

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 ($\geq 85\%$) 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 ($\geq 95\%$), обычное сходство ($\geq 85\%$)
	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

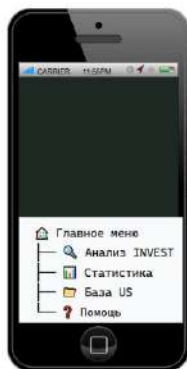
Заголовок	Многовариантный экспорт результатов
Акторы	руководитель проекта, аналитик
Предусловие	Имеются результаты анализа или улучшенная история
Ограничения	Поддержка форматов: TXT, CSV
	Экспорт как анализа, так и улучшенных историй
	Автоматическое именование файлов
Триггер	Пользователь нажимает кнопку "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



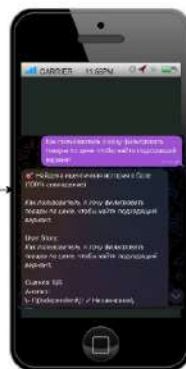
анализ US



пользователь выбирает историю



система находит похожую историю в базе



нажимаем кнопку экспорт



выходит выбор формата для экспорта



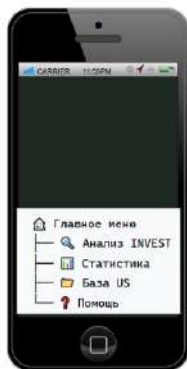
после выбора формата txt происходит автоматический экспорт



