```python
# main.py — чистая версия с поддержкой состояния визуализатора

import os
import json
import uuid
import pickle
import tempfile
from typing import Dict, List, Any, Optional
from io import BytesIO
from update_projects import update_projects_if_templates_changed

import pandas as pd
from fastapi import FastAPI, HTTPException, Request
from fastapi.responses import JSONResponse, FileResponse
from fastapi.staticfiles import StaticFiles
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel


# ============================================================================
# КОНФИГУРАЦИЯ
# ============================================================================

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
TEMPLATES_PATH = os.path.join(BASE_DIR, "formula_templates.json")
SIGNAL_INDEX_PATH = os.path.join(BASE_DIR, ".signal_index.pkl")


# ============================================================================
# PYDANTIC МОДЕЛИ
# ============================================================================

class VisualizerStateRequest(BaseModel):
    """Запрос на сохранение состояния визуализатора"""
    session_token: str
    state: Dict[str, Any]


class VisualizerStateResponse(BaseModel):
    """Ответ с состоянием визуализатора"""
    success: bool
    state: Optional[Dict[str, Any]] = None
    message: Optional[str] = None


class VisualizeSessionRequest(BaseModel):
    """Запрос на создание сессии визуализации"""
    signals: List[str]
    code: str = ""
    visualizer_state: Optional[Dict[str, Any]] = None


# ============================================================================
# ГЛОБАЛЬНОЕ СОСТОЯНИЕ
# ============================================================================

STATE = {
    "settings": None,
    "signals": None,
    "signal_index": None,
    "templates": None
}

# Хранилище сессий визуализатора (в памяти)
```

```
visualize_sessions: Dict[str, Dict[str, Any]] = {}


# ============================================================================
# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — ЗАГРУЗКА НАСТРОЕК
# ============================================================================

def load_settings() -> Dict:
    """Загружает настройки из settings.json"""
    with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
        return json.load(f)


def load_templates() -> Dict:
    """Загружает шаблоны формул"""
    if not os.path.exists(TEMPLATES_PATH):
        return {"templates": []}
    with open(TEMPLATES_PATH, "r", encoding="utf-8") as f:
        return json.load(f)


# ============================================================================
# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — СИГНАЛЫ
# ============================================================================

def load_signals_from_folder(folder: str) -> List[Dict]:
    """Загружает описания сигналов из CSV файлов"""
    folder_abs = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")

    signals_map = {}
    for name in os.listdir(folder_abs):
        if not name.lower().endswith(".csv"):
            continue
        path = os.path.join(folder_abs, name)
        try:
            try:
                df = pd.read_csv(path, sep=';')[['Tagname', 'Description',
'Engineering Unit']]
            except KeyError:
                df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
            df = df.dropna(subset=['Tagname'])

            for _, row in df.iterrows():
                tag = str(row['Tagname']).strip()
                desc = "" if pd.isna(row['Description']) else
str(row['Description']).strip()
                try:
                    unit = "" if pd.isna(row['Engineering Unit']) else
str(row['Engineering Unit']).strip()
                except KeyError:
                    unit = ""
                desc_full = ", ".join([x for x in [desc, unit] if x])

                if tag:
                    signals_map[tag] = {
                        "Tagname": tag,
                        "Description": desc_full,
                        "EngineeringUnit": unit
                    }
        except Exception as e:
            print(f"[WARN] failed to read {path}: {e}")

```

```
    out = list(signals_map.values())
    out.sort(key=lambda x: x["Tagname"])
    return out


def load_project_signals(folder: str) -> List[Dict]:
    """Загружает сигналы из проектов (синтетические сигналы)"""
    folder_abs = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
    if not os.path.isdir(folder_abs):
