**Последний 111. С Хелиус PR и Трейдинг вью 29.09**

import React, { useMemo, useRef, useState, useEffect, useContext, createContext } from "react";

import { LineChart, Line, Tooltip, ResponsiveContainer, XAxis, YAxis, ReferenceArea, CartesianGrid } from "recharts";

// ✦ Alpha2 Super Parser — UI 18.09 (+ добавления 29.09) — обновлённый супер‑интерфейс

// ВАЖНО: структуру не ломал, только добавил/перенастроил то, что ты попросил.

// Новое:

// 1) Сигналы: в нижнем большом окне теперь список ОЧИЩЕННЫХ хайп‑слов (слово, кто сказал [ссылка], дата/время, источник TG/Twitter).

// 2) Аккаунты: загрузка Tw/TG аккаунтов, загрузка правил, выбор интервала парсинга (5/15/30/60 мин), менеджер API‑ключей (добавлены 5 новых сервисов, сохранение+индикатор/логи).

// 3) Убран блок ключей со всех страниц — ключи доступны ТОЛЬКО на вкладке «Аккаунты».

// 4) Tradeview: режим наложений BUY/SELL/WHALE/HYPE остаётся «нарисованным у нас», добавлен диапазон кита (от…до SOL).

// 5) Helius: старт/стоп, список пройденных минтов (строгие проверки — демо), с инфо и кнопкой «Скачать минт».

// 6) CEX Radar: старт/стоп, поиск, таблица первых листингов и кнопка «Открыть» (ссылка на Birdeye).

// 7) Учиться: загрузка книг, глобальных правил, тонких настроек, графиков/фото.

// ───────── helpers

const cls = (...a) => a.filter(Boolean).join(" ");

const pretty = (v) => (v===0?0:(v|| "—"));

const download = (name, text, mime) => { const blob = new Blob([text], {type: mime||"text/plain"}); const url = URL.createObjectURL(blob); const a=document.createElement("a"); a.href=url; a.download=name; a.click(); URL.revokeObjectURL(url); };

const loadLS = (k, d) => { try{ const v = localStorage.getItem(k); return v? JSON.parse(v): d; }catch{return d;} };

const saveLS = (k, v) => { try{ localStorage.setItem(k, JSON.stringify(v)); }catch{} };

const uid = () => Math.random().toString(36).slice(2,9);

// утилита для уникализации слов (для тестов и панели «очищенные слова»)

function uniqueHypeWords(rows){

const map = new Map();

rows.forEach(r=>{

const key = (r.word||"").trim().toUpperCase();

if(!key) return;

const cur = map.get(key);

if(!cur || (r.detectedAt||"") > (cur.ts||"")){

map.set(key, { word:r.word, author:r.author, link:r.link, ts:r.detectedAt, src:r.source });

}

});

return Array.from(map.values()).sort((a,b)=> (b.ts||"").localeCompare(a.ts||""));

}

// ───────── mini UI kit

const Card = ({className, ...p}) => <div className={cls("rounded-2xl border border-zinc-800 bg-zinc-950", className)} {...p}/>;

const CardHeader = ({className, right, children, ...p}) => <div className={cls("px-4 py-3 border-b border-zinc-800 flex items-center justify-between", className)} {...p}><div>{children}</div>{right}</div>;

const CardTitle = ({className, ...p}) => <div className={cls("text-sm font-semibold", className)} {...p}/>;

const CardContent = ({className, ...p}) => <div className={cls("p-4", className)} {...p}/>;

const Button = ({variant="solid", size="md", className, ...p}) => (

<button

className={cls(

"inline-flex items-center justify-center rounded-lg transition-colors",

size==="sm"?"h-8 px-3 text-sm":"h-10 px-4 text-sm",

variant==="ghost"?"bg-transparent border border-zinc-700 hover:bg-zinc-900":

variant==="danger"?"bg-red-600 hover:bg-red-500":

variant==="success"?"bg-emerald-600 hover:bg-emerald-500":"bg-zinc-800 hover:bg-zinc-700",

className

)}

{...p}

/>

);

const Input = React.forwardRef((props, ref)=> (

<input ref={ref} {...props} className={cls("h-10 w-full rounded-lg border border-zinc-800 bg-zinc-900 px-3 text-sm outline-none focus:ring-1 focus:ring-zinc-600", props.className)}/>

));

const NumberInput = (props)=> <Input type="number" {...props}/>;

const Textarea = React.forwardRef((props, ref)=> (

<textarea ref={ref} {...props} className={cls("min-h-[120px] w-full rounded-lg border border-zinc-800 bg-zinc-900 px-3 py-2 text-sm outline-none focus:ring-1 focus:ring-zinc-600", props.className)}/>

));

const Label = (p)=> <label className={cls("text-xs text-zinc-400", p.className)} {...p}/>;

const Led = ({ok, className}) => <span title={ok?"активен":"ошибка"} className={cls("inline-block w-2.5 h-2.5 rounded-full", ok?"bg-emerald-500":"bg-red-600", className)} />

const Badge = ({children, tone="default", className}) => (

<span className={cls("inline-flex items-center rounded-md px-2 py-0.5 text-[11px] font-medium", tone==="ok"?"bg-emerald-700 text-white":tone==="warn"?"bg-amber-700 text-white":tone==="bad"?"bg-red-700 text-white":"bg-zinc-800 text-white", className)}>{children}</span>

