ User Story
	Доступен LLM сервис
Ограничения	Сохраняется цепочка улучшений с историей версий
	Можно улучшать несколько раз подряд
	Улучшенные версии автоматически добавляются в коллекцию при оценке ≥4
Триггер	Пользователь нажимает кнопку "Улучшить историю"
Основной сценарий	1. Система создает/продолжает цепочку улучшений
	2. LLM генерирует улучшенную версию с учетом предыдущего анализа
	3. Система сохраняет все версии с timestamp
	4. Пользователь может анализировать улучшенную версию
	5. Пользователь может улучшать дальше или просмотреть историю улучшений
Альтернативный сценарий	4а. Пользователь экспортирует улучшенную версию без анализа

UC-03: Просмотр базы User Stories

Заголовок	Просмотр и навигация по базе User Stories
Акторы	системный аналитик, начинающий аналитик
Предусловие	База содержит хотя бы одну User Story
Ограничения	Пагинация по 5 историй на страницу
	Фильтрация по качеству (золотые/обычные)

	Просмотр деталей каждой истории
Триггер	Пользователь нажимает кнопку "База US"
Основной сценарий	1. Система показывает список историй с пагинацией
	2. Для каждой истории отображается: текст (обрезанный), оценка, статус (золотая/обычная)
	3. Пользователь может листать страницы
	4. При выборе истории показываются полные детали: текст, анализ, дата добавления
	5. Доступны действия: проанализировать, улучшить
Исключительный сценарий	2а. Если база примеров недоступна, система использует встроенные базовые примеры

UC-04: Автоматический поиск похожих User Stories

Заголовок	Интеллектуальный поиск похожих User Stories
Акторы	система (автоматически)
Предусловие	Пользователь отправил User Story для анализа
Ограничения	Три уровня поиска: точные совпадения (100%), высокое сходство (≥95%), обычное сходство (≥85%)
	Использование кэша поисковых запросов
	Приоритет золотым историям
Триггер	Автоматически после получения User Story
Основной сценарий	1. Система ищет точные совпадения (нормализованный текст)
	2. При нахождении - использует кэшированный анализ (исключает LLM)
	3. При высоком сходстве предлагает выбор из похожих историй

Основной сценарий	4. Пользователь может выбрать готовую историю или использовать свою
	5. При выборе готовой истории увеличивается счетчик использования

UC-05: Экспорт результатов

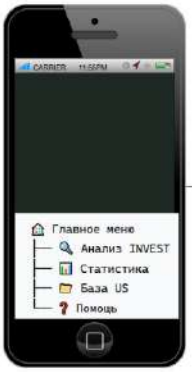
Заголовок	Многовариантный экспорт результатов
Акторы	руководитель проекта, аналитик
Предусловие	Имеются результаты анализа или улучшенная история
Ограничения	Поддержка форматов: TXT, CSV
	Экспорт как анализа, так и улучшенных историй
	Автоматическое именование файлов
Триггер	Пользователь нажимает кнопку "Экспорт"
Основной сценарий	1. Система предлагает выбор формата экспорта
	2. Для анализа: включает историю, анализ, временную метку
	3. Для улучшенных историй: только текст улучшенной версии
	4. Файл отправляется как документ Telegram
	5. Пользователь возвращается к предыдущему контексту

UC-06: Просмотр статистики

Заголовок	Расширенная статистика системы
Акторы	руководитель проекта, аналитик
Предусловие	Бот работает более 1 часа
Ограничения	Статистика обновляется в реальном времени
	Включает метрики LLM и кэширования

Триггер	Пользователь нажимает кнопку "Экспорт"
Основной сценарий	1. Система показывает: аптайм, пользователей, всего историй, золотых историй
	2. Показывает эффективность кэширования (hit rate)
	3. Отображает статистику LLM: запросы, токены, использование кэша
	4. Рассчитывает экономию за счет кэширования

