📘 Архитектура: Telegram Bot + WebApp Wallet (Amanita)

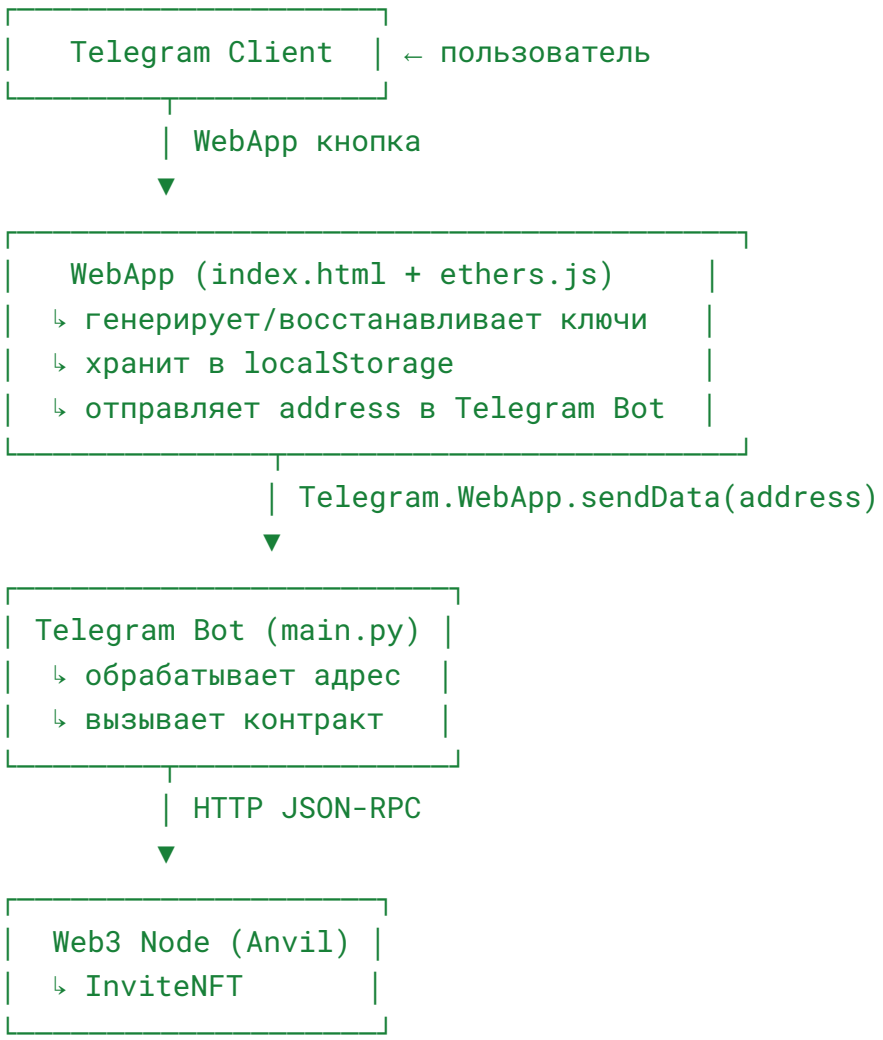
Версия: `v0.1 MVP`

Автор: CTO

Цель: Упростить пользовательский онбординг в Web3 без Metamask, через Telegram-бот + WebApp

---

🧱 1. Общая архитектура



⚙️ 2. Компоненты

🔹 Telegram Bot (`bot/main.py`)

- Написан на `python-telegram-bot`
- Обрабатывает `/start`, `inviteCode`, `web_app_data`
- Отвечает за валидацию, вызов смарт-контракта
- Использует:
    - `web3.py`
    - приватный ключ продавца (SELLER_ROLE)

🔹 WebApp Wallet (`webapp/`)

- Одностраничный HTML + JS
- Работает в Telegram WebView
- Подключает:
    - `ethers.js`
    - `Telegram.WebApp API`
- Реализует:
    - генерацию кошелька (`create`)
    - восстановление по сид-фразе (`restore`)
    - хранение в `localStorage`
    - отправку `address` в бота

🔹 Web3 Node (`Hardhat`, `Anvil`)

- Слушает на `http://127.0.0.1:8545`
- Выполняет транзакции
- Контракт `InviteNFT` задеплоен заранее

🚀 3. Процесс запуска на Mac

🛠️ Предполагаемая структура проекта:

```
AMANITA/
├── bot/                # Python-бот
│   └── main.py
├── webapp/             # Telegram WebApp
│   ├── index.html
│   ├── main.js
│   └── styles.css
├── contracts/
│   └── InviteNFT.sol
├── .env
└── README.md
```

🧪 Шаги запуска в dev-среде (3 терминала)

✅ Терминал 1: Web3-нода

```
npx hardhat node
```

✅ Терминал 2: HTTP WebApp

```
cd amanita/webapp
python3 -m http.server 8080
```

🔗 Доступно по `http://localhost:8080/index.html`

✅ Терминал 3: Telegram-бот

```
cd amanita/bot
python3 main.py
```

Файл `.env`:

```
BOT_TOKEN=123456:ABCD...
SELLER_PRIVATE_KEY=0xabc...
WEB3_PROVIDER_URL=http://127.0.0.1:8545
WEBAPP_URL=http://localhost:8080/index.html
```

🔗 4. Взаимодействие бота с WebApp

🔹 Вызов WebApp из бота

```
from telegram import InlineKeyboardButton, InlineKeyboardMarkup

keyboard = InlineKeyboardMarkup([
  [InlineKeyboardButton(
    text="🔐 Создать кошелёк",
    web_app={"url": WEBAPP_URL}
  )]
])

bot.send_message(chat_id, text="Создай кошелёк",
reply_markup=keyboard)
```

🔹 Получение адреса от WebApp

```
@application.message_handler(content_types=['web_app_data'])
def receive_wallet(update, context):
    wallet_address = update.message.web_app_data.data
    user_id = update.message.from_user.id
    # Сохраняем или привязываем адрес
    # Вызываем activateAndMintInvites(...)
```

🧩 5. Как WebApp решает задачи

✅ Генерация кошелька

- `ethers.Wallet.createRandom()`
- Сохраняется: `mnemonic`, `address`, `privateKey` в `localStorage`
- Отправка `address` в бот: `Telegram.WebApp.sendData(address)`

✅ Восстановление

- Ввод сид-фразы
- `ethers.Wallet.fromPhrase(mnemonic)`
- Повторная привязка адреса

📡 6. WebApp = альтернатива Metamask

Возможность	Metamask	Amanita WebApp
Хранение ключей	В расширении	В localStorage
Сеть	Выбор вручную	Указана жёстко
Подпись транзакций	Через UI	Не реализована (пока)
Вызов контракта	С клиента	С бэкенда через bot
Резервное восстановление	Seed-фраза	Seed-фраза

➡ В MVP — только `address` хранится, подпись вне клиента (через SELLER).

⚠️ 7. Ограничения

- WebApp работает **только в WebView Telegram**

- `localStorage` не синхронизируется между устройствами
- Потеря сид-фразы = потеря доступа
- Пока нет полноценной подписи/отправки транзакций от пользователя

📌 8. Резюме

Компонент	Ответственность
Telegram Bot	Инвайты, управление адресами, контракты
WebApp Wallet	Генерация и восстановление адреса
Web3 Node	Исполнение `activateAndMintInvites`