);

// Table (возвращено; использовалось в предыдущей версии)

const Table = ({columns, data, onRowClick, emptyText}) => (

<div className="w-full overflow-auto rounded-xl border border-zinc-800">

<table className="w-full text-left text-sm text-white">

<thead className="bg-zinc-900/60"><tr>{columns.map(c=> <th key={c.key} className="px-4 py-3 font-medium whitespace-nowrap">{c.title}</th>)}</tr></thead>

<tbody>

{(!data||data.length===0)&&(<tr><td colSpan={columns.length} className="px-4 py-10 text-center text-zinc-500">{emptyText||"Нет данных"}</td></tr>)}

{data?.map((row,i)=> (

<tr key={row.id||i} className="border-t border-zinc-800/80 hover:bg-zinc-900/40" onClick={()=>onRowClick?.(row)}>

{columns.map(c=> <td key={c.key} className="px-4 py-3 align-top">{c.render? c.render(row): row[c.key]}</td>)}

</tr>

))}

</tbody>

</table>

</div>

);

// ───────── demo data

const seedSignals = [

{ id:"1", word:"Dark MAGA", isOG:true, type:"слово", detectedAt:"2025-09-17 11:24", source:"Twitter", author:"@elonmusk", link:"https://x.com/search?q=Dark%20MAGA", image:"", tweetCount:842, communitySize:125000, nameChanges:0, spamScore:0.08, devTeam:null, communityLink:"https://t.me/example", contract:"", chain:"Solana", safety:{noMint:true,burnLP:true,blacklist:false}, summary:"Трендовая фраза — высокий шанс клонов. OG подтверждён." },

{ id:"2", word:"STEVE", isOG:false, type:"токен", detectedAt:"2025-09-17 12:03", source:"Telegram", author:"@memelabs", link:"https://t.me/memelabs/42", image:"", tweetCount:120, communitySize:21000, nameChanges:2, spamScore:0.42, devTeam:"unknown", communityLink:"https://x.com/search?q=STEVE", contract:"So1xxxx...abcd", chain:"Solana", safety:{noMint:false,burnLP:false,blacklist:true}, summary:"Много клонов, высокий спам-индекс. Вероятен быстрый памп/дамп." }

];

function genSeries(n=180, base=1){ const out=[]; let v=base; for(let i=0;i<n;i++){ v = Math.max(0.1, v + (Math.random()-0.5)\*0.03); out.push({i, v: +v.toFixed(4)});} return out; }

export default function App(){

const [tab, setTab] = useState("signals");

// глобальные состояния

const [running, setRunning] = useState(false);

const [logs, setLogs] = useState([]);

const log = (msg)=> setLogs(x=> [...x, {ts:new Date().toISOString(), msg}]);

// ключи — теперь только на «Аккаунты»

const [keys,setKeys]=useState(loadLS("keys", {

groq:"", apify:"", twitterBearer:"", helius:"", quicknode:"", pumpfun:"", nansen:"",

solscan:"", birdeye:"", gmgn:"", solsniffer:"", rugcheck:""

}));

const [keyStatus,setKeyStatus]=useState(loadLS("keyStatus", {}));

useEffect(()=> saveLS("keys", keys), [keys]);

useEffect(()=> saveLS("keyStatus", keyStatus), [keyStatus]);

const saveKey=(k)=>{

const ok = !!(keys[k] && String(keys[k]).trim());

setKeyStatus(s=>({...s, [k]: ok?"ok":"error"}));

if(!ok) log(`[keys] Пустой ключ: ${k}`); else log(`[keys] Сохранён ключ: ${k}`);

};

// сигналы

const [rows, setRows] = useState(seedSignals);

const [query, setQuery] = useState("");

const [onlyOG, setOnlyOG] = useState(true);

const [cadenceMin, setCadenceMin] = useState(5); // управляется вкладкой «Аккаунты»

// алерты

const [alerts, setAlerts] = useState(loadLS("alerts", []));

const unread = alerts.filter(a=>!a.read).length;

useEffect(()=> saveLS("alerts", alerts), [alerts]);

// статусы

const [statusLine, setStatusLine] = useState({helius:"—", signals:"—", filters:"#sol #new #pump", ai:"вход малым, SL 20%"});

useEffect(()=>{

const i = setInterval(()=>{

const a = ["swap +$"+(1000+Math.random()\*9000).toFixed(0), `liq ${Math.random()>0.5?"+":"-"}${(5+Math.random()\*20).toFixed(0)}%/10m`];

const s = ["Pump↑","Spread↓","Vol↑"][Math.floor(Math.random()\*3)];

setStatusLine(v=>({...v, helius: a.join(", "), signals: s }));

}, 3000);

return ()=>clearInterval(i);

},[]);

// Helius AR (без изменений по требованиям)

const [arText, setArText] = useState("");

const [arRunning, setArRunning] = useState(false);

const [arTimer, setArTimer] = useState(null);

const [arRows, setArRows] = useState([]);

const startHeliusAR=()=>{

const list = parseTickers(arText);

if(!list.length){ alert("Добавьте тикеры (по одному на строке или через запятую)"); return; }

setArRunning(true);

const id = setInterval(()=>{

const t = list[Math.floor(Math.random()\*list.length)];

const addr = `So1${uid()}${uid()}`.slice(0,16)+"...";

const now = new Date().toISOString().replace("T"," ").slice(0,19);

const row = { id: uid(), name: t.startsWith("$")? t : `$${t}`, mint: addr, team: ["anon","doxxed","unknown"][Math.floor(Math.random()\*3)], source: "Helius/DAS", original: "yes", ts: now };

setArRows(rs=> [row, ...rs].slice(0,200));

setAlerts(a=> [{ts:new Date().toISOString(), type:"mint", asset: row.name, msg:`Найден оригинальный минт ${row.mint}`}, ...a].slice(0,200));