главное меню

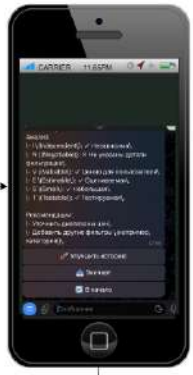


анализ US

пользователь вбивает историю

система находит похожую историю в базе

нажимаем кнопку экспорт

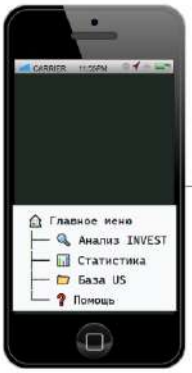


выходит выбор формата для экспорта

после выбора формата txt происходит автоматический экспорт

экспортированный файл можно сразу посмотреть

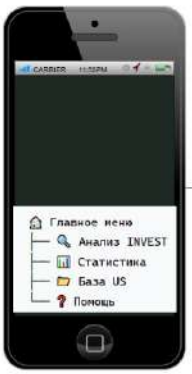
главное меню



база US

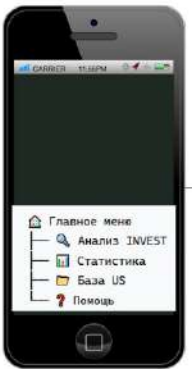
база US

главное меню



статистика

главное меню



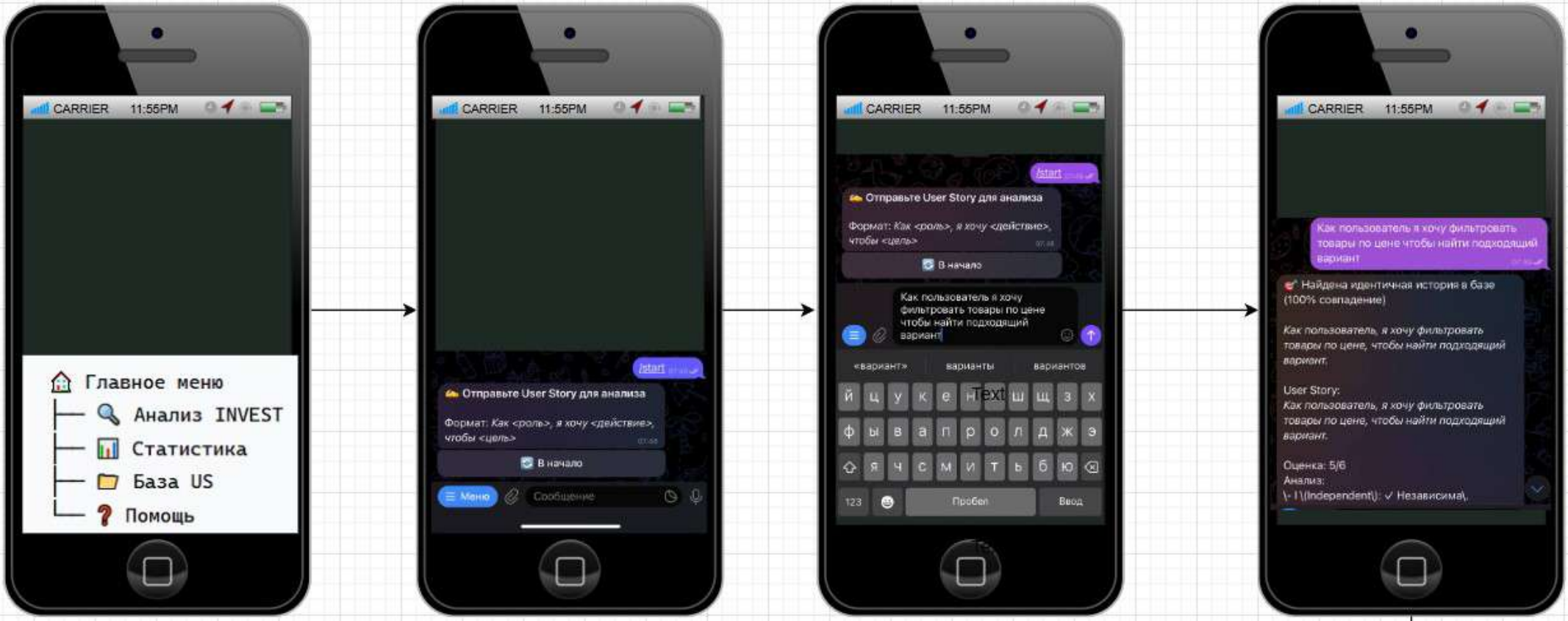
помощь

главное меню

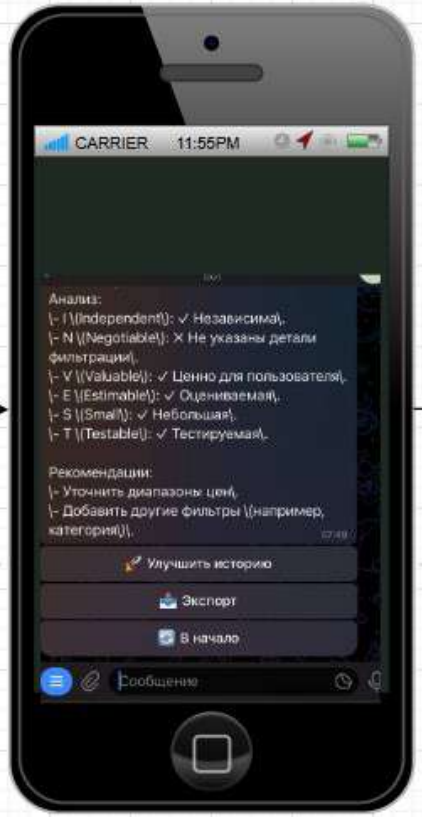
анализ US

пользователь вбивает историю

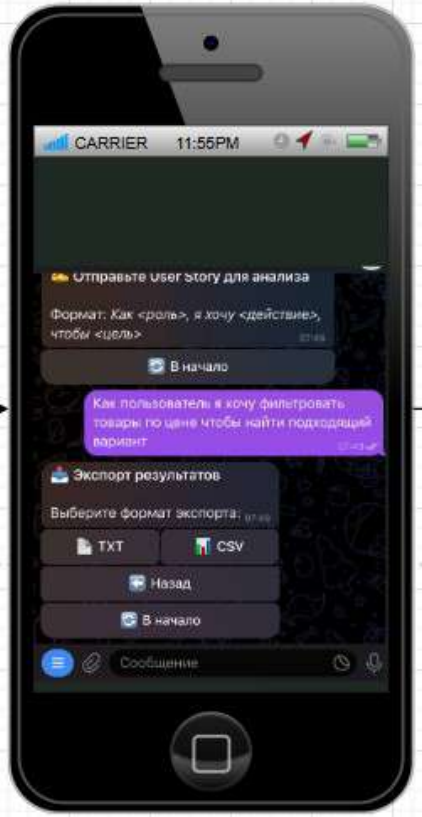
система находит похожую историю в базе



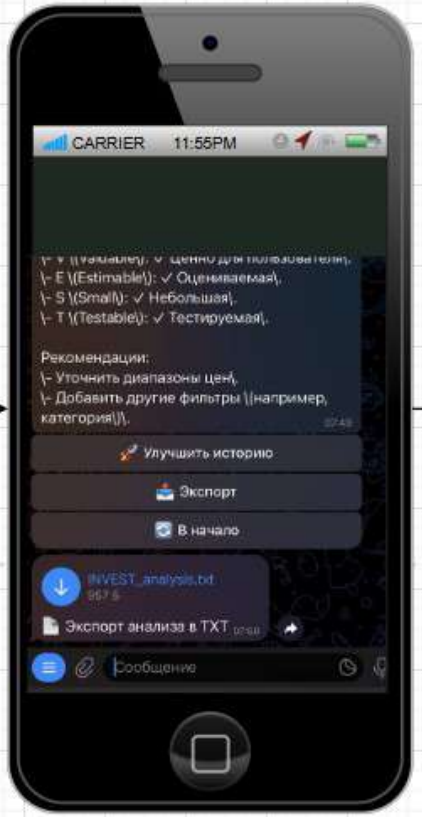
нажимаем кнопку экспорт



выходит выбор формата для экспорта

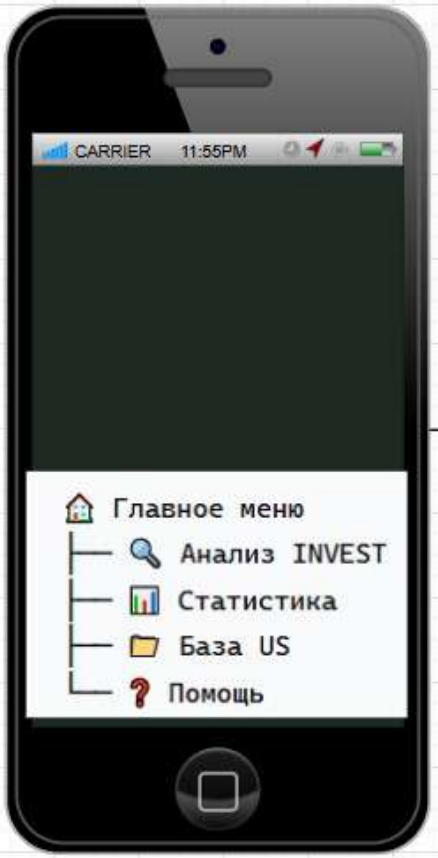


после выбора формата txt происходит автоматический экспорт



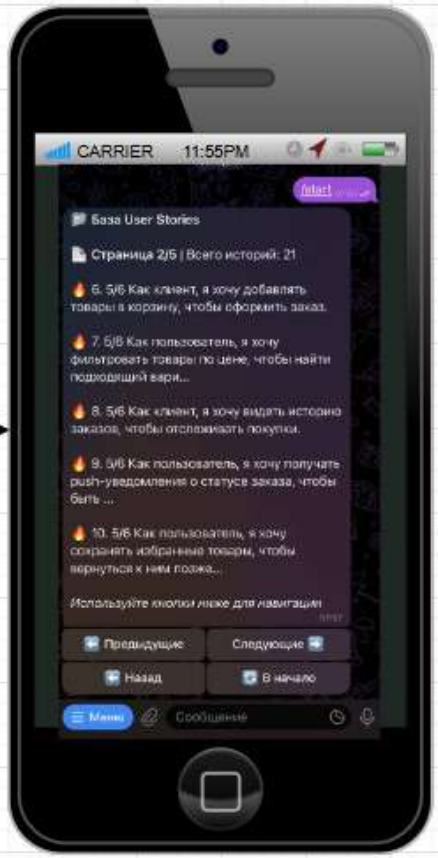
экспортированный файл можно сразу посмотреть