        return []

    out = []
    for name in os.listdir(folder_abs):
        if not name.endswith(".json"):
            continue
        path = os.path.join(folder_abs, name)
        try:
            with open(path, "r", encoding="utf-8") as f:
                payload = json.load(f)

            proj = payload.get("project", {}) or {}
            code = (proj.get("code") or "").strip()

            if not code:
                continue

            desc = (proj.get("description") or "").strip()
            dim = (proj.get("dimension") or "").strip()

            out.append({
                "Tagname": code,
                "Description": desc,
                "EngineeringUnit": dim,
                "Type": proj.get("type", "")
            })
        except Exception as e:
            print(f"[WARN] failed to read project {path}: {e}")
            continue

    out.sort(key=lambda x: x["Tagname"])
    return out


def refresh_signals_cache():
    """Обновляет кэш сигналов (базовые + из проектов)"""
    settings = STATE["settings"] or {}
    base_folder = settings.get("signalDataFolder")
    proj_folder = settings.get("projectDataFolder")

    base = load_signals_from_folder(base_folder) if base_folder else []
    proj = load_project_signals(proj_folder) if proj_folder else []

    merged = {}
    for s in base:
        merged[s["Tagname"]] = s
    for s in proj:
        merged[s["Tagname"]] = s  # проекты перекрывают CSV

    out = list(merged.values())
    out.sort(key=lambda x: x["Tagname"])
    STATE["signals"] = out


# ============================================================================
```

```
# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — ИНДЕКС СИГНАЛОВ
# ============================================================================

def build_signal_index(folder: str) -> Dict[str, List[str]]:
    """Строит индекс: signal_name -> list of files"""
    folder_abs = folder if os.path.isabs(folder) else os.path.normpath(os.path.join(BASE_DIR, folder))

    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"Signal data folder not found: {folder_abs}")

    signal_index = {}
    print(f"[INFO] Building signal index from {folder_abs}...")

    for filename in os.listdir(folder_abs):
        if not filename.lower().endswith(".csv"):
            continue

        filepath = os.path.join(folder_abs, filename)
        try:
            df_header = pd.read_csv(filepath, nrows=0, encoding="ISO-8859-2", sep=";")
            columns = df_header.columns.tolist()
            signal_columns = [c for c in columns if c not in ["DATE", "TIME", "datetime"]]

            for signal_name in signal_columns:
                if signal_name not in signal_index:
                    signal_index[signal_name] = []
                signal_index[signal_name].append(filepath)

            print(f"  ✓ {filename}: {len(signal_columns)} signals")
        except Exception as e:
            print(f"  ✗ Failed to index {filename}: {e}")
            continue

    print(f"[OK] Total unique signals indexed: {len(signal_index)}")
    return signal_index


def load_signal_index(folder: str) -> Dict[str, List[str]]:
    """Загружает индекс из кэша или перестраивает"""
    folder_abs = folder if os.path.isabs(folder) else os.path.normpath(os.path.join(BASE_DIR, folder))

    def get_folder_state(path: str) -> dict:
        if not os.path.isdir(path):
            return {}
        state = {}
        for name in os.listdir(path):
            if name.lower().endswith(".csv"):
                filepath = os.path.join(path, name)
                state[name] = os.path.getmtime(filepath)
        return state

    current_state = get_folder_state(folder_abs)

    # Пробуем загрузить кэш
    if os.path.exists(SIGNAL_INDEX_PATH):
        try:
            with open(SIGNAL_INDEX_PATH, "rb") as f:
                cached_data = pickle.load(f)

            if isinstance(cached_data, dict) and "_folder_state" in cached_data:
                cached_state = cached_data["_folder_state"]
                cached_index = cached_data["index"]
```

```
253
254                 if cached_state == current_state:
255                     print(f"[OK] Signal index loaded from cache ({len(cached_index)}
    signals)")
256                     return cached_index
257                 else:
258                     print(f"[INFO] CSV files changed, rebuilding index...")
259             else:
260                 print(f"[INFO] Old cache format, rebuilding index...")