}, 4000 + Math.random()\*3000);

setArTimer(id);

};

const stopHeliusAR=()=>{ setArRunning(false); if(arTimer){ clearInterval(arTimer); setArTimer(null);} };

const parseTickers=(t)=> Array.from(new Set((t||"").replace(/\$/g, "").replace(/[\,\;]+/g, "\n").split(/\n+/).map(s=>s.trim()).filter(Boolean))).slice(0,200);

// Tradeview

const [tradeQuery, setTradeQuery] = useState("");

const [chartData, setChartData] = useState(genSeries());

const [overlayEvents, setOverlayEvents] = useState([]);

const [reasons, setReasons] = useState([]);

// диапазон кита

const [whaleMin, setWhaleMin] = useState(loadLS("whaleMin", 3));

const [whaleMax, setWhaleMax] = useState(loadLS("whaleMax", 19));

const [winSec, setWinSec] = useState(loadLS("winSec", 10));

const [aggr, setAggr] = useState(loadLS("aggr", false));

useEffect(()=>{ saveLS("whaleMin", whaleMin); saveLS("whaleMax", whaleMax); saveLS("winSec", winSec); saveLS("aggr", aggr); },[whaleMin,whaleMax,winSec,aggr]);

const [hypeMetrics, setHypeMetrics] = useState({tw1h:0, tw3h:0, tg1h:0, tg3h:0});

useEffect(()=>{

const id = setInterval(()=>{

setChartData(prev=>{ const last = prev[prev.length-1]?.v ?? 1; const next = Math.max(0.1, last + (Math.random()-0.5)\*0.02); const i = (prev[prev.length-1]?.i ?? 0)+1; return [...prev.slice(-179), {i, v:+next.toFixed(4)}]; });

if(Math.random()>0.92){ const i = chartData[chartData.length-1]?.i ?? 0; const t = Math.random(); const add=(type,color)=> setOverlayEvents(ev=>[...ev, {i,type,color,id:uid()}].slice(-200));

if(t>0.66) add('BUY','#16a34a'); else if(t>0.33) add('SELL','#dc2626'); else add('WHALE','#60a5fa'); if(Math.random()>0.5) add('HYPE','#eab308');

setReasons(makeReasons([...overlayEvents])); setHypeMetrics(h=>({ tw1h: Math.min(9999, h.tw1h + Math.floor(10+Math.random()\*30)), tw3h: Math.min(99999, h.tw3h + Math.floor(30+Math.random()\*90)), tg1h: Math.min(9999, h.tg1h + Math.floor(5+Math.random()\*20)), tg3h: Math.min(99999, h.tg3h + Math.floor(20+Math.random()\*60)), })); }

}, 1200);

return ()=>clearInterval(id);

},[chartData, overlayEvents]);

const makeReasons=(ev)=> ev.slice(-6).reverse().map(e=>{

if(e.type==='BUY') return `BUY: ≥3 стратегий зелёные; AVWAP над ценой; всплеск объёма; киты IN в диапазоне ${whaleMin}–${whaleMax} SOL за ${winSec}s${aggr?" (Aggressive)":""}.`;

if(e.type==='SELL') return `SELL: потеря AVWAP; RSI-слом; Whales OUT; отрицательная дельта объёма.`;

if(e.type==='WHALE') return `Whale IN: ≥2 кошельков в диапазоне ${whaleMin}–${whaleMax} SOL / ${winSec}s${aggr?" (Aggressive: 1 кошелёк допускается)":""}.`;

if(e.type==='HYPE') return `HYPE: рост упоминаний Twitter/Telegram (1h/3h).`;

return `${e.type}`;

});

const onSignals=()=>{ const base = genSeries(); setChartData(base); const ov = [ {i:20,type:'BUY',color:'#16a34a',id:uid()}, {i:60,type:'WHALE',color:'#60a5fa',id:uid()}, {i:110,type:'HYPE',color:'#eab308',id:uid()}, {i:150,type:'SELL',color:'#dc2626',id:uid()} ]; setOverlayEvents(ov); setReasons(makeReasons(ov)); };

// Helius — новый поток минтов

const [helRunning, setHelRunning] = useState(false);

const [helTimer, setHelTimer] = useState(null);

const [helRows, setHelRows] = useState([]); // {name, mint, sol, safe, original, hasTw, team}

const startHel=()=>{

setHelRunning(true);

const id = setInterval(()=>{

const r = { id:uid(), name:`$${["DOGE","PEPE","TRUMP","CAT","WATER"][Math.floor(Math.random()\*5)]}${Math.floor(Math.random()\*100)}`, mint:`So1${uid()}${uid()}`.slice(0,16)+"...", sol: +(Math.random()\*20+1).toFixed(2), safe: true, original: Math.random()>0.4, hasTw: Math.random()>0.5, team:["anon","doxxed","unknown"][Math.floor(Math.random()\*3)], ts: new Date().toISOString().replace("T"," ").slice(0,19) };

setHelRows(x=> [r, ...x].slice(0,200));