главное меню



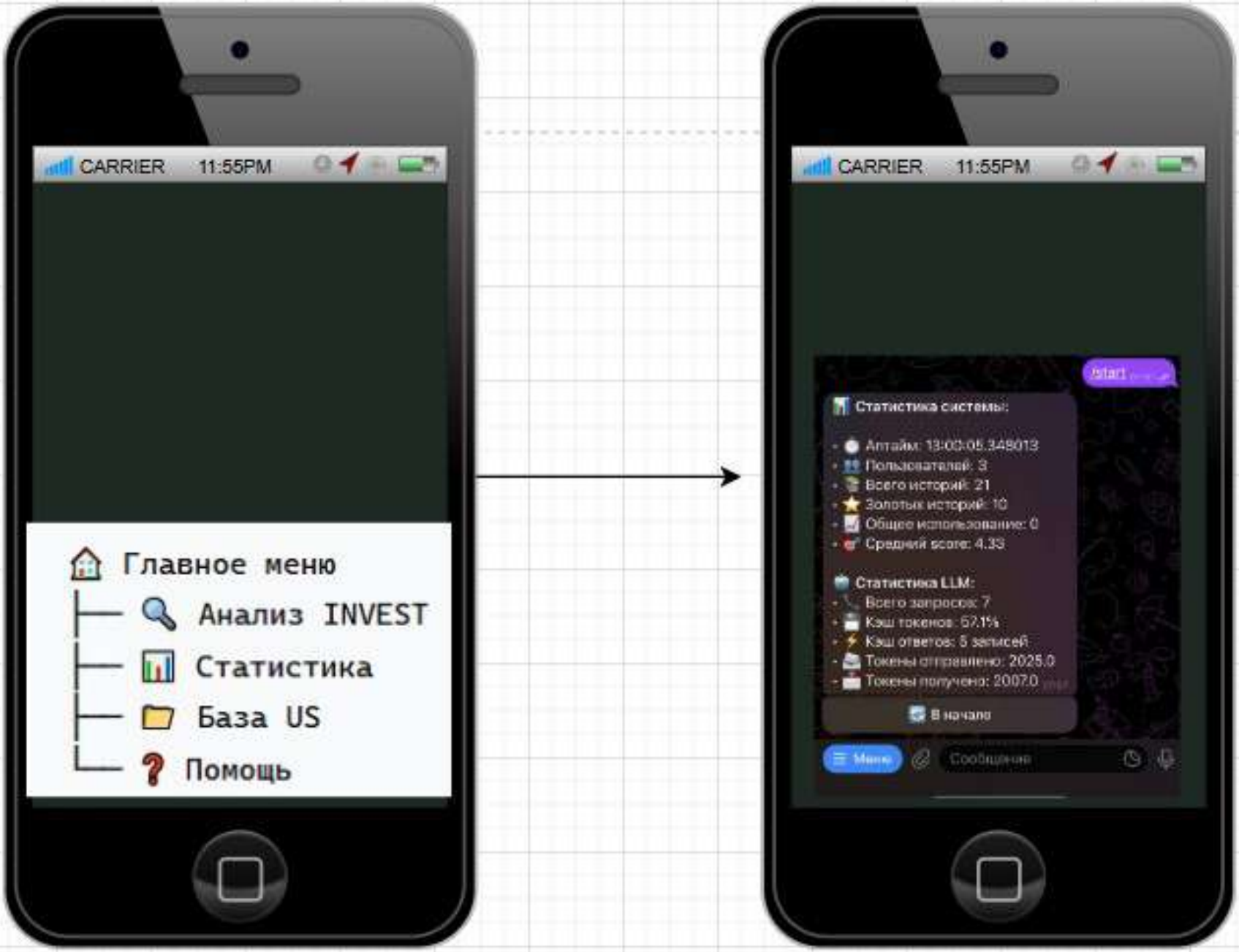
база US

база US



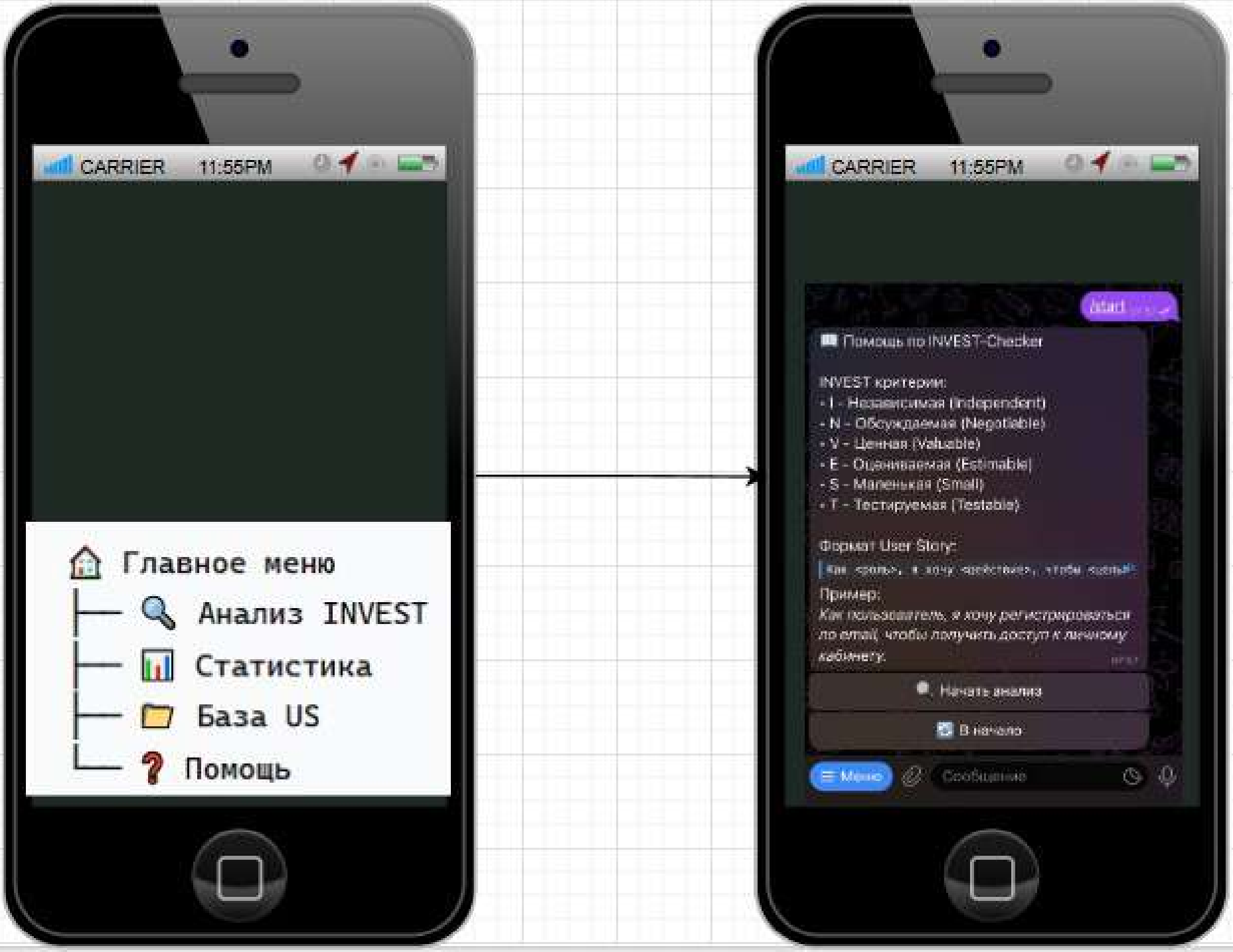
главное меню

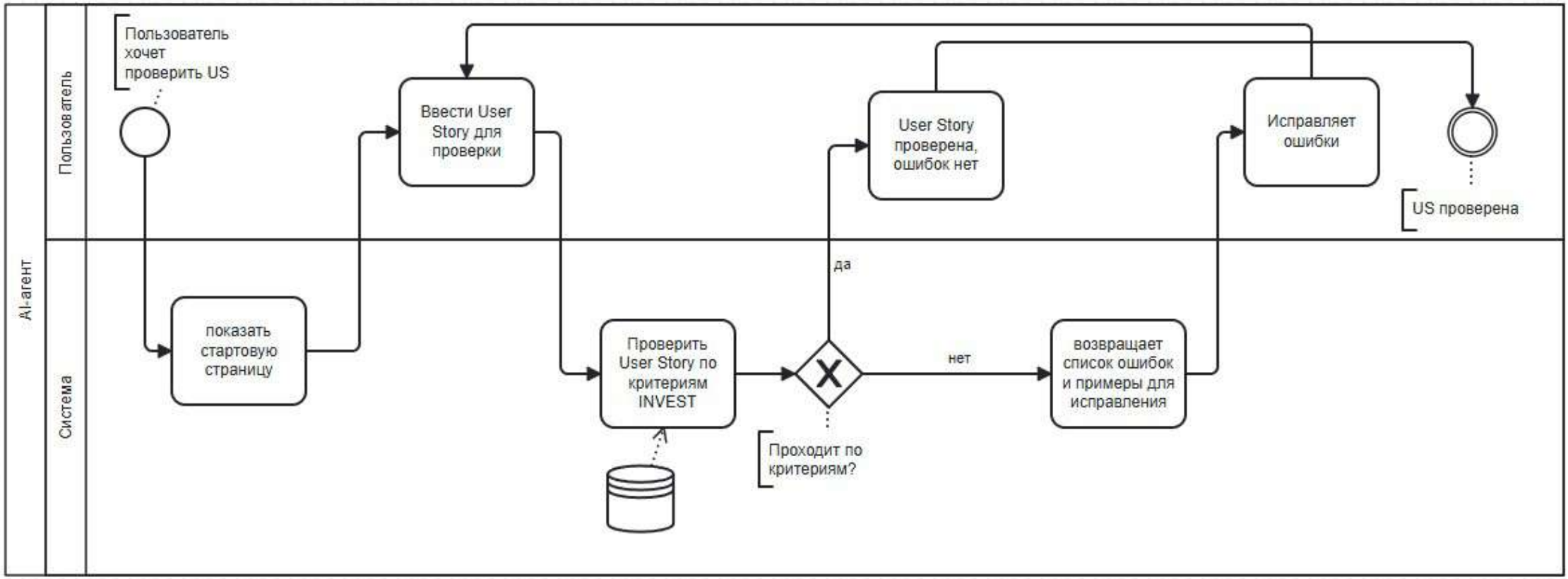