261         except Exception as e:
262             print(f"[WARN] Failed to load cached index: {e}")
263
264     # Перестраиваем индекс
265     index = build_signal_index(folder)
266
267     # Сохраняем с метаданными
268     try:
269         cache_data = {"index": index, "_folder_state": current_state}
270         with open(SIGNAL_INDEX_PATH, "wb") as f:
271             pickle.dump(cache_data, f)
272         print(f"[OK] Signal index cached with folder state")
273     except Exception as e:
274         print(f"[WARN] Failed to cache signal index: {e}")
275
276     return index
277
278
279 def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
    pd.DataFrame]:
280     """Загружает только нужные сигналы из только нужных файлов"""
281     folder_abs = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
282
283     signal_index = STATE.get("signal_index", {})
284     if not signal_index:
285         raise RuntimeError("Signal index not initialized")
286
287     signal_names_set = set(signal_names)
288     found_signals = {}
289     files_to_load = set()
290
291     for signal_name in signal_names_set:
292         if signal_name in signal_index:
293             files_to_load.update(signal_index[signal_name])
294
295     print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)}
    files")
296
297     for filepath in files_to_load:
298         try:
299             df = pd.read_csv(filepath, encoding="ISO-8859-2", sep=";")
300
301             df["TIME"] = df["TIME"].str.replace(",", ".", regex=False)
302             df["TIME"] = df["TIME"].str.split(".").str[0]
303             combined = df["DATE"] + " " + df["TIME"]
304             df["datetime"] = pd.to_datetime(combined, format="%d.%m.%Y %H:%M:%S",
    errors="coerce")
305             df = df.dropna(subset=["datetime"])
306             df = df.drop(['DATE', 'TIME'], axis=1)
307             df = df.sort_values("datetime")
308
309             available_columns = set(df.columns) & signal_names_set
310             for signal_name in available_columns:
311                 if signal_name not in found_signals:
312                     found_signals[signal_name] = df[["datetime", signal_name]].copy()
```

```
313                     found_signals[signal_name].columns = ["datetime", "value"]
314         except Exception as e:
315             print(f"[WARN] Failed to read {filepath}: {e}")
316             continue
317
318     return found_signals
319
320
321 # ================================================================================
322 # ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — ПРОЕКТЫ И ЗАВИСИМОСТИ
323 # ================================================================================
324
325 def get_project_path(filename: str) -> str:
326     """Возвращает абсолютный путь к файлу проекта"""
327     folder = STATE["settings"].get("projectDataFolder")
328     if not folder:
329         raise RuntimeError("projectDataFolder not configured")
330
331     project_dir = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
332
333     if '..' in filename or '/' in filename or '\\' in filename:
334         raise HTTPException(status_code=400, detail="Invalid filename")
335
336     path = os.path.join(project_dir, filename)
337     if not path.startswith(project_dir):
338         raise HTTPException(status_code=400, detail="Path traversal attempt")
339
340     return path
341
342
343 def extract_input_signals_from_project(project_data: Dict) -> List[str]:
344     """Извлекает имена входных сигналов из данных проекта"""
345     elements = project_data.get("elements", {})
346     input_signals = []
347
348     for elem_id, elem_data in elements.items():
349         if elem_data.get("type") == "input-signal":
350             props = elem_data.get("props", {})
351             signal_name = props.get("name")
352             if signal_name:
353                 input_signals.append(signal_name)
354
355     return input_signals
356
357
358 def load_project_by_code(code: str) -> Dict | None:
359     """Загружает проект по его коду (Tagname)"""
360     folder = STATE["settings"].get("projectDataFolder")
361     if not folder:
362         return None
363
364     folder_abs = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
365     if not os.path.isdir(folder_abs):
366         return None
367
368     for name in os.listdir(folder_abs):
369         if not name.endswith(".json"):
370             continue
371         path = os.path.join(folder_abs, name)
372         try:
373             with open(path, "r", encoding="utf-8") as f:
374                 payload = json.load(f)
375             proj = payload.get("project", {})
```

```
            if proj.get("code") == code:
                return {
                    "project": proj,