}, 3500);

setHelTimer(id);

};

const stopHel=()=>{ setHelRunning(false); if(helTimer){ clearInterval(helTimer); setHelTimer(null);} };

// CEX Radar

const [cexRunning, setCexRunning] = useState(false);

const [cexQuery, setCexQuery] = useState("");

const [cexRows, setCexRows] = useState([]);

const cexSearch=()=>{

const token = cexQuery || "TOKEN";

const mk = (ex,mins)=>({ exchange:ex, date:new Date(Date.now()+mins\*60000).toISOString().slice(0,10), time:new Date(Date.now()+mins\*60000).toISOString().slice(11,19), team:["anon","doxxed","unknown"][Math.floor(Math.random()\*3)], first:true, url:`https://birdeye.so/token/${encodeURIComponent(token)}?chain=solana` });

setCexRows([mk("Binance",15), mk("OKX",30), mk("Bybit",45), mk("MEXC",60)]);

};

// вспомогательные компоненты

const columnsSignals=[

{ key:"word", title:"Название / слово", render:(r)=> (

<div className="flex items-center gap-3">

<div className="size-9 rounded-lg bg-zinc-800"/>

<div>

<div className="font-medium">{r.word}</div>

<div className="text-[11px] text-zinc-500">{r.type} · {r.chain||"-"}</div>

</div>

<Badge tone={r.isOG?"ok":""} className="ml-2">{r.isOG?"OG":"Clone"}</Badge>

</div>

)},

{ key:"detectedAt", title:"Дата/время" },

{ key:"author", title:"Автор", render:(r)=> <a className="text-sky-300" href={r.link} target="\_blank" rel="noreferrer">{r.author}</a> },

{ key:"metrics", title:"Метрики", render:(r)=> <div className="text-xs text-zinc-300 leading-5">Твитов: <b>{r.tweetCount??"-"}</b><br/>Комьюнити: <b>{r.communitySize??"-"}</b><br/>Смен названия: <b>{r.nameChanges??0}</b> · Спам: <b>{Math.round((r.spamScore??0)\*100)}%</b></div> },

{ key:"safety", title:"Безопасность", render:(r)=> <div className="text-xs"><Badge className="mr-1" tone={r.safety?.noMint?"ok":""}>no mint</Badge><Badge className="mr-1" tone={r.safety?.burnLP?"ok":""}>burn LP</Badge><Badge className="mr-1" tone={!r.safety?.blacklist?"ok":"bad"}>{r.safety?.blacklist?"blacklist":"ok"}</Badge></div> },

{ key:"contract", title:"Контракт / ссылки", render:(r)=> <div className="text-xs"><div className="font-mono">{pretty(r.contract)}</div>{r.communityLink&& <div className="mt-1"><a className="text-sky-300" href={r.communityLink} target="\_blank" rel="noreferrer">Комьюнити</a></div>}</div> },

{ key:"summary", title:"Резюме" }

];

// панель очищенных слов (переименована, чтобы не конфликтовать с hypeMetrics)

const hypeWords = useMemo(()=> uniqueHypeWords(rows.filter(r=> (!onlyOG||r.isOG) && (!query || (`${r.word} ${r.author} ${r.contract||""}`.toLowerCase().includes(query.toLowerCase()))))), [rows,onlyOG,query]);

// ======= ВКЛАДКИ =======

const Signals = (

<Card>

<CardHeader><CardTitle>Поток сигналов</CardTitle></CardHeader>

<CardContent>

<div className="mb-4 flex flex-col md:flex-row md:items-center gap-3">

<div className="flex-1"><Input placeholder="Поиск: слово, автор, контракт" value={query} onChange={e=>setQuery(e.target.value)}/></div>

<div className="flex items-center gap-2 text-sm"><input type="checkbox" checked={onlyOG} onChange={e=>setOnlyOG(e.target.checked)}/><span>Только OG</span></div>

<div className="text-xs text-zinc-500">Интервал парсинга: <b>{cadenceMin} мин</b> (настраивается во вкладке «Аккаунты»)</div>

</div>

<Table columns={columnsSignals} data={rows.filter(r=> (!onlyOG||r.isOG) && (!query || (`${r.word} ${r.author} ${r.summary||""} ${r.contract||""}`.toLowerCase().includes(query.toLowerCase()))))} emptyText="Нет сигналов. Запустите парсер или импортируйте JSON."/>

{/\* Нижняя большая панель — только очищенные хайп‑слова \*/}

<div className="mt-4 rounded-xl border border-zinc-800 bg-zinc-900/40 p-4">

<div className="text-sm font-semibold mb-2">Очищенные хайп‑слова</div>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-3">

{hypeWords.length? hypeWords.map((h,i)=>(

<div key={(h.word||"")+i} className="rounded-lg border border-zinc-800 p-3 text-sm">

<div className="font-medium">{h.word}</div>

<div className="text-xs text-zinc-400 mt-1 flex items-center gap-2">

<a className="text-sky-300" href={h.link} target="\_blank" rel="noreferrer">{h.author}</a>

<span>· {h.ts}</span>

<Badge>{h.src}</Badge>

</div>

</div>

)): <div className="text-zinc-500">Пока нет очищенных слов — дождитесь результатов.</div>}