статистика

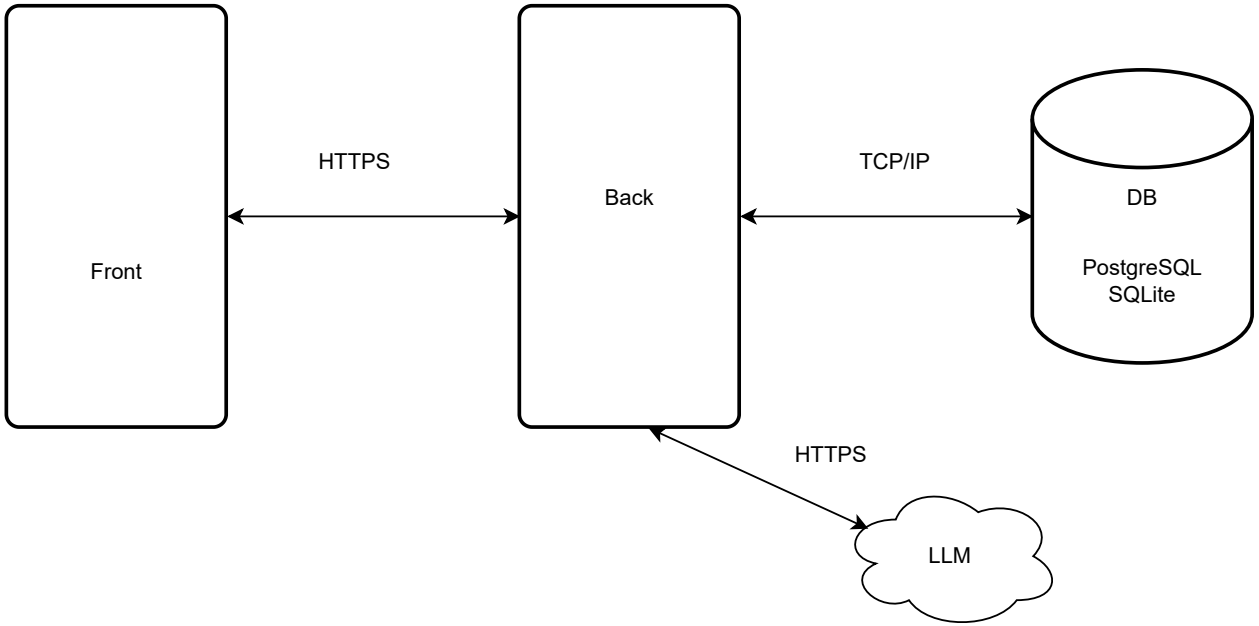


главное меню

помощь




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?