экспортированный файл можно сразу посмотреть



главное меню



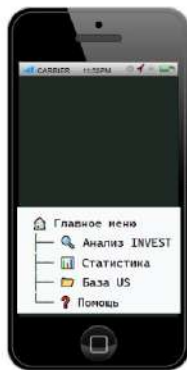
база US



база US



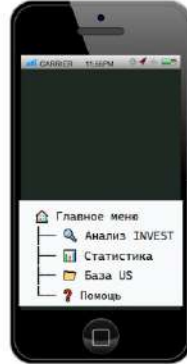
главное меню



статистика



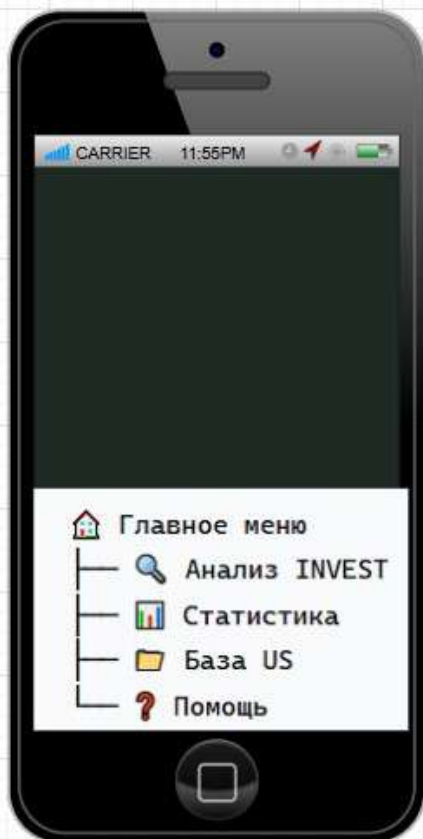
главное меню



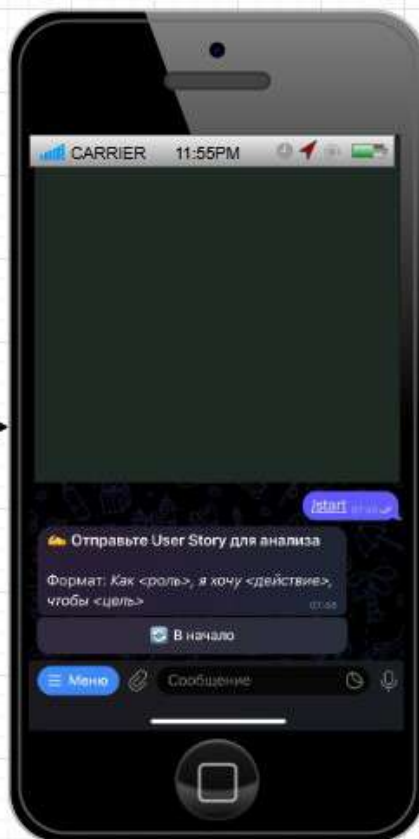
помощь



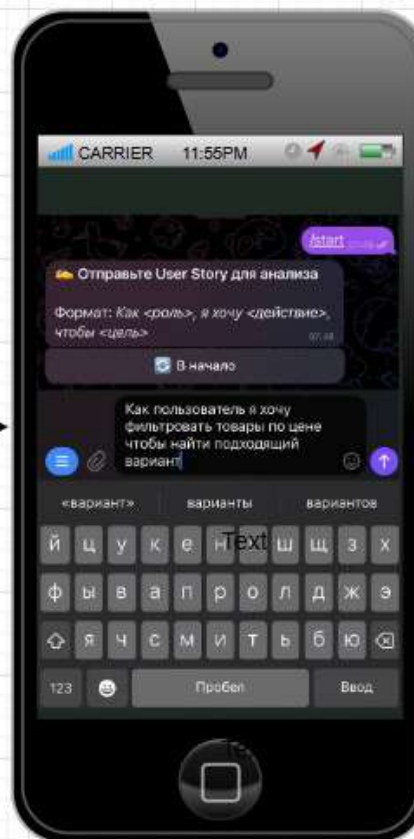
главное меню



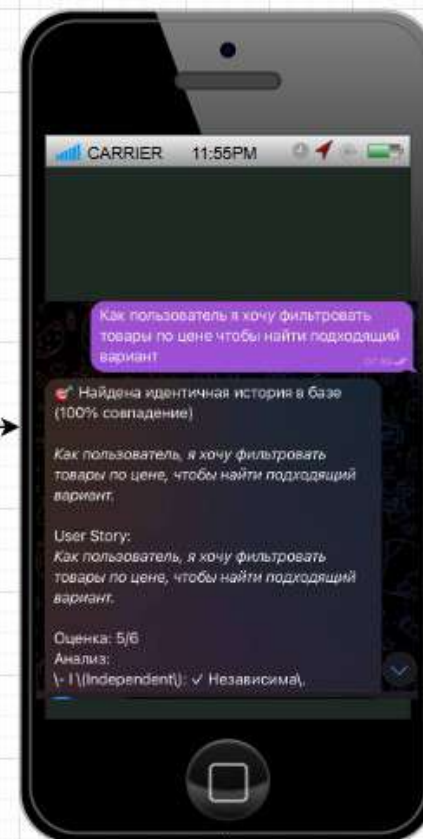
анализ US



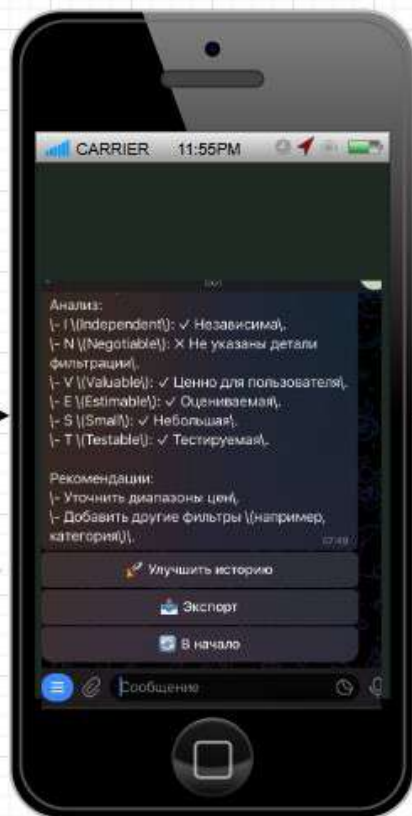
пользователь вбивает историю



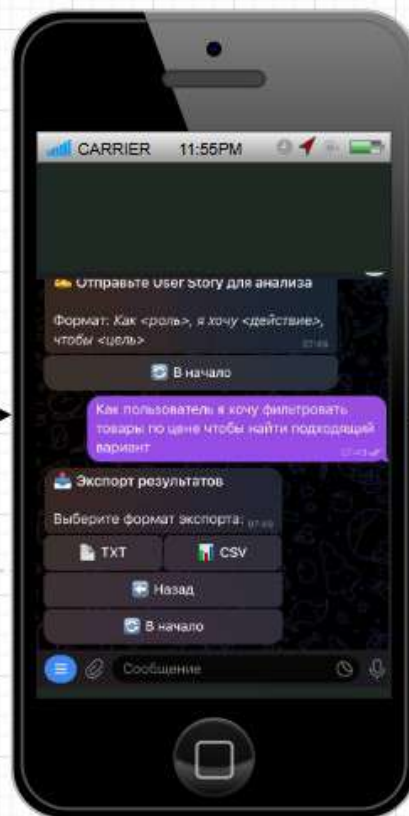
система находит похожую историю в базе



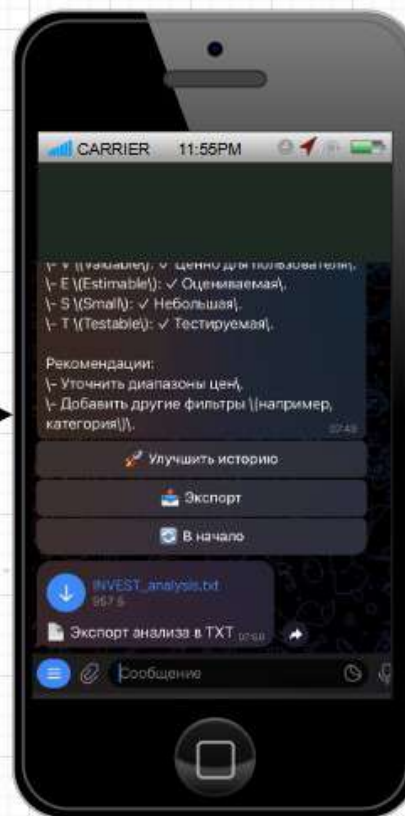
нажимаем кнопку экспорт



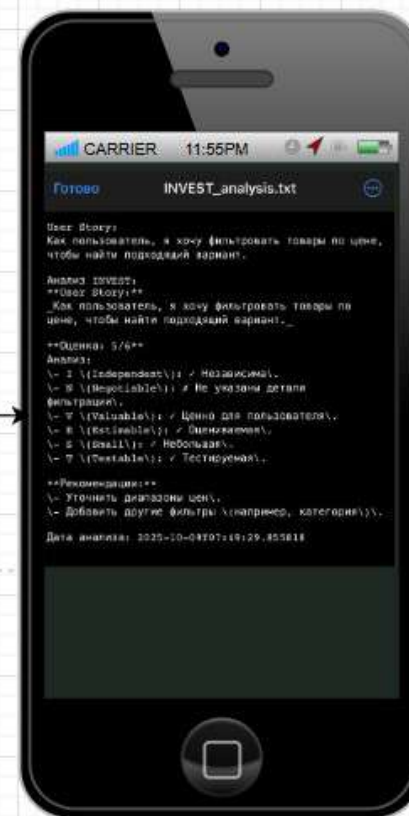
выходит выбор формата для экспорта



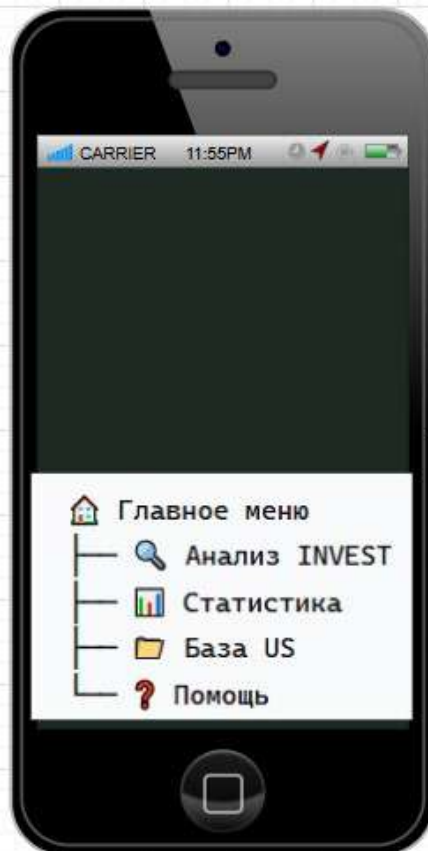
после выбора формата txt происходит автоматический экспорт