</div>

</div>

</CardContent>

</Card>

);

const Accounts = (

<Card>

<CardHeader right={<div className="flex items-center gap-2 text-sm"><Badge tone="warn">Только для Сигналы</Badge></div>}>

<CardTitle>Аккаунты и настройки парсинга</CardTitle>

</CardHeader>

<CardContent>

{/\* Загрузка аккаунтов \*/}

<div className="grid md:grid-cols-2 gap-4">

<div>

<Label>Загрузить аккаунты Twitter (CSV/JSON, по @handle)</Label>

<Input type="file" accept=".csv,application/json" onChange={async (e)=>{ const f=e.target.files?.[0]; if(!f) return; const txt=await f.text(); let list=[]; try{ list=JSON.parse(txt); if(!Array.isArray(list)) list=[]; }catch{ list = txt.split(/\r?\n/).map(s=>s.trim()).filter(Boolean);} log(`[accounts] Twitter +${list.length}`); }} />

</div>

<div>

<Label>Загрузить группы Telegram (CSV/JSON, по ссылкам)</Label>

<Input type="file" accept=".csv,application/json" onChange={async (e)=>{ const f=e.target.files?.[0]; if(!f) return; const txt=await f.text(); let list=[]; try{ list=JSON.parse(txt); if(!Array.isArray(list)) list=[]; }catch{ list = txt.split(/\r?\n/).map(s=>s.trim()).filter(Boolean);} log(`[accounts] Telegram +${list.length}`); }} />

</div>

</div>

{/\* Правила парсинга \*/}

<div className="mt-4">

<Label>Правила парсинга (JSON) — Twitter/TG</Label>

<Input type="file" accept="application/json" onChange={async (e)=>{ const f=e.target.files?.[0]; if(!f) return; const txt=await f.text(); try{ JSON.parse(txt); log(`[rules] Загружены правила (${f.name})`);}catch{ log(`[rules] Ошибка JSON в ${f.name}`);} }} />

</div>

{/\* Интервал парсинга \*/}

<div className="mt-4 grid grid-cols-2 md:grid-cols-4 gap-3">

{[5,15,30,60].map(v=> (

<Button key={v} variant={cadenceMin===v?"success":"ghost"} onClick={()=>{ setCadenceMin(v); log(`[cadence] ${v} мин`); }}>{v} мин</Button>

))}

</div>

{/\* Менеджер API‑ключей \*/}

<div className="mt-6">

<div className="text-sm font-semibold mb-2">API ключи</div>

<div className="grid md:grid-cols-2 gap-3">

{Object.keys(keys).map(k=> (

<div key={k} className="flex items-center gap-2">

<Label className="w-32 capitalize">{k}</Label>

<Input type="password" placeholder={`ключ ${k}`} value={keys[k]} onChange={e=>setKeys({...keys, [k]: e.target.value})}/>

<Button size="sm" onClick={()=>saveKey(k)}>Сохранить</Button>

<Led ok={keyStatus[k]==='ok'} />

</div>

))}

</div>

<div className="text-xs text-zinc-500 mt-1">Индикатор зелёный — ключ активен; красный — ошибка (подробности см. «Логи»).</div>

</div>

</CardContent>

</Card>

);

const HeliusAR = (

<Card>

<CardHeader right={<div className="flex items-center gap-2">{arRunning? <Badge tone="ok">Работает</Badge>: <Badge tone="bad">Стоп</Badge>}<Button onClick={startHeliusAR}>Поиск</Button><Button variant="ghost" onClick={stopHeliusAR}>Стоп</Button></div>}>

<CardTitle>Helius AR — отслеживание будущих названий</CardTitle>

</CardHeader>

<CardContent>

<div className="grid grid-cols-1 md:grid-cols-3 gap-3">

<div className="md:col-span-2">

<Label>Список тикеров (по одному на строке или через запятую)</Label>

<Textarea placeholder={`$TRUMP\n$PEPE\n$DOGE ...`} value={arText} onChange={e=>setArText(e.target.value)} />

<p className="text-xs text-zinc-500 mt-1">Helius включает таймер и ждёт появления <b>оригинальных</b> минтов с этими названиями.</p>

</div>

<div>

<Label>Результаты</Label>

<div className="text-xs text-zinc-500">Новые совпадения появятся в таблице и в «Алерты».</div>

</div>

</div>

<div className="mt-4 rounded-lg border border-zinc-800">

<table className="w-full text-sm">

<thead className="bg-zinc-900/60"><tr><th className="px-3 py-2">Название</th><th className="px-3 py-2">Минт</th><th className="px-3 py-2">Команда</th><th className="px-3 py-2">Источник</th><th className="px-3 py-2">Ориг.</th><th className="px-3 py-2">Дата/время</th></tr></thead>

<tbody>

{arRows.length? arRows.map((r)=> (

<tr key={r.id} className="border-t border-zinc-800/60"><td className="px-3 py-2">{r.name}</td><td className="px-3 py-2 font-mono">{r.mint}</td><td className="px-3 py-2">{r.team}</td><td className="px-3 py-2">{r.source}</td><td className="px-3 py-2">{r.original?"yes":"no"}</td><td className="px-3 py-2">{r.ts}</td></tr>

)): <tr><td className="px-3 py-3 text-center text-zinc-500" colSpan={6}>Пусто</td></tr>}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

const ChartWithOverlays = (

<div className="relative w-full h-[380px] rounded-xl border border-zinc-800 bg-zinc-900/40">