                    "formula": payload.get("code", ""),
                    "elements": payload.get("elements", {})
                }
        except Exception as e:
            print(f"[WARN] Error reading project {path}: {e}")
            continue

    return None


def is_base_signal(signal_name: str) -> bool:
    """Проверяет, есть ли сигнал в архиве (базовый сигнал с данными)"""
    signal_index = STATE.get("signal_index", {})
    return signal_name in signal_index


def resolve_signal_dependencies(
    signal_names: List[str],
    visited: set = None,
    resolved: Dict[str, Dict] = None
) -> tuple[set, Dict[str, Dict]]:
    """Рекурсивно разворачивает зависимости сигналов"""
    if visited is None:
        visited = set()
    if resolved is None:
        resolved = {}

    base_signals = set()

    for signal_name in signal_names:
        if not signal_name or signal_name in visited:
            continue
        visited.add(signal_name)

        if is_base_signal(signal_name):
            base_signals.add(signal_name)
            continue

        project = load_project_by_code(signal_name)
        if project is None:
            base_signals.add(signal_name)
            print(f"[WARN] Signal '{signal_name}' not found in archive or projects")
            continue

        formula = project.get("formula", "")
        dependencies = extract_input_signals_from_project(project)

        print(f"[INFO] Synthetic signal '{signal_name}' depends on: {dependencies}")

        resolved[signal_name] = {
            "formula": formula,
            "dependencies": dependencies
        }

        sub_base, _ = resolve_signal_dependencies(dependencies, visited, resolved)
        base_signals.update(sub_base)

    return base_signals, resolved


def topological_sort_signals(synthetic_signals: Dict[str, Dict]) -> List[str]:
    """Топологическая сортировка синтетических сигналов"""
```

```
    if not synthetic_signals:
        return []

    in_degree = {name: 0 for name in synthetic_signals}
    graph = {name: [] for name in synthetic_signals}

    for name, data in synthetic_signals.items():
        for dep in data.get("dependencies", []):
            if dep in synthetic_signals:
                graph[dep].append(name)
                in_degree[name] += 1

    queue = [name for name, degree in in_degree.items() if degree == 0]
    result = []

    while queue:
        node = queue.pop(0)
        result.append(node)

        for neighbor in graph[node]:
            in_degree[neighbor] -= 1
            if in_degree[neighbor] == 0:
                queue.append(neighbor)

    if len(result) != len(synthetic_signals):
        cyclic = [name for name in synthetic_signals if name not in result]
        raise ValueError(f"Циклическая зависимость между сигналами: {cyclic}")

    return result


# ============================================================================
# FASTAPI ПРИЛОЖЕНИЕ
# ============================================================================

app = FastAPI(title="Logic Scheme Editor API")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


@app.on_event("startup")
def startup():
    """Инициализация при запуске"""
    settings = load_settings()
    STATE["settings"] = settings

    project_dir = settings.get("projectDataFolder")
    if project_dir and not os.path.isabs(project_dir):
        project_dir = os.path.normpath(os.path.join(BASE_DIR, project_dir))

    update_projects_if_templates_changed(
        project_dir=project_dir,
        templates_path=TEMPLATES_PATH
    )

    folder = settings.get("signalDataFolder")
    if not folder:
        raise RuntimeError("settings.json: signalDataFolder is required")

```

```
506      refresh_signals_cache()
507      STATE["templates"] = load_templates()
508      STATE["signal_index"] = load_signal_index(settings.get("signalArchiveFolder"))
509
510      print(f"[OK] Loaded signals: {len(STATE['signals'])}")
511      print(f"[OK] Signal index has {len(STATE['signal_index'])} unique signals")
512      print(f"[OK] Loaded templates: {len(STATE['templates'].get('templates', []))}")
513
514
515  # ============================================================================
516  # API — НАСТРОЙКИ И СИГНАЛЫ
517  # ============================================================================
518
519  @app.get("/api/settings")
520  def api_settings():
521      """Возвращает настройки приложения"""
522      return STATE["settings"]
523
524
525  @app.get("/api/signals")
526  def api_signals(q: str = "", limit: int = 50):
527      """Поиск сигналов по маске (* — wildcard)"""
528      signals = STATE["signals"] or []
529
530      if not q:
531          result = {"items": signals[:limit], "total": len(signals)}