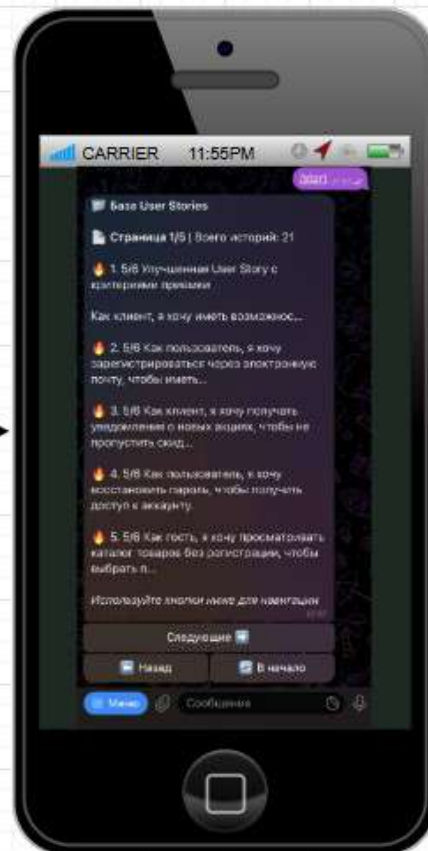
экспортированный файл можно сразу посмотреть