<ResponsiveContainer width="100%" height="100%">

<LineChart data={chartData} margin={{left:12,right:12,top:8,bottom:8}}>

<CartesianGrid stroke="#222" />

<XAxis dataKey="i" hide/>

<YAxis domain={["dataMin","dataMax"]} width={50} stroke="#666"/>

<Tooltip formatter={(v)=>v} labelFormatter={(l)=>`t=${l}`} contentStyle={{background:"#0a0a0a", border:"1px solid #3f3f46"}}/>

<Line type="monotone" dataKey="v" dot={false} strokeWidth={2}/>

{overlayEvents.map(e=> (

<ReferenceArea key={e.id} x1={e.i-0.5} x2={e.i+0.5} y1={0} y2={99999} fill={e.color} fillOpacity={0.18} stroke={e.color} strokeOpacity={0.4} />

))}

</LineChart>

</ResponsiveContainer>

<div className="absolute top-2 right-2 flex gap-2 text-[11px]">

<Badge tone="ok">зелёный = BUY</Badge>

<Badge tone="bad">красный = SELL</Badge>

<Badge className="bg-sky-700 text-white">синий = WHALE</Badge>

<Badge className="bg-amber-600 text-white">жёлтый = HYPE</Badge>

</div>

</div>

);

const Tradeview = (

<Card>

<CardHeader>

<CardTitle>Tradeview — инфо‑режим (график «как Birdeye», но локально)</CardTitle>

</CardHeader>

<CardContent>

<div className="grid grid-cols-1 md:grid-cols-5 gap-3 items-center">

<div className="md:col-span-2 flex items-center gap-2">

<Input placeholder="вставьте название или контракт" value={tradeQuery} onChange={e=>setTradeQuery(e.target.value)} />

<Button onClick={onSignals}>Сигналы</Button>

</div>

<div className="flex items-center gap-2 text-sm"><Label>Whale SOL от</Label><NumberInput className="w-20" min={1} max={whaleMax} value={whaleMin} onChange={e=>setWhaleMin(Math.max(1, Math.min(parseInt(e.target.value||"3"), whaleMax)))}/></div>

<div className="flex items-center gap-2 text-sm"><Label>до</Label><NumberInput className="w-20" min={whaleMin} max={100} value={whaleMax} onChange={e=>setWhaleMax(Math.max(whaleMin, parseInt(e.target.value||"19")))}/></div>

<div className="flex items-center gap-2 text-sm"><Label>Окно (сек)</Label><NumberInput className="w-20" min={1} max={60} value={winSec} onChange={e=>setWinSec(parseInt(e.target.value||"10"))}/></div>

<div className="flex items-center gap-2 text-sm"><input type="checkbox" checked={aggr} onChange={e=>setAggr(e.target.checked)}/><span>Aggressive</span></div>

</div>

<div className="mt-4 grid grid-cols-1 lg:grid-cols-3 gap-4">

<div className="lg:col-span-2">

{ChartWithOverlays}

<div className="mt-3 grid grid-cols-2 gap-3 text-xs">

<div className="rounded-lg border border-zinc-800 p-3">

<div className="text-zinc-400 mb-1">Метрики хайпа</div>

<div>Twitter: 1h <b>{hypeMetrics.tw1h}</b> · 3h <b>{hypeMetrics.tw3h}</b></div>

<div>Telegram: 1h <b>{hypeMetrics.tg1h}</b> · 3h <b>{hypeMetrics.tg3h}</b></div>

</div>

<div className="rounded-lg border border-zinc-800 p-3">

<div className="text-зinc-400 mb-1">Причины (почему так)</div>

<ul className="list-disc ml-5 leading-6">

{reasons.length? reasons.map((t,i)=>(<li key={i}>{t}</li>)) : <li>Нажмите «Сигналы», чтобы получить анализ.</li>}

</ul>

</div>

</div>

</div>

<div className="space-y-3">

<Card>

<CardHeader><CardTitle>Сигналы</CardTitle></CardHeader>

<CardContent className="text-sm space-y-2">

{overlayEvents.slice(-10).reverse().map(e=> (

<div key={e.id} className="flex items-center gap-2"><span className="inline-block w-2 h-2 rounded-full" style={{background:e.color}}></span><span>{e.type}</span><Badge className="ml-auto">t≈{e.i}</Badge></div>

))}

{!overlayEvents.length && <div className="text-zinc-500">Сигналов пока нет</div>}

</CardContent>

</Card>

<Card>

<CardHeader><CardTitle>Потоки</CardTitle></CardHeader>

<CardContent className="text-sm space-y-2">

<div>Whale IN/OUT · BUY/SELL · Risk</div>

<div className="text-xs text-zinc-500">(реально: Helius swaps/liquidity + RugCheck + DexScreener/Birdeye)</div>

</CardContent>

</Card>

<Card>

<CardHeader><CardTitle>Загрузка знаний</CardTitle></CardHeader>

<CardContent className="text-sm space-y-2">

<div className="grid grid-cols-1 gap-2">

<label className="text-xs text-zinc-400">Книги <Input type="file" multiple /></label>

<label className="text-xs text-zinc-400">Глобальные правила <Input type="file" multiple /></label>

<label className="text-xs text-zinc-400">Тонкие настройки <Input type="file" multiple /></label>

<label className="text-xs text-zinc-400">Графики/фото <Input type="file" multiple /></label>