532      else:
533          import re
534          escaped = re.escape(q).replace(r"\*", ".*")
535          rx = re.compile("^" + escaped + "$", re.IGNORECASE)
536          items = [s for s in signals if rx.match(s["Tagname"])]
537          result = {"items": items[:max(1, min(limit, 500))], "total": len(items)}
538
539      return JSONResponse(
540          content=result,
541          headers={
542              "Cache-Control": "no-cache, no-store, must-revalidate",
543              "Pragma": "no-cache",
544              "Expires": "0"
545          }
546      )
547
548
549  @app.get("/api/formula-templates")
550  def api_formula_templates():
551      """Возвращает шаблоны формул"""
552      return STATE.get("templates") or {"templates": []}
553
554
555  # ============================================================================
556  # API — ПРОЕКТЫ
557  # ============================================================================
558
559  @app.get("/api/project/list")
560  def list_projects():
561      """Список всех проектов"""
562      folder = STATE["settings"].get("projectDataFolder")
563      if not folder:
564          raise HTTPException(status_code=500, detail="Project folder not configured")
565
566      project_dir = folder if os.path.isabs(folder) else
     os.path.normpath(os.path.join(BASE_DIR, folder))
567      os.makedirs(project_dir, exist_ok=True)
568
569      projects = []
```

```
    for fname in sorted(os.listdir(project_dir)):
        if not fname.endswith(".json"):
            continue
        path = os.path.join(project_dir, fname)
        try:
            with open(path, "r", encoding="utf-8") as f:
                payload = json.load(f)
        except Exception:
            continue
        project_meta = payload.get("project", {})
        projects.append({
            "filename": fname,
            "code": project_meta.get("code") or project_meta.get("tagname") or "",
            "description": project_meta.get("description") or "",
            "type": project_meta.get("type") or ""
        })
    return {"projects": projects}


@app.post("/api/project/save")
async def save_project(request: Request):
    """Сохраняет проект"""
    try:
        data = await request.json()
        filename = data.get("filename")
        content = data.get("content")

        if not filename or not content:
            raise HTTPException(status_code=400, detail="Filename and content are required")

        path = get_project_path(filename)

        with open(path, "w", encoding="utf-8") as f:
            json.dump(content, f, indent=2)

        # Обновляем кэш сигналов
        refresh_signals_cache()

        print(f"[OK] Project saved: {filename}, signals cache refreshed: {len(STATE['signals'])} signals")
        return {"status": "ok", "message": f"Project saved to {filename}"}

    except HTTPException as e:
        raise e
    except Exception as e:
        print(f"Error saving project: {e}")
        raise HTTPException(status_code=500, detail="Internal server error during save")


@app.get("/api/project/load/{filename}")
def load_project(filename: str):
    """Загружает проект"""
    try:
        path = get_project_path(filename)

        if not os.path.exists(path):
            raise HTTPException(status_code=404, detail="Project not found")

        with open(path, "r", encoding="utf-8") as f:
            content = json.load(f)

        return content

```

```
    except HTTPException as e:
        raise e
    except Exception as e:
        print(f"Error loading project: {e}")
        raise HTTPException(status_code=500, detail="Internal server error during load")


# ============================================================================
# API — ДАННЫЕ СИГНАЛОВ
# ============================================================================

@app.post("/api/signal-data")
async def api_signal_data(request: Request):
    """Загружает данные сигналов из архива"""
    try:
        data = await request.json()
        signal_names = data.get("signal_names", [])
        output_format = data.get("format", "parquet")

        if not signal_names:
            raise HTTPException(status_code=400, detail="signal_names is required")

        folder = STATE["settings"].get("signalArchiveFolder")
        if not folder:
            raise HTTPException(status_code=500, detail="signalArchiveFolder not configured")

        signals_data = load_signal_data_optimized(signal_names, folder)

        response = {
            "found": list(signals_data.keys()),
            "not_found": [s for s in signal_names if s not in signals_data],
            "format": output_format
        }

        if not signals_data:
            raise HTTPException(status_code=404, detail="No signals found")