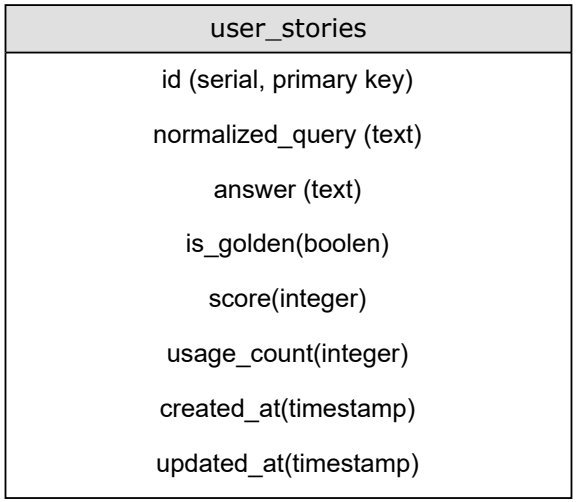

Front
HTTPS
Back
TCP/IP
DB
PostgreSQL
SQLite
HTTPS
LLM

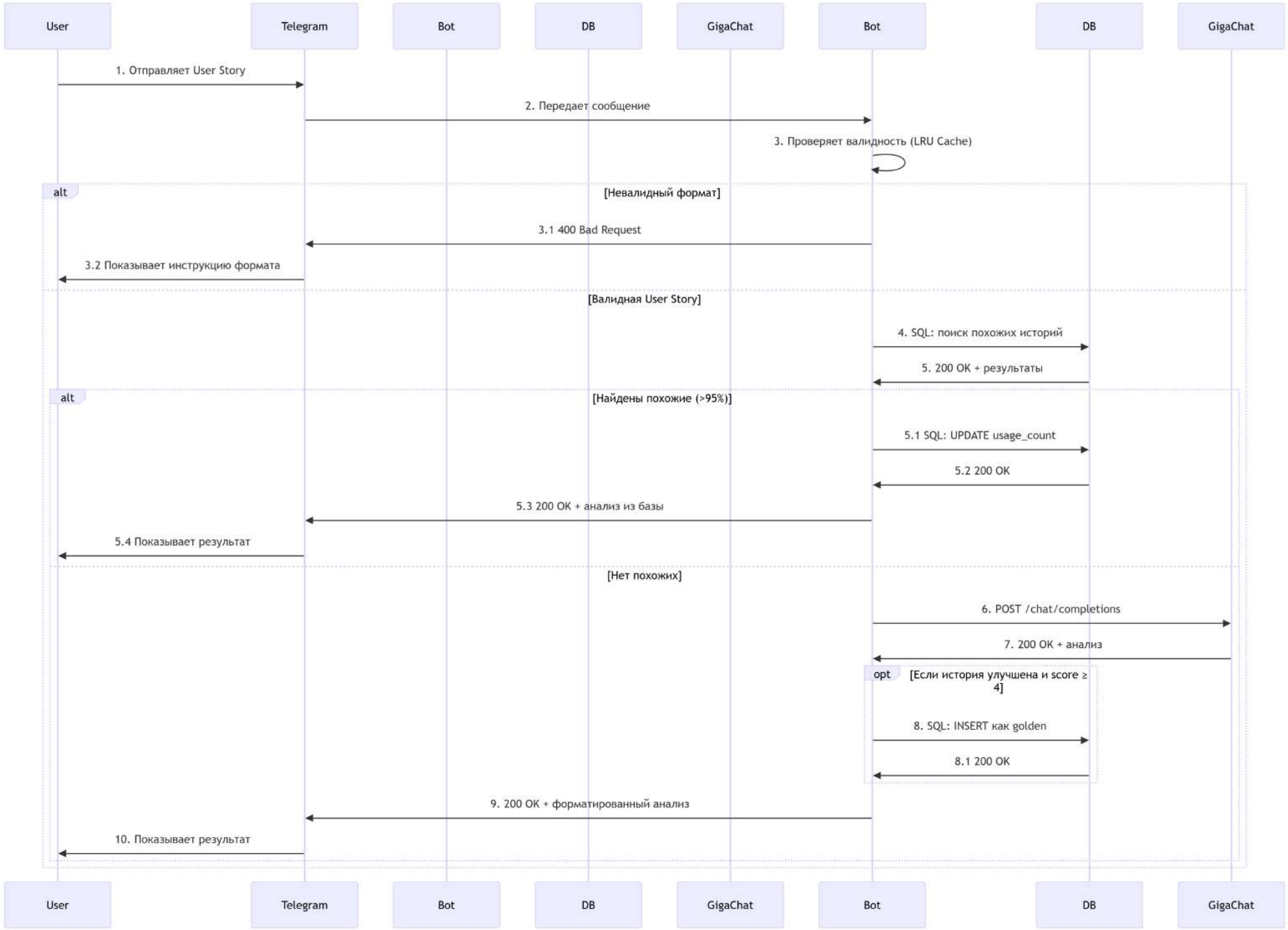
Модель данных

Таблица: user_stories

Атрибут	Тип	Описание
id	INTEGER (PK)	Уникальный идентификатор истории
query	TEXT	Оригинальный текст User Story
normalized_query	TEXT	Нормализованный текст для поиска
answer	TEXT	Результат анализа INVEST
is_golden	BOOLEAN	Признак "золотой" истории
score	INTEGER	Оценка INVEST (0-6)
usage_count	INTEGER	Счетчик использований
created_at	DATETIME	Дата создания
updated_at	DATETIME	Дата обновления