главное меню



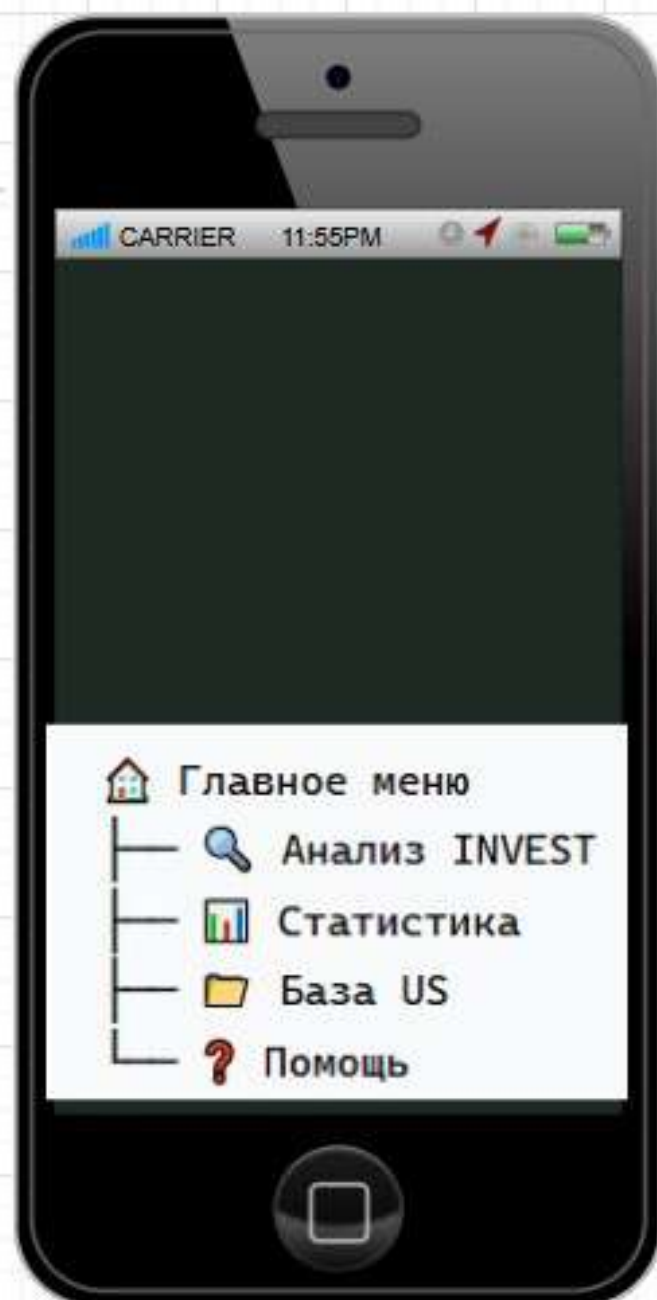
база US



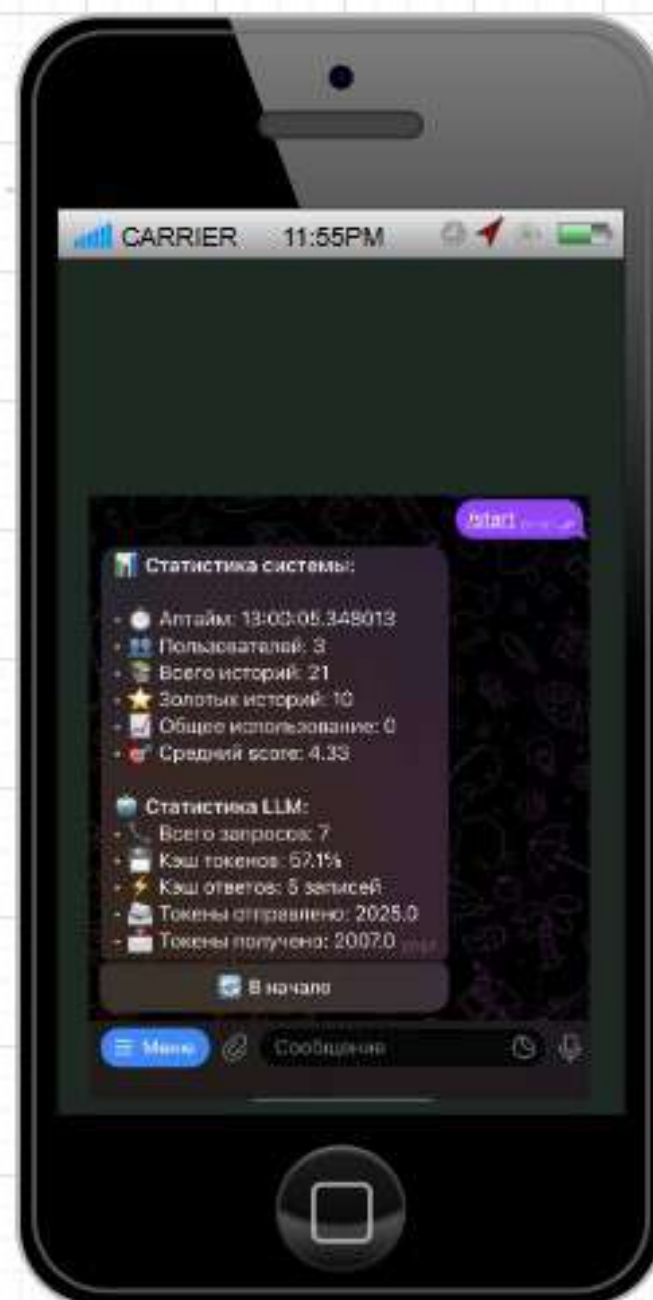
база US



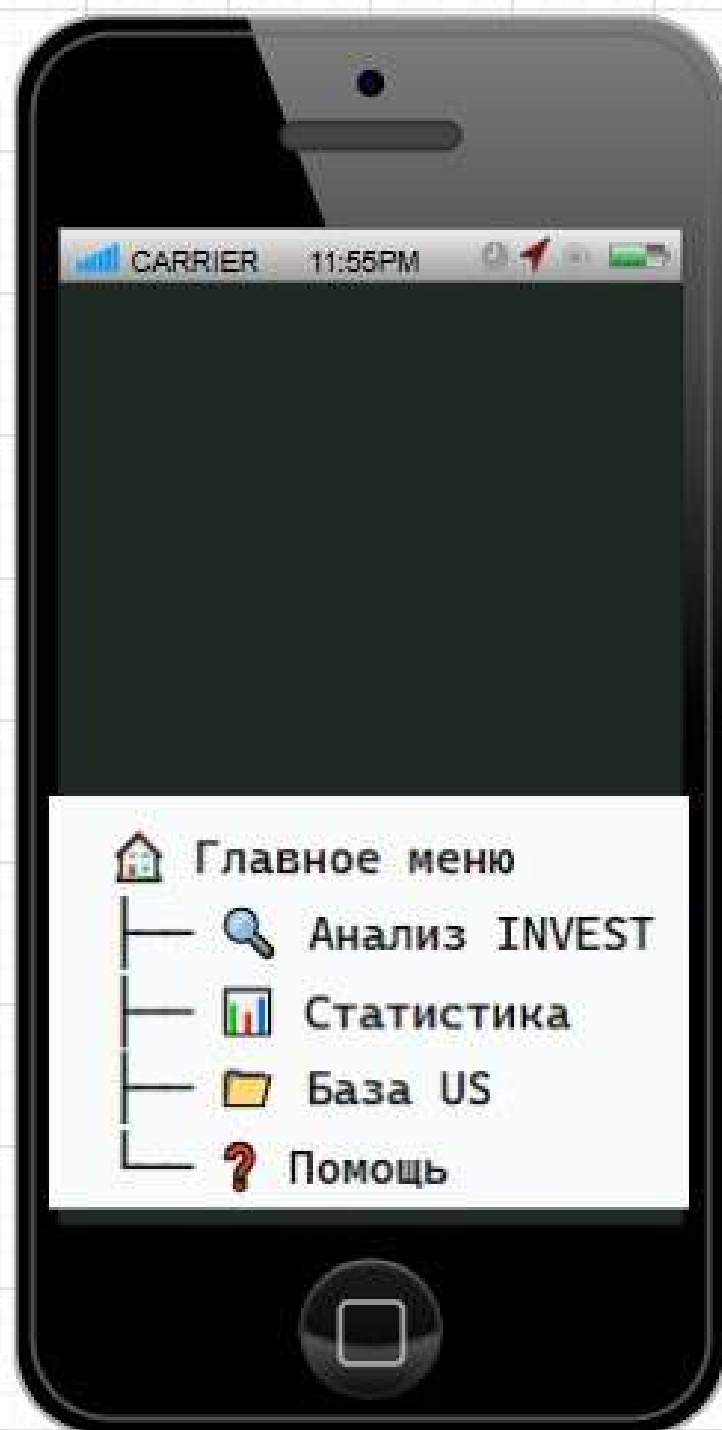
главное меню



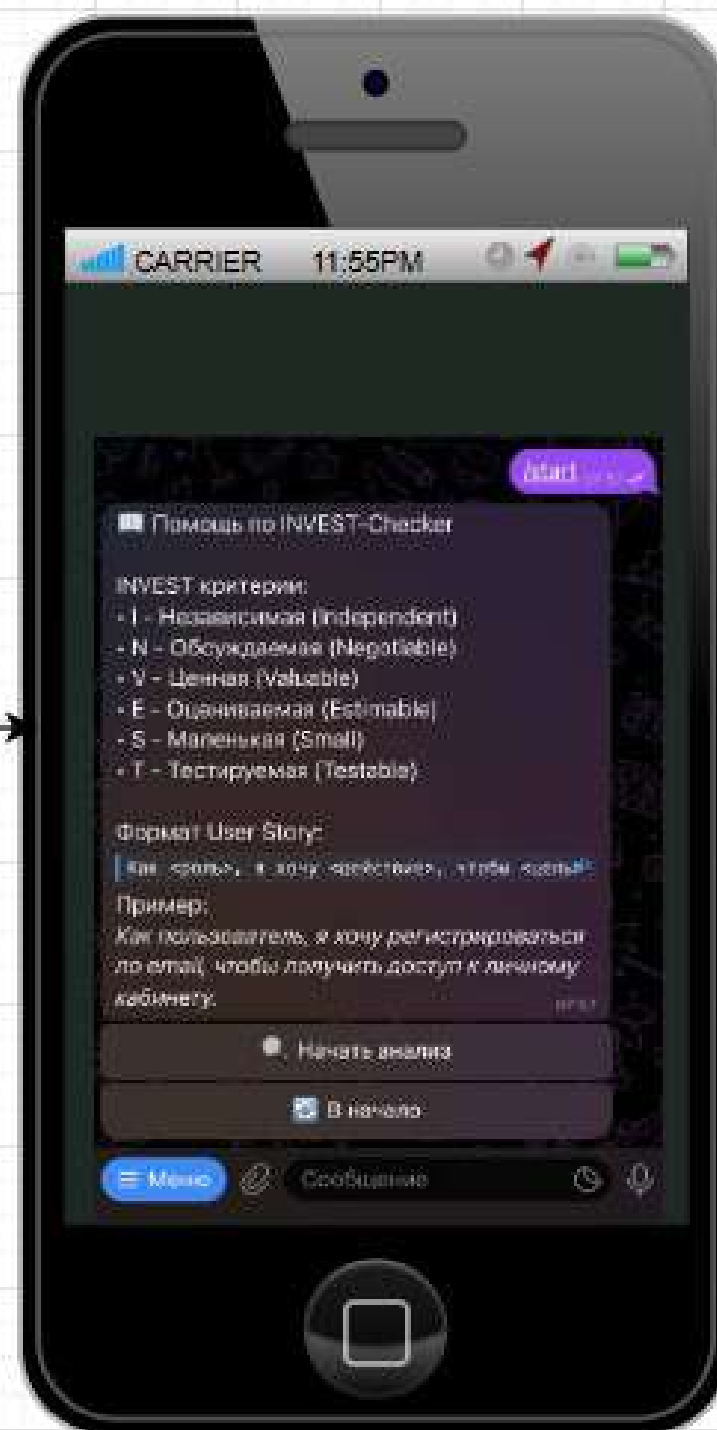
статистика

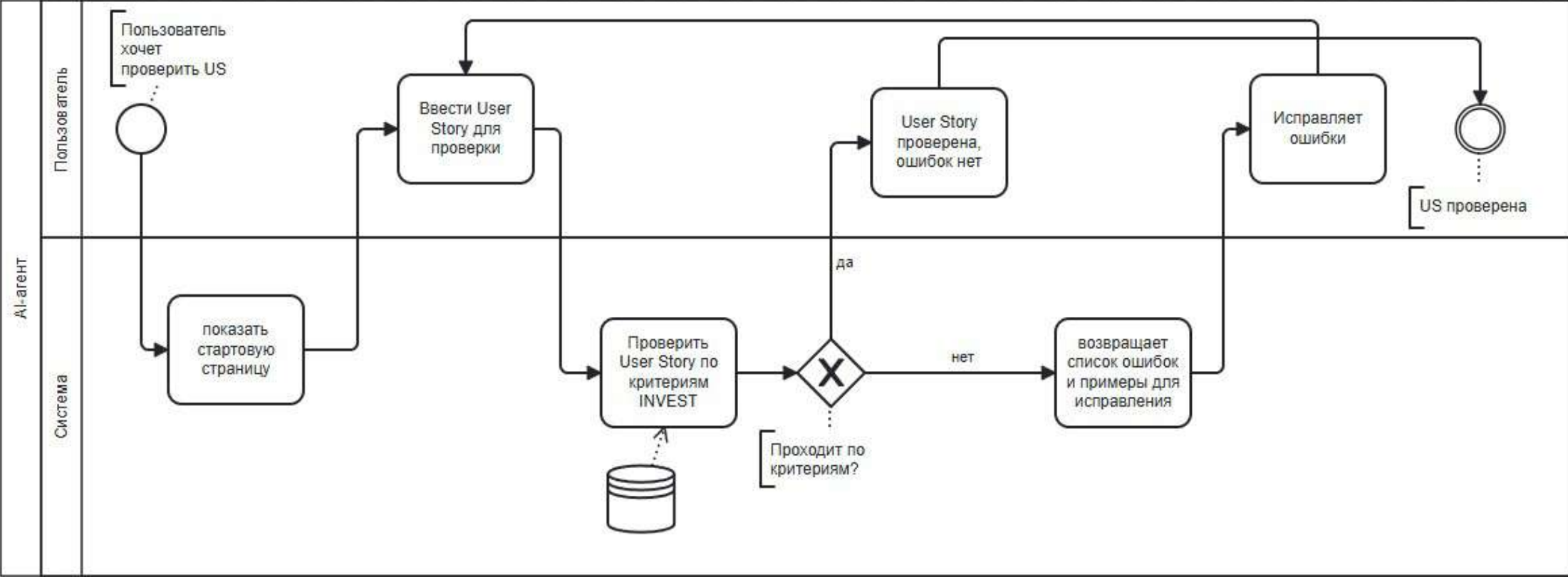


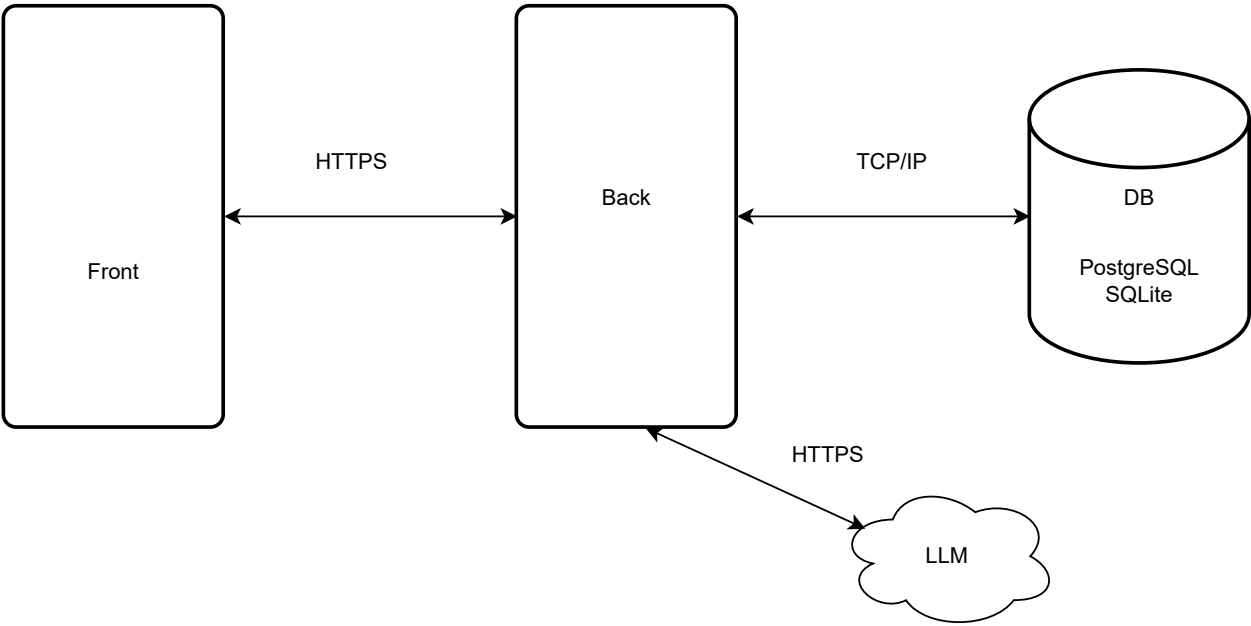
главное меню



помощь







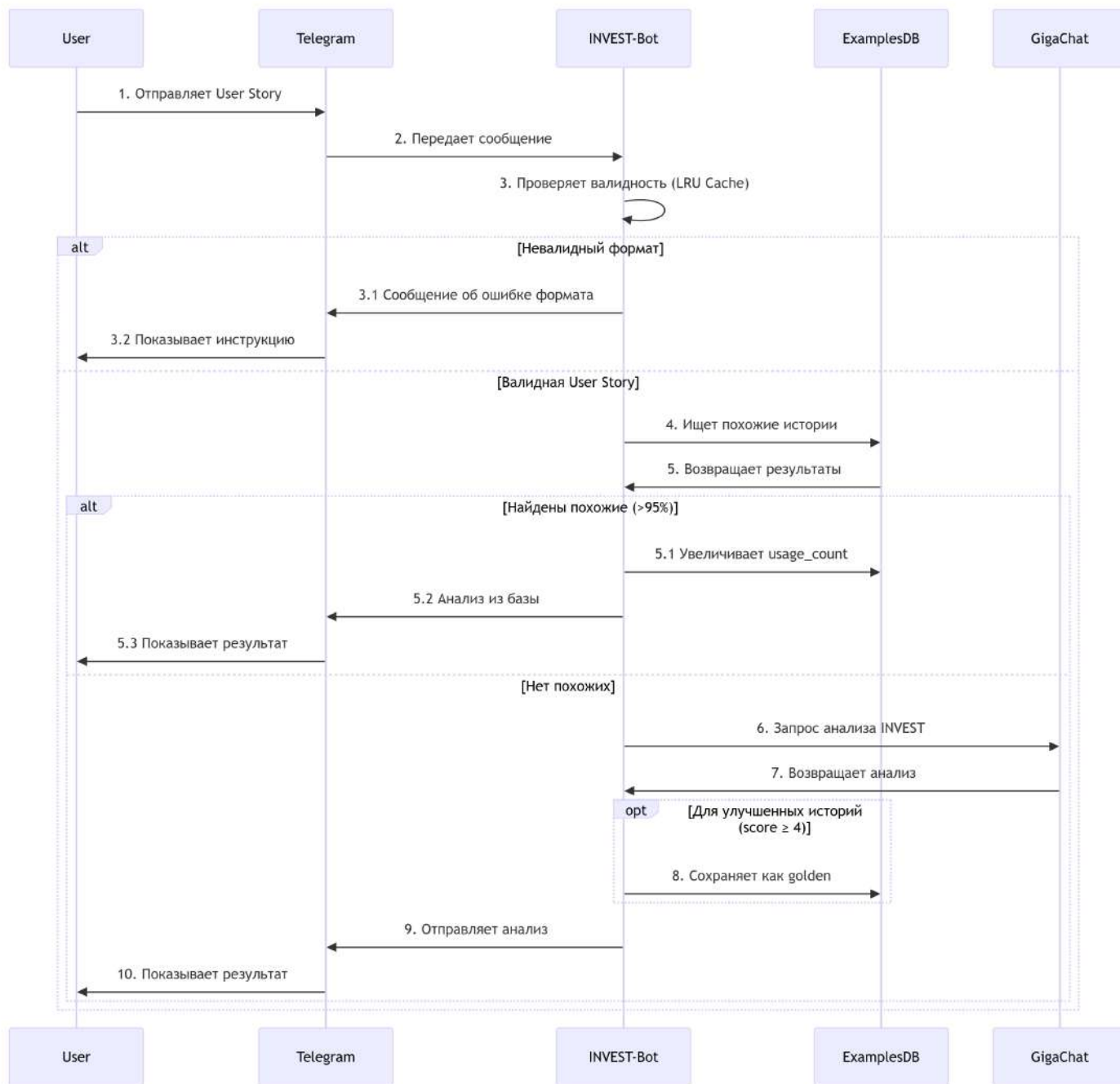
Модель данных