        if output_format == "parquet":
            return await _export_parquet(signals_data, response)
        else:
            return await _export_json(signals_data, response)

    except HTTPException as e:
        raise e
    except Exception as e:
        print(f"Error in api_signal_data: {e}")
        raise HTTPException(status_code=500, detail=str(e))


async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
    """Экспортирует данные в Parquet"""
    try:
        with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
            tmp_path = tmp.name

        rows = []
        for signal_name, df in signals_data.items():
            df_copy = df.copy()
            df_copy["signal_name"] = signal_name
            rows.append(df_copy)

        combined = pd.concat(rows, ignore_index=True)
        combined.to_parquet(tmp_path, compression='snappy', index=False)
```

```
695
696         file_size = os.path.getsize(tmp_path)
697         print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size /
    1024 / 1024:.2f} MB")
698
699         return FileResponse(
700             tmp_path,
701             media_type="application/octet-stream",
702             filename="signal_data.parquet",
703             headers={"X-Signal-Meta": json.dumps(meta)}
704         )
705     except Exception as e:
706         print(f"[ERROR] Parquet export failed: {e}")
707         raise
708
709
710 async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
711     """Экспортирует данные в JSON"""
712     try:
713         data_dict = {}
714         for signal_name, df in signals_data.items():
715             df_copy = df.copy()
716             df_copy["datetime"] = df_copy["datetime"].astype(str)
717             data_dict[signal_name] = df_copy.to_dict(orient="records")
718
719         response_data = {**meta, "data": data_dict}
720         return JSONResponse(response_data)
721     except Exception as e:
722         print(f"[ERROR] JSON export failed: {e}")
723         raise
724
725
726 @app.post("/api/resolve-signals")
727 async def api_resolve_signals(request: Request):
728     """Разворачивает зависимости сигналов (матрёшку)"""
729     try:
730         data = await request.json()
731         signal_names = data.get("signals", [])
732
733         print(f"[INFO] Resolving dependencies for signals: {signal_names}")
734
735         base_signals, synthetic_signals = resolve_signal_dependencies(signal_names)
736         computation_order = topological_sort_signals(synthetic_signals)
737
738         print(f"[INFO] Base signals: {base_signals}")
739         print(f"[INFO] Synthetic signals: {list(synthetic_signals.keys())}")
740         print(f"[INFO] Computation order: {computation_order}")
741
742         return {
743             "base_signals": list(base_signals),
744             "synthetic_signals": synthetic_signals,
745             "computation_order": computation_order
746         }
747
748     except ValueError as ve:
749         raise HTTPException(status_code=400, detail=str(ve))
750     except Exception as e:
751         print(f"[ERROR] resolve-signals failed: {e}")
752         raise HTTPException(status_code=500, detail=str(e))
753
754
755 # ============================================================================
756 # API — ВИЗУАЛИЗАТОР
757 # ============================================================================
758
```

```
@app.post("/api/visualize/session")
async def create_visualize_session(request: Request):
    """Создаёт сессию визуализации"""
    try:
        data = await request.json()
        signals = data.get("signals", [])
        code = data.get("code", "")
        visualizer_state = data.get("visualizer_state")

        if not isinstance(signals, list):
            raise HTTPException(status_code=400, detail="signals must be a list")

        token = uuid.uuid4().hex

        visualize_sessions[token] = {
            "signals": signals,
            "code": code,
            "visualizer_state": visualizer_state
        }

        print(f"[OK] Created visualize session: {token}, signals: {len(signals)}, has_state: {visualizer_state is not None}")

        return {"token": token}

    except HTTPException as e:
        raise e
    except Exception as e:
        print(f"[ERROR] create_visualize_session failed: {e}")
        raise HTTPException(status_code=500, detail=str(e))


@app.get("/api/visualize/session/{token}")
async def get_visualize_session(token: str):
    """Возвращает данные сессии визуализации"""
    session = visualize_sessions.get(token)

    if not session:
        raise HTTPException(status_code=404, detail="Session not found")

    return {
        "signals": session.get("signals", []),
        "code": session.get("code", ""),