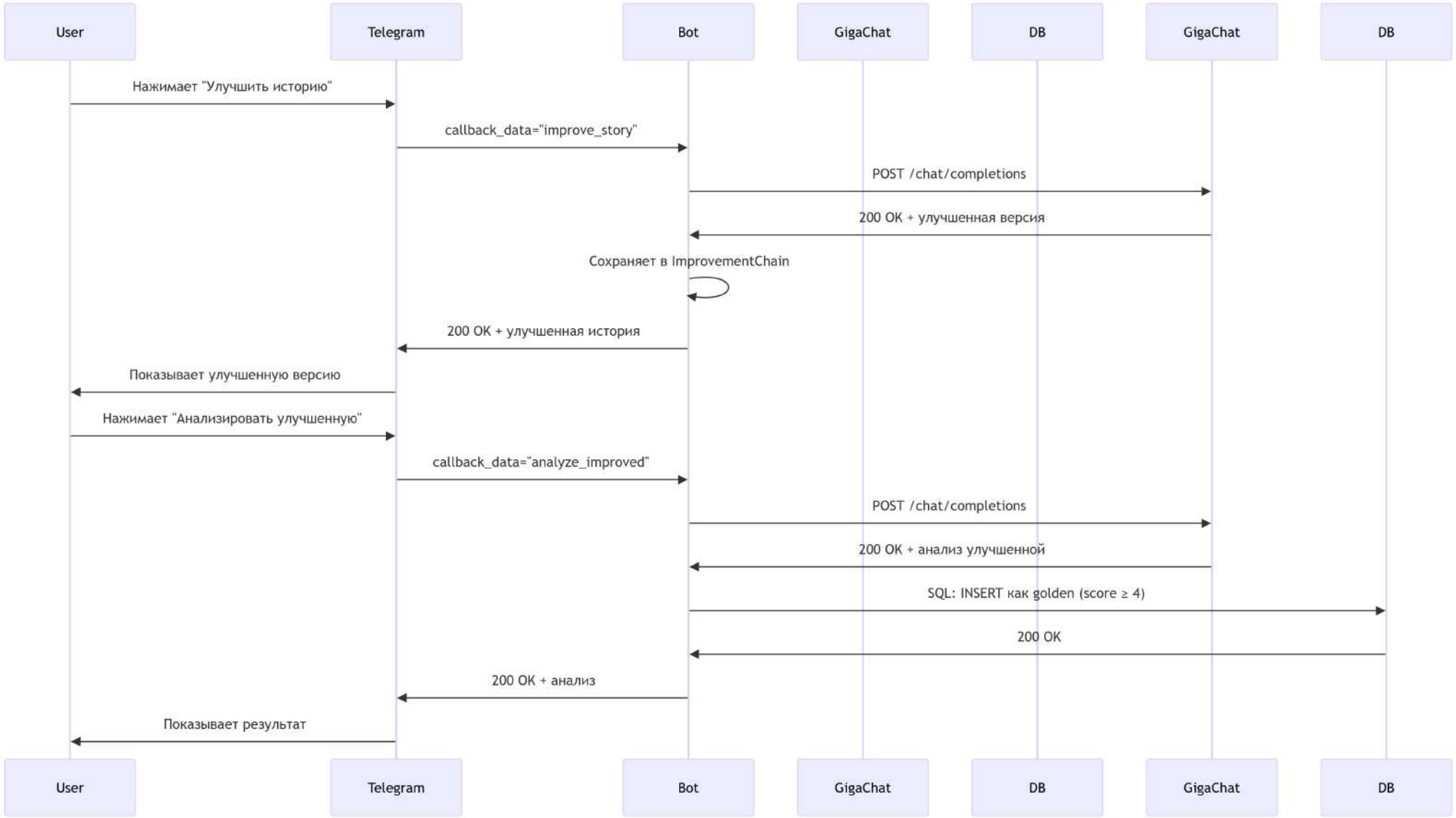
ERD диаграмма




User
Telegram
Bot
DB
GigaChat
Bot
DB
GigaChat
1. Отправляет User Story
2. Передает сообщение
3. Проверяет валидность (LRU Cache)
alt
[Невалидный формат]
3.1 400 Bad Request
3.2 Показывает инструкцию формата
[Валидная User Story]
4. SQL: поиск похожих историй
5. 200 OK + результаты
alt
[Найдены похожие (>95%)]
5.1 SQL: UPDATE usage_count
5.2 200 OK
5.3 200 OK + анализ из базы
5.4 Показывает результат
[Нет похожих]
6. POST /chat/completions
7. 200 OK + анализ
opt
[Если история улучшена и score ≥ 4]
8. SQL: INSERT как golden
8.1 200 OK
9. 200 OK + форматированный анализ
10. Показывает результат
User
Telegram
Bot
DB
GigaChat
Bot
DB
GigaChat

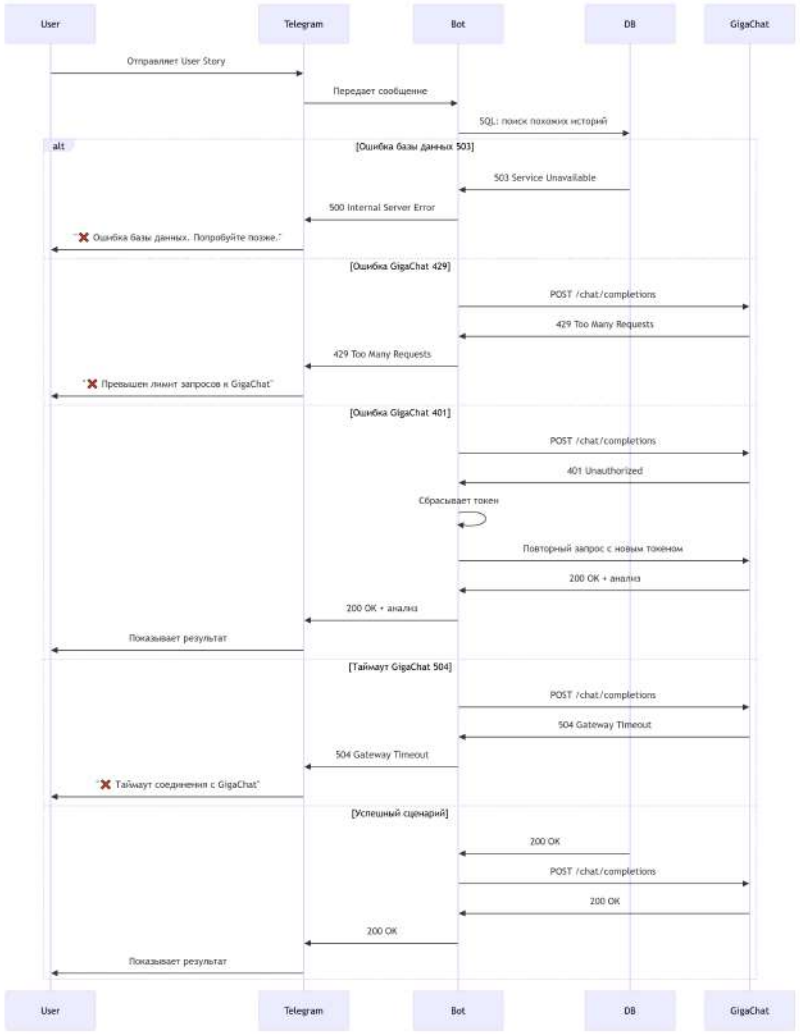
Основной сценарий анализа User Story

№	Описание	Компонент	HTTP Status
1	Пользователь отправляет User Story в Telegram бот	User → Telegram	-
2	Telegram передает сообщение обработчику бота	Telegram → INVEST-Bot	-
3	Бот проверяет валидность User Story (использует LRU кэш)	INVEST-Bot	-
3.1	Если формат невалидный - отправляет сообщение об ошибке	INVEST-Bot → Telegram	400 Bad Request
3.2	Пользователь получает инструкцию по формату	Telegram → User	-
4	Бот ищет похожие истории в базе данных	INVEST-Bot → PostgreSQL DB	-
5	База данных возвращает результаты поиска	PostgreSQL DB → INVEST-Bot	200 OK
5.1	Если найдены похожие истории (>95%), бот увеличивает счетчик использования	INVEST-Bot → PostgreSQL DB	-
5.2	База данных подтверждает обновление	PostgreSQL DB → INVEST-Bot	200 OK
5.3	Бот отправляет анализ из базы данных	INVEST-Bot → Telegram	200 OK
5.4	Пользователь получает результат анализа	Telegram → User	-
6	Если похожих историй не найдено, бот обращается к GigaChat API	INVEST-Bot → GigaChat API	POST /chat/completions
7	GigaChat API возвращает анализ User Story	GigaChat API → INVEST-Bot	200 OK
8	Для улучшенных историй с оценкой ≥ 4 бот сохраняет в базу как golden	INVEST-Bot → PostgreSQL DB	SQL INSERT
8.1	База данных подтверждает сохранение	PostgreSQL DB → INVEST-Bot	200 OK
9	Бот отправляет форматированный анализ	INVEST-Bot → Telegram	200 OK
10	Пользователь получает результат анализа	Telegram → User	-