</div>

</CardContent>

</Card>

</div>

</div>

</CardContent>

</Card>

);

const Helius = (

<Card>

<CardHeader right={<div className="flex items-center gap-2"><Button onClick={startHel}>Старт</Button><Button variant="ghost" onClick={stopHel}>Стоп</Button>{helRunning? <Badge tone="ok">работает</Badge>: <Badge tone="bad">стоп</Badge>}</div>}>

<CardTitle>Helius мониторинг (прошедшие 5–9 проверок)</CardTitle>

</CardHeader>

<CardContent>

<div className="rounded-lg border border-зinc-800 overflow-hidden">

<table className="w-full text-sm">

<thead className="bg-zinc-900/60"><tr>

<th className="px-3 py-2">Безоп.</th><th className="px-3 py-2">Ориг.</th><th className="px-3 py-2">Twitter</th><th className="px-3 py-2">Команда</th><th className="px-3 py-2">Название</th><th className="px-3 py-2">Минт</th><th className="px-3 py-2">SOL</th><th className="px-3 py-2">Скачать</th>

</tr></thead>

<tbody>

{helRows.length? helRows.map(r=> (

<tr key={r.id} className="border-t border-zinc-800/60">

<td className="px-3 py-2">{r.safe?"✓":"✗"}</td>

<td className="px-3 py-2">{r.original?"✓":""}</td>

<td className="px-3 py-2">{r.hasTw?"✓":""}</td>

<td className="px-3 py-2">{r.team}</td>

<td className="px-3 py-2">{r.name}</td>

<td className="px-3 py-2 font-mono">{r.mint}</td>

<td className="px-3 py-2">{r.sol}</td>

<td className="px-3 py-2"><Button size="sm" onClick={()=>download(`${r.name}\_mint.txt`, r.mint, 'text/plain')}>⬇</Button></td>

</tr>

)): <tr><td className="px-3 py-3 text-center text-zinc-500" colSpan={8}>Нет данных — нажмите Старт</td></tr>}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

const CEXRadar = (

<Card>

<CardHeader right={<div className="flex items-center gap-2"><Input placeholder="контракт или тикер" value={cexQuery} onChange={e=>setCexQuery(e.target.value)} className="w-60"/><Button onClick={()=>{ setCexRunning(true); cexSearch(); }}>Старт</Button><Button variant="ghost" onClick={()=>{ setCexRunning(false); setCexRows([]); }}>Стоп</Button></div>}>

<CardTitle>CEX Radar — первые листинки</CardTitle>

</CardHeader>

<CardContent>

<div className="rounded-lg border border-zinc-800 overflow-hidden">

<table className="w-full text-sm">

<thead className="bg-zinc-900/60"><tr>

<th className="px-3 py-2">Биржа</th><th className="px-3 py-2">Дата</th><th className="px-3 py-2">Время</th><th className="px-3 py-2">Команда</th><th className="px-3 py-2">Открыть</th>

</tr></thead>

<tbody>

{cexRows.length? cexRows.map((r,i)=> (

<tr key={i} className="border-t border-zinc-800/60">

<td className="px-3 py-2">{r.exchange}</td>

<td className="px-3 py-2">{r.date}</td>

<td className="px-3 py-2">{r.time}</td>

<td className="px-3 py-2">{r.team}</td>

<td className="px-3 py-2"><a className="text-sky-300" href={r.url} target="\_blank" rel="noreferrer">Birdeye</a></td>

</tr>

)): <tr><td className="px-3 py-3 text-center text-zinc-500" colSpan={5}>Нажмите Старт и введите название/контракт</td></tr>}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

const Learn = (

<Card>

<CardHeader><CardTitle>Учиться (Knowledge)</CardTitle></CardHeader>

<CardContent>

<div className="grid md:grid-cols-2 gap-4">

<div className="space-y-2">

<Label>Книги</Label>

<Input type="file" multiple />

<Label>Глобальные правила</Label>

<Input type="file" multiple />

</div>

<div className="space-y-2">

<Label>Тонкие настройки</Label>

<Input type="file" multiple />

<Label>Графики/фото</Label>

<Input type="file" multiple />

</div>

</div>

</CardContent>

</Card>

);

const Chat = (

<Card>

<CardHeader><CardTitle>Чат с ИИ (демо)</CardTitle></CardHeader>

<CardContent>

<div className="max-h-64 overflow-auto rounded-lg border border-zinc-800 p-2 text-sm bg-zinc-900/40">

<div className="text-zinc-500">Пусто</div>

</div>

<div className="mt-2 flex gap-2">

<Input id="ask" placeholder="Спросить у ИИ…"/>

<Button onClick={()=>{ const el=document.getElementById("ask"); const v=el&&el.value; if(v){ setAlerts(a=>[{ts:new Date().toISOString(), type:"ai", asset:"chat", msg:"ИИ ответил (демо)"}, ...a]); el.value=""; } }}>Спросить</Button>

</div>

</CardContent>

</Card>

);

const Logs = (

<Card>

<CardHeader right={<Badge tone={unread?"warn":""}>🔔 {unread}</Badge>}><CardTitle>Логи</CardTitle></CardHeader>

<CardContent>

<div className="text-xs text-zinc-300 whitespace-pre-wrap min-h-[120px]">{logs.map(l=>`[${new Date(l.ts).toLocaleTimeString()}] ${l.msg}`).join("\n")||"Пока пусто"}</div>