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

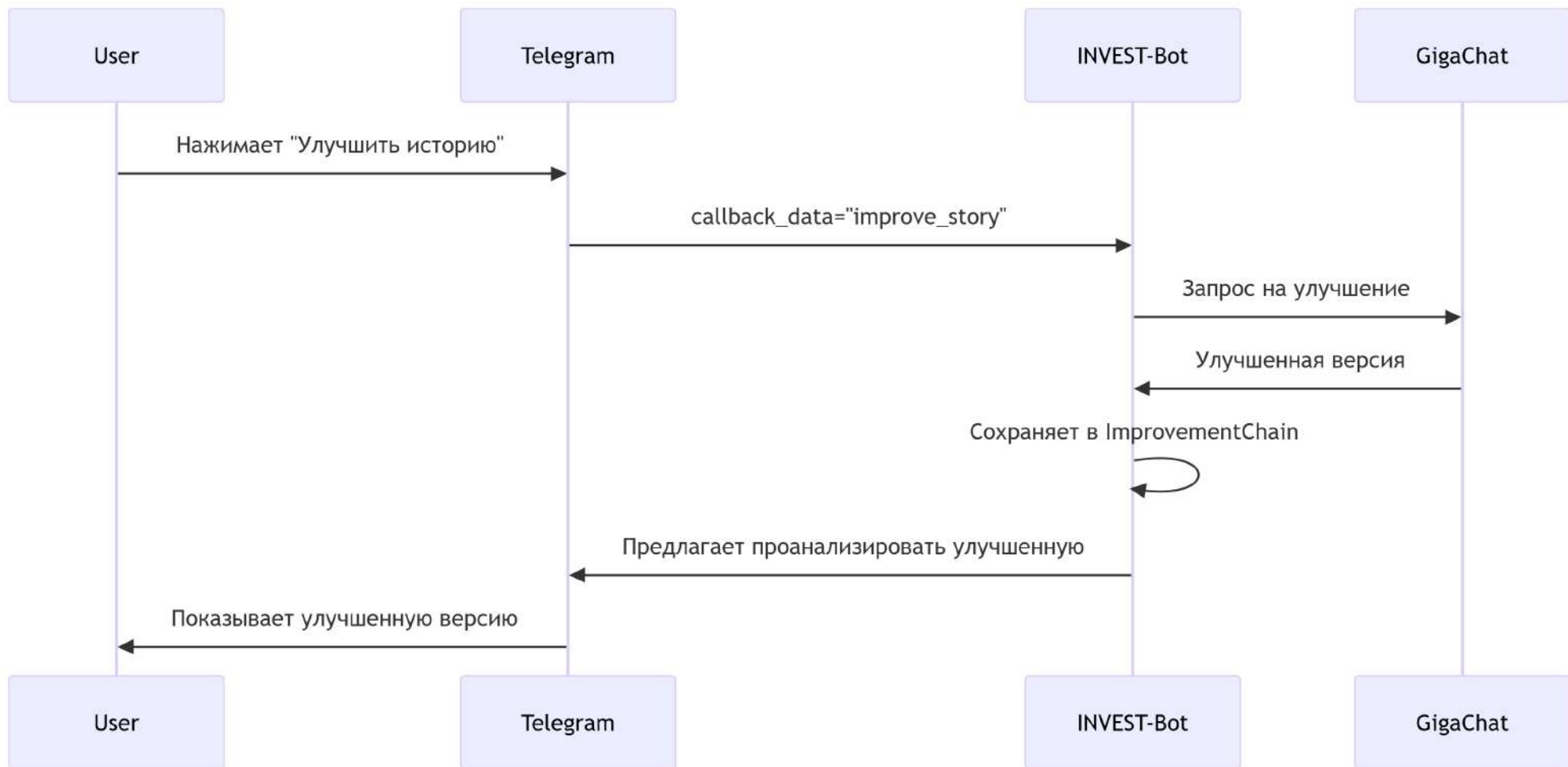
ERD диаграмма

user_stories
id (serial, primary key)
normalized_query (text)
answer (text)
is_golden(boolean)
score(integer)
usage_count(integer)
created_at(timestamp)
updated_at(timestamp)



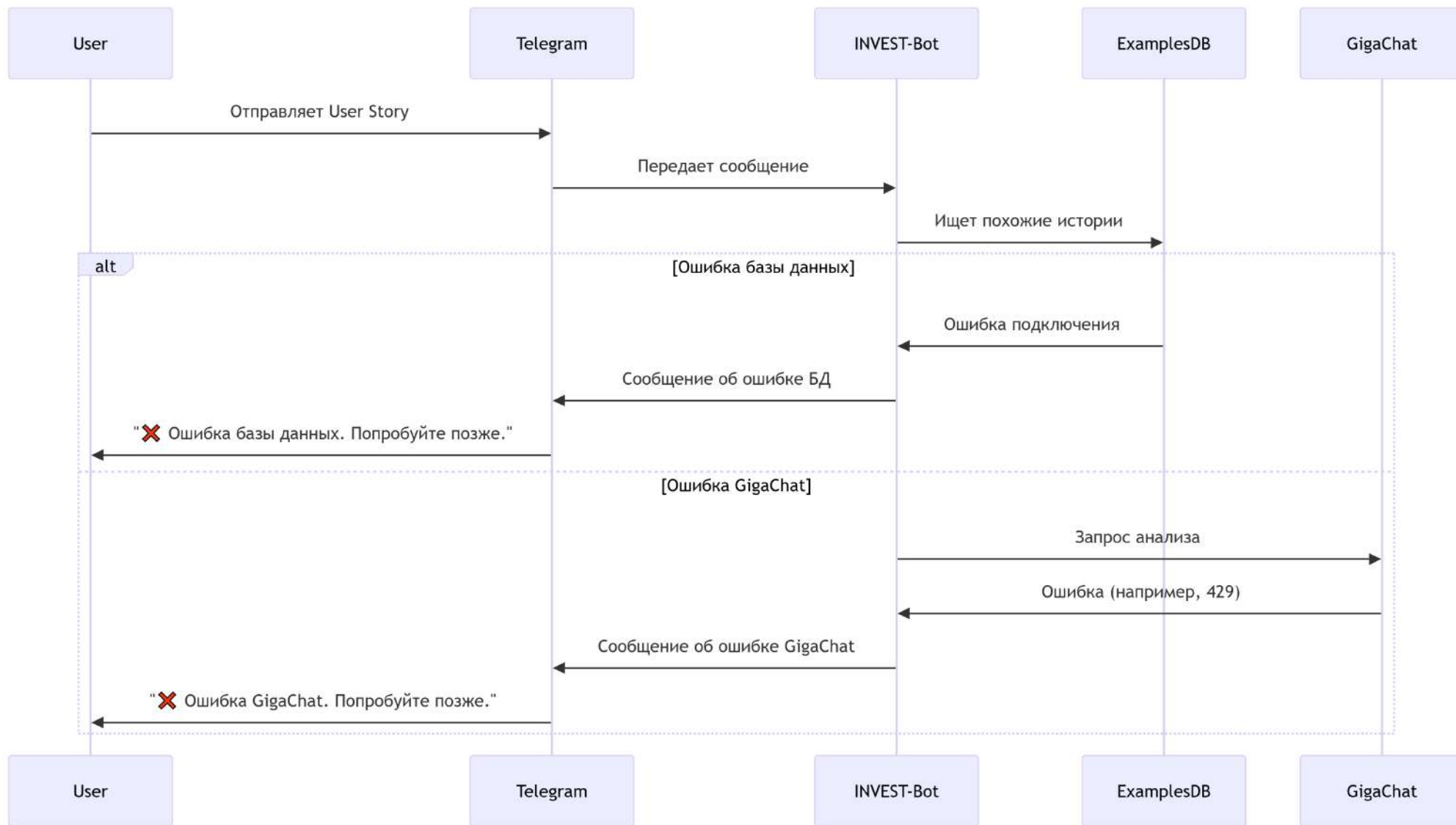
Основной сценарий анализа

№	Описание	Компонент
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
3.2	Пользователь получает инструкцию по формату	Telegram → User
4	Бот ищет похожие истории в базе данных	INVEST-Bot → ExamplesDB
5	База данных возвращает результаты поиска	ExamplesDB → INVEST-Bot
5.1	Если найдены похожие истории (>95%), бот увеличивает счетчик использования	INVEST-Bot → ExamplesDB
5.2	Бот отправляет анализ из базы данных	INVEST-Bot → Telegram
5.3	Пользователь получает результат анализа	Telegram → User
6	Если похожих историй не найдено, бот обращается к GigaChat API	INVEST-Bot → GigaChat
7	GigaChat API возвращает анализ User Story	GigaChat → INVEST-Bot
8	Для улучшенных историй с оценкой ≥ 4 бот сохраняет в базу как golden	INVEST-Bot → ExamplesDB
9	Бот отправляет форматированный анализ	INVEST-Bot → Telegram
10	Пользователь получает результат анализа	Telegram → User