User
Telegram
Bot
GigaChat
DB
GigaChat
DB
Нажимает "Улучшить историю"
callback_data="improve_story"
POST /chat/completions
200 OK + улучшенная версия
Сохраняет в ImprovementChain
200 OK + улучшенная история
Показывает улучшенную версию
Нажимает "Анализировать улучшенную"
callback_data="analyze_improved"
POST /chat/completions
200 OK + анализ улучшенной
SQL: INSERT как golden (score ≥ 4)
200 OK
200 OK + анализ
Показывает результат
User
Telegram
Bot
GigaChat
DB
GigaChat
DB

Сценарий улучшения истории

№	Описание	Компонент	HTTP Status
1	Пользователь нажимает кнопку "Улучшить историю"	User → Telegram	-
2	Telegram передает callback запрос боту	Telegram → INVEST-Bot	callback_data="improve_story"
3	Бот отправляет запрос на улучшение истории в GigaChat	INVEST-Bot → GigaChat API	POST /chat/completions
4	GigaChat API возвращает улучшенную версию User Story	GigaChat API → INVEST-Bot	200 OK
5	Бот сохраняет улучшенную версию в цепочку улучшений	INVEST-Bot	-
6	Бот отправляет улучшенную версию пользователю	INVEST-Bot → Telegram	200 OK
7	Пользователь видит улучшенную версию истории	Telegram → User	-
8	Пользователь нажимает "Анализировать улучшенную"	User → Telegram	-
9	Telegram передает callback запрос боту	Telegram → INVEST-Bot	callback_data="analyze_improved"
10	Бот отправляет улучшенную историю на анализ в GigaChat	INVEST-Bot → GigaChat API	POST /chat/completions
11	GigaChat API возвращает анализ улучшенной истории	GigaChat API → INVEST-Bot	200 OK
12	Если оценка ≥ 4, бот сохраняет улучшенную историю как golden	INVEST-Bot → PostgreSQL DB	SQL INSERT
13	База данных подтверждает сохранение	PostgreSQL DB → INVEST-Bot	200 OK
14	Бот отправляет анализ улучшенной истории	INVEST-Bot → Telegram	200 OK
15	Пользователь получает результат анализа	Telegram → User	-


User
Telegram
Bot
DB
GigaChat
Отправляет User Story
Передает сообщение
SQL: поиск похожих историй
alt
[Ошибка базы данных 503]
503 Service Unavailable
500 Internal Server Error
"❌ Ошибка базы данных. Попробуйте позже."
[Ошибка GigaChat 429]
POST /chat/completions
429 Too Many Requests
429 Too Many Requests
"❌ Превышен лимит запросов к GigaChat"
[Ошибка GigaChat 401]
POST /chat/completions
401 Unauthorized
Сбрасывает токен
Повторный запрос с новым токеном
200 OK + анализ
200 OK + анализ
Показывает результат
[Таймаут GigaChat 504]
POST /chat/completions
504 Gateway Timeout
504 Gateway Timeout
"❌ Таймаут соединения с GigaChat"
[Успешный сценарий]
200 OK
POST /chat/completions
200 OK
200 OK
Показывает результат
User
Telegram
Bot
DB
GigaChat

3. Сценарии обработки ошибок

№	Описание	Компонент	HTTP Status
1	Пользователь отправляет User Story в Telegram бот	User → Telegram	-
2	Telegram передает сообщение обработчику бота	Telegram → INVEST-Bot	-
3	Бот пытается найти похожие истории в базе данных	INVEST-Bot → PostgreSQL DB	-
3.1	Ошибка базы данных: сервис недоступен	PostgreSQL DB → INVEST-Bot	503 Service Unavailable
3.2	Бот отправляет сообщение об ошибке базы данных	INVEST-Bot → Telegram	500 Internal Server Error
3.3	Пользователь получает уведомление об ошибке	Telegram → User	"❌ Ошибка базы данных. Попробуйте позже."
4	Ошибка GigaChat: превышен лимит запросов	GigaChat API → INVEST-Bot	429 Too Many Requests
4.1	Бот отправляет сообщение о превышении лимита	INVEST-Bot → Telegram	429 Too Many Requests
4.2	Пользователь получает уведомление о лимите	Telegram → User	"❌ Превышен лимит запросов к GigaChat"
5	Ошибка GigaChat: неавторизованный доступ	GigaChat API → INVEST-Bot	401 Unauthorized
5.1	Бот сбрасывает токен аутентификации	INVEST-Bot	-
5.2	Бот повторяет запрос с новым токеном	INVEST-Bot → GigaChat API	POST /chat/completions
5.3	GigaChat API успешно обрабатывает повторный запрос	GigaChat API → INVEST-Bot	200 OK
5.4	Бот отправляет успешный результат	INVEST-Bot → Telegram	200 OK
5.5	Пользователь получает результат анализа	Telegram → User	-
6	Ошибка GigaChat: таймаут соединения	GigaChat API → INVEST-Bot	504 Gateway Timeout
6.1	Бот отправляет сообщение о таймауте	INVEST-Bot → Telegram	504 Gateway Timeout

6.2	Пользователь получает уведомление о таймауте	Telegram → User	"❌ Таймаут соединения с GigaChat"
7	Успешный сценарий: все компоненты работают корректно	Все компоненты	200 OK
7.1	База данных возвращает результаты	PostgreSQL DB → INVEST-Bot	200 OK
7.2	GigaChat API возвращает анализ	GigaChat API → INVEST-Bot	200 OK
7.3	Бот отправляет финальный результат	INVEST-Bot → Telegram	200 OK
7.4	Пользователь получает результат анализа	Telegram → User	-

REST API в табличном виде для INVEST-Checker AI Agent

1. Endpoint: Анализ User Story

Request

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
user_story	string	body	Текст User Story для анализа	да
similarity_threshold	float	query	Порог схожести для поиска (0.0-1.0)	нет
use_cache	boolean	query	Использовать кэширование	нет