</CardContent>

</Card>

);

// ======= SELFTESTS =======

const [tests, setTests] = useState([]);

useEffect(()=>{

const T = []; const A=(name, cond, details="")=>T.push({name, ok:!!cond, details});

// parseTickers

const ticks = parseTickers(" $TRUMP, PEPE;DOGE\nDOGE\n$WATER ");

A("parseTickers unique+$-strip", JSON.stringify(ticks) === JSON.stringify(["TRUMP","PEPE","DOGE","WATER"]), JSON.stringify(ticks));

// uniqueHypeWords

const uh = uniqueHypeWords([{word:"A",author:"@x",link:"/",detectedAt:"2025-09-01 10:00",source:"Twitter"},{word:"a",author:"@y",link:"/2",detectedAt:"2025-09-01 11:00",source:"Telegram"}]);

A("uniqueHypeWords picks latest", uh.length===1 && uh[0].author==='@y');

// genSeries

const s = genSeries(10, 1); A("genSeries len=10", s.length===10 && typeof s[0].v === "number");

// CSV join (из прошлых тестов)

const sample = [{word:"A", detectedAt:"t", author:"b", contract:"c", summary:"x\ny"}];

const csv = sample.map(r=> [r.word,r.detectedAt,r.author,r.contract??"", (r.summary??"").replace(/\n/g," ")].map(x=>`"${String(x).replace(/"/g,'""')}"`).join(",")).join("\n");

A("CSV flattens newlines", csv.includes('"x y"'));

// Доп. тест: CSV одна строка

A("CSV one row", csv.split("\n").length===1);

setTests(T);

},[]);

const SelfTests = (

<Card>

<CardHeader><CardTitle>Selftests</CardTitle></CardHeader>

<CardContent>

<div className="grid sm:grid-cols-2 lg:grid-cols-3 gap-2">

{tests.map((t,i)=>(

<div key={i} className="flex items-center gap-2 rounded-lg border border-zinc-800 bg-zinc-900/50 px-3 py-2 text-sm">

<Badge tone={t.ok?"ok":"bad"}>{t.ok?"PASS":"FAIL"}</Badge>

<span>{t.name}</span>

{t.details? <code className="text-zinc-400 truncate">{String(t.details)}</code>: null}

</div>

))}

{!tests.length && <div className="text-zinc-500">Running…</div>}

</div>

</CardContent>

</Card>

);

// ======= RENDER =======

return (

<div className="min-h-screen w-full bg-black text-white">

<div className="mx-auto max-w-[1400px] px-6 py-6">

{/\* Header \*/}

<div className="flex items-center justify-between">

<div>

<h1 className="text-xl font-semibold">Super Parser AI — интерфейс</h1>

<p className="text-xs text-zinc-400">Dark UI • Groq · Apify · Helius · QuickNode · Pumpfun · Nansen</p>

</div>

<div className="flex items-center gap-2">

<Button className="bg-emerald-600" onClick={()=>setRunning(!running)}>{running?"Стоп":"Старт"}</Button>

<Button onClick={()=>download("signals.json", JSON.stringify(rows,null,2), "application/json")}>JSON</Button>

<Button onClick={()=>{ const headers=["Название/слово","Дата/время","Автор","Контракт","Резюме"].join(","); const csv = rows.map(r=> [r.word,r.detectedAt,r.author,r.contract??"", (r.summary??"").replace(/\n/g," ")].map(x=>`"${String(x).replace(/"/g,'""')}"`).join(",")).join("\n"); download("signals.csv", headers+"\n"+csv, "text/csv;charset=utf-8"); }}>CSV</Button>

</div>

</div>

<Tabs value={tab} onValueChange={setTab} className="mt-6">

<TabsList>

<TabsTrigger value="signals">Сигналы</TabsTrigger>

<TabsTrigger value="accounts">Аккаунты</TabsTrigger>

<TabsTrigger value="heliusAR">Helius AR</TabsTrigger>

<TabsTrigger value="tradeview">Tradeview</TabsTrigger>

<TabsTrigger value="filters">Фильтры</TabsTrigger>

<TabsTrigger value="alerts">Алерты</TabsTrigger>

<TabsTrigger value="helius">Helius</TabsTrigger>

<TabsTrigger value="cexradar">CEX Radar</TabsTrigger>

<TabsTrigger value="learn">Учиться</TabsTrigger>

<TabsTrigger value="chat">Чат</TabsTrigger>

<TabsTrigger value="logs">Логи</TabsTrigger>

</TabsList>

<TabsContent value="signals">{Signals}</TabsContent>

<TabsContent value="accounts">{Accounts}</TabsContent>

<TabsContent value="heliusAR">{HeliusAR}</TabsContent>

<TabsContent value="tradeview">{Tradeview}</TabsContent>

<TabsContent value="filters">{/\* без изменений \*/}</TabsContent>

<TabsContent value="alerts">{/\* без изменений (алерты из других вкладок) \*/}</TabsContent>

<TabsContent value="helius">{Helius}</TabsContent>

<TabsContent value="cexradar">{CEXRadar}</TabsContent>

<TabsContent value="learn">{Learn}</TabsContent>

<TabsContent value="chat">{Chat}</TabsContent>

<TabsContent value="logs">{Logs}</TabsContent>

</Tabs>

<div className="mt-6">{SelfTests}</div>

</div>

</div>

);

}