Сценарий улучшения истории

№	Описание	Компонент
1	Пользователь нажимает кнопку "Улучшить историю"	User → Telegram
2	Telegram передает callback запрос боту	Telegram → INVEST-Bot
3	Бот отправляет запрос на улучшение истории в GigaChat	INVEST-Bot → GigaChat
4	GigaChat API возвращает улучшенную версию User Story	GigaChat → INVEST-Bot
5	Бот сохраняет улучшенную версию в цепочку улучшений	INVEST-Bot
6	Бот отправляет улучшенную версию пользователю	INVEST-Bot → Telegram
7	Пользователь видит улучшенную версию истории	Telegram → User



Сценарий обработки ошибок

№	Описание	Компонент
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 → ExamplesDB
3.1	Ошибка базы данных: сервис недоступен	ExamplesDB → INVEST-Bot
3.2	Бот отправляет сообщение об ошибке базы данных	INVEST-Bot → Telegram
3.3	Пользователь получает уведомление об ошибке	Telegram → User
4	Ошибка GigaChat: превышен лимит запросов	GigaChat → INVEST-Bot
4.1	Бот отправляет сообщение об ошибке GigaChat	INVEST-Bot → Telegram
4.2	Пользователь получает уведомление об ошибке	Telegram → User

Telegram Bot API Documentation

1. Команды бота

Основные команды

Команда	Описание	Пример использования
<code>/start</code>	Начало работы, показывает приветственное сообщение и главное меню	<code>/start</code>
<code>/help</code>	Показывает справку по использованию бота и критериям INVEST	<code>/help</code>
<code>/stats</code>	Показывает статистику бота и базы данных	<code>/stats</code>

Текстовые сообщения





Тип сообщения	Формат	Пример	Ответ бота
User Story	"Как <роль>, я хочу <действие>, чтобы <цель> "	"Как пользователь, я хочу регистрироваться по email, чтобы получить доступ к личному кабинету"	Анализ INVEST с оценкой и рекомендациями
Альтернативный формат	Любой текст, похожий на User Story	"Мне нужно видеть статистику продаж для принятия решений"	Анализ с предупреждением о нестандартном формате

2. Callback действия (инлайн-кнопки)




Клавиатура главного меню

Кнопка	Callback data	Действие
 Анализ INVEST	analyze_invest	Переход к вводу User Story
 Статистика	stats	Показ статистики системы
 База US	show_database	Просмотр базы User Stories
 Помощь	help	Показ справки

Клавиатура результатов анализа

Кнопка	Callback data	Условия показа
 Улучшить историю	improve_story	Всегда
 Показать историю улучшений	show_improvement_history	Если есть цепочка улучшений
 Экспорт	export	Всегда
 Добавить в базу	add_to_db	Только для историй с оценкой $\geq 5/6$

Клавиатура улучшенной истории

Кнопка	Callback data	Действие
 Проанализировать улучшенную	analyze_improved	Анализ улучшенной версии
 Улучшить заново	improve_again	Дополнительное улучшение
 Экспорт улучшенной	export_improved	Экспорт улучшенной версии

Навигационные кнопки

Кнопка	Callback data	Действие
 Назад	back	Возврат к предыдущему состоянию
 В начало	restart	Возврат в главное меню

3. Форматы ответов

Успешный анализ User Story

```
text
**User Story:**
_Как пользователь, я хочу зарегистрироваться по email, чтобы получить доступ к личному кабинету_

**Оценка: 5/6**
```

****Анализ:****

- I (Independent): ✔ Независима
- N (Negotiable): ✗ Не указано, что открыто для обсуждения
- V (Valuable): ✔ Ценность для пользователя очевидна
- E (Estimable): ✔ Можно оценить трудозатраты
- S (Small): ✔ Небольшая по объему
- T (Testable): ✔ Можно проверить функционал

****Рекомендации:****


- Уточнить возможность обсуждения деталей реализации
- Добавить критерии успеха для регистрации

Сообщения об ошибках

Тип ошибки	Сообщение	Причина
Невалидный формат	"✗ Это не похоже на User Story. Правильный формат: <i>Как <роль>, я хочу <действие>, чтобы <цель></i> "	Неправильная структура сообщения
Ошибка GigaChat	"✗ Ошибка при анализе через GigaChat. Проверьте подключение к интернету."	Проблемы с API GigaChat
Ошибка базы данных	"✗ Ошибка базы данных. Попробуйте позже."	Проблемы с подключением к PostgreSQL

Статистика системы

text

 Статистика системы:

- 🕒 Аптайм: 2 days, 5:30:15
- 👤 Пользователей: 150
- 📖 Всего историй: 1,247
- ★ Золотых историй: 89
- 📈 Общее использование: 2,845
- 🎯 Средний score: 4.2
- 💾 Эффективность кэша: 45.3%

Критерии приемки (АС)

АС-001: Анализ валидной User Story

Дано: Пользователь отправляет валидную User Story

Когда: Выполняется запрос анализа

Тогда: Система возвращает анализ с оценкой INVEST и рекомендациями

АС-002: Обработка невалидного формата

Дано: Пользователь отправляет текст в неправильном формате

Когда: Выполняется проверка валидности

Тогда: Система возвращает сообщение об ошибке с инструкцией

АС-003: Использование кэша

Дано: Пользователь отправляет уже анализировавшуюся User Story

Когда: Выполняется поиск в кэше

Тогда: Система возвращает результат из кэша

АС-004: Поиск с порогом схожести

Дано: В базе есть истории с разной степенью схожести

Когда: Пользователь отправляет запрос

Тогда: Система возвращает истории с схожестью \geq установленного порога

АС-005: Улучшение User Story

Дано: Пользователь нажимает "Улучшить историю"

Когда: Выполняется запрос улучшения

Тогда: Система возвращает улучшенную версию с сохранением цели

АС-006: Предотвращение дубликатов

Дано: Пользователь пытается добавить существующую User Story

Когда: Выполняется проверка уникальности

Тогда: Система пропускает дубликат (в seed) или показывает сообщение

АС-007: Автоматическое добавление улучшенных историй

Дано: Пользователь улучшает историю через ИИ

Когда: Оценка улучшенной истории $\geq 4/6$

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 asynprg

NFR-SEC-001: Базовая аутентификация

Требование: Доступ только через авторизованного Telegram бота

Реализация: Telegram Bot Token