Response (200 OK)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
analysis	object	body	Результат анализа	да
score	integer	body	Оценка по INVEST (0-6)	да
criteria	object	body	Результаты по критериям	да
independent	boolean	body	Независимость	да
negotiable	boolean	body	Обсуждаемость	да
valuable	boolean	body	Ценность	да
estimable	boolean	body	Оцениваемость	да
small	boolean	body	Маленький размер	да
testable	boolean	body	Тестируемость	да
recommendations	array	body	Рекомендации по улучшению	да
similar_stories	array	body	Похожие истории из базы	нет
source	string	body	Источник анализа (cache/db/gigachat)	да
processing_time	float	body	Время обработки в секундах	да

Response (400 Bad Request)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
error	object	body	Информация об ошибке	да
code	string	body	"INVALID_USER_STORY"	да
message	string	body	Описание ошибки формата	да
details	object	body	Детали ошибки	нет

Response (429 Too Many Requests)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
error	object	body	Информация об ошибке	да
code	string	body	"RATE_LIMIT_EXCEEDED"	да
message	string	body	"Превышен лимит запросов к GigaChat"	да
retry_after	integer	body	Время до повтора в секундах	да

Response (503 Service Unavailable)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
error	object	body	Информация об ошибке	да
code	string	body	"GIGACHAT_UNAVAILABLE"	да
message	string	body	"Сервис GigaChat временно недоступен"	да

2. Endpoint: Улучшение User Story

Request

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
user_story	string	body	Текст User Story для улучшения	да
current_analysis	string	body	Текущий анализ (опционально)	нет
improvement_focus	array	body	Критерии для фокуса улучшения	нет

Response (200 OK)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
improved_story	string	body	Улучшенная версия User Story	да
improvements	array	body	Список внесенных улучшений	да
original_score	integer	body	Оценка оригинальной версии	нет
expected_score	integer	body	Ожидаемая оценка улучшенной версии	да

Response (400 Bad Request)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
error	object	body	Информация об ошибке	да
code	string	body	"IMPROVEMENT_FAILED"	да
message	string	body	"Не удалось улучшить User Story"	да

3. Endpoint: Поиск похожих историй
Request

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
query	string	query	Текст для поиска	да
threshold	float	query	Порог схожести (0.0-1.0)	нет
limit	integer	query	Лимит результатов	нет
only_golden	boolean	query	Только золотые истории	нет

Response (200 OK)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
stories	array	body	Найденные истории	да
id	integer	body	ID истории	да
query	string	body	Текст истории	да
answer	string	body	Анализ истории	да
similarity	float	body	Процент схожести	да
score	integer	body	Оценка INVEST	да
is_golden	boolean	body	Золотая история	да
total_count	integer	body	Общее количество найденных	да

4. Endpoint: Получение статистики
Request

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
period	string	query	Период (day/week/month)	нет

Response (200 OK)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
statistics	object	body	Статистика системы	да
total_stories	integer	body	Всего историй в базе	да
golden_stories	integer	body	Количество золотых историй	да
average_score	float	body	Средняя оценка	да
cache_hit_rate	float	body	Эффективность кэша	да
total_requests	integer	body	Всего запросов	да
successful_analyses	integer	body	Успешных анализов	да
performance	object	body	Метрики производительности	да
average_response_time	float	body	Среднее время ответа	да
gigachat_availability	float	body	Доступность GigaChat	да

5. Endpoint: Добавление истории в базу
Request

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
query	string	body	Текст User Story	да
answer	string	body	Анализ INVEST	да
is_golden	boolean	body	Золотая история	нет

score	integer	body	Оценка (0-6)	нет

Response (201 Created)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
id	integer	body	ID созданной записи	да
status	string	body	"created"	да
normalized_query	string	body	Нормализованный текст	да

Response (409 Conflict)

Название параметра	Тип данных	Находится в	Описание	Обязательность параметра
error	object	body	Информация об ошибке	да
code	string	body	"STORY_ALREADY_EXISTS"	да
message	string	body	"Такая история уже существует в базе"	да
existing_id	integer	body	ID существующей записи	да

Swagger UI интерфейс (визуальное представление)

Основные разделы:

1. Analysis

- **POST /analyze** - Анализ User Story
  - Parameters: user_story (required), similarity_threshold, use_cache
  - Responses: 200, 400, 429, 503

2. Improvement

- **POST /improve** - Улучшение User Story
  - Parameters: user_story (required), current_analysis, improvement_focus
  - Responses: 200, 400

3. Stories

- **GET /stories/search** - Поиск похожих историй
  - Query parameters: query (required), threshold, limit, only_golden
  - Response: 200
- **POST /stories** - Добавление истории в базу
  - Body: query (required), answer (required), is_golden, score
  - Responses: 201, 409

4. System

- **GET /statistics** - Статистика системы
  - Query parameter: period
  - Response: 200

Try it out примеры:

Для каждого эндпоинта в Swagger UI будут доступны:

- **Example values** с предзаполненными данными
- **Execute** кнопка для тестирования API
- **Responses** с примерами успешных ответов и ошибок
- **Schema** для просмотра структуры данных

Авторизация:

- API Key (X-API-Key header)
- Bearer Token (JWT)

Критерии приемки (AC)

AC-001: Анализ валидной User Story

Дано: Пользователь отправляет валидную User Story
Когда: Выполняется запрос анализа
Тогда: Система возвращает анализ с оценкой INVEST и рекомендациями

AC-002: Обработка невалидного формата

Дано: Пользователь отправляет текст в неправильном формате
Когда: Выполняется проверка валидности
Тогда: Система возвращает сообщение об ошибке с инструкцией

AC-003: Использование кэша

Дано: Пользователь отправляет уже анализировавшуюся User Story
Когда: Выполняется поиск в кэше
Тогда: Система возвращает результат из кэша

AC-004: Поиск с порогом схожести

Дано: В базе есть истории с разной степенью схожести
Когда: Пользователь отправляет запрос
Тогда: Система возвращает истории с схожестью ≥ установленного порога

AC-005: Улучшение User Story

Дано: Пользователь нажимает "Улучшить историю"
Когда: Выполняется запрос улучшения
Тогда: Система возвращает улучшенную версию с сохранением цели

AC-006: Предотвращение дубликатов

Дано: Пользователь пытается добавить существующую User Story
Когда: Выполняется проверка уникальности
Тогда: Система пропускает дубликат (в seed) или показывает сообщение

AC-007: Автоматическое добавление улучшенных историй

Дано: Пользователь улучшает историю через ИИ
Когда: Оценка улучшенной истории ≥ 4/6
Тогда: Система автоматически сохраняет ее как golden историю

Нефункциональные требования (NFR)

NFR-PERF-001: Время отклика

Требование: Анализ User Story выполняется за разумное время (< 10 секунд)
Реализация: Асинхронная обработка, кэширование

NFR-REL-001: Отказоустойчивость GigaChat

Требование: Система обрабатывает ошибки GigaChat API
Реализация: Try-catch блоки, повторные попытки

NFR-REL-002: Целостность данных

Требование: Данные сохраняются в PostgreSQL
Реализация: ACID транзакции через asyncpg

NFR-SEC-001: Базовая аутентификация

Требование: Доступ только через авторизованного Telegram бота
Реализация: Telegram Bot Token