        "visualizer_state": session.get("visualizer_state")
    }


@app.post("/api/visualize/save-state")
async def save_visualizer_state(request: VisualizerStateRequest) -> VisualizerStateResponse:
    """Сохраняет состояние визуализатора (вызывается из Streamlit)"""
    try:
        # Сохраняем состояние в сессию
        if request.session_token in visualize_sessions:
            visualize_sessions[request.session_token]["visualizer_state"] = request.state
        else:
            # Создаём новую запись если сессии нет
            visualize_sessions[request.session_token] = {
                "signals": [],
                "code": "",
                "visualizer_state": request.state
            }

        print(f"[OK] Saved visualizer state for session: {request.session_token}")
```

```
821
822         return VisualizerStateResponse(
823             success=True,
824             state=request.state,
825             message="Состояние сохранено"
826         )
827     except Exception as e:
828         print(f"[ERROR] save_visualizer_state failed: {e}")
829         return VisualizerStateResponse(
830             success=False,
831             message=f"Ошибка сохранения: {str(e)}"
832         )
833
834
835 @app.get("/api/visualize/get-state/{session_token}")
836 async def get_visualizer_state(session_token: str) -> VisualizerStateResponse:
837     """Возвращает состояние визуализатора (вызывается из редактора)"""
838     session = visualize_sessions.get(session_token)
839
840     if session is None:
841         return VisualizerStateResponse(
842             success=False,
843             message="Сессия не найдена"
844         )
845
846     state = session.get("visualizer_state")
847
848     if state is None:
849         return VisualizerStateResponse(
850             success=False,
851             message="Состояние визуализатора не сохранено"
852         )
853
854     return VisualizerStateResponse(
855         success=True,
856         state=state
857     )
858
859
860 # ============================================================================
861 # СТАТИЧЕСКИЕ ФАЙЛЫ (ФРОНТЕНД)
862 # ============================================================================
863
864 WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
865 app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")
866
867 # server/update_projects.py
868 import os, json, hashlib, subprocess, tempfile, shutil # Импортируем shutil
869
870 BASE_DIR = os.path.dirname(os.path.abspath(__file__))
871 HASH_PATH = os.path.join(BASE_DIR, ".formula_templates.hash")
872 NODE_SCRIPT = os.path.normpath(os.path.join(BASE_DIR, "..", "web", "js",
    "regenerate_code.js"))
873
874 def _file_hash(path: str) -> str:
875     h = hashlib.sha256()
876     with open(path, "rb") as f:
877         for chunk in iter(lambda: f.read(1024 * 1024), b""):
878             h.update(chunk)
879     return h.hexdigest()
880
881 def update_projects_if_templates_changed(project_dir: str, templates_path: str):
882     if not os.path.isdir(project_dir):
883         print(f"[WARN] project dir not found: {project_dir}")
884         return
```

```
885      if not os.path.exists(templates_path):
886          print(f"[WARN] templates not found: {templates_path}")
887          return
888      if not os.path.exists(NODE_SCRIPT):
889          raise RuntimeError(f"Node script not found: {NODE_SCRIPT}")
890
891      new_hash = _file_hash(templates_path)
892      old_hash = None
893      if os.path.exists(HASH_PATH):
894          with open(HASH_PATH, "r", encoding="utf-8") as f:
895              old_hash = f.read().strip()
896
897      if new_hash == old_hash:
898          print("[OK] formula_templates.json not changed")
899          return
900
901      print("[INFO] Templates changed -> regenerating project codes...")
902
903      for fname in os.listdir(project_dir):
904          if not fname.endswith(".json"):
905              continue
906          path = os.path.join(project_dir, fname)
907          try:
908              result = subprocess.run(
909                  ["node", NODE_SCRIPT, path, templates_path],
910                  check=True,
911                  capture_output=True,
912                  text=True
913              )
914
915              if not result.stdout.strip():
916                  print(f"[ERROR] node returned empty stdout for {fname}")
917                  if result.stderr.strip():
918                      print(f"[STDERR]\n{result.stderr}")
919                  raise ValueError(f"Empty stdout from node script for {fname}")
920
921              updated = json.loads(result.stdout)
922
923              # Используем NamedTemporaryFile с директорией, если она доступна,
924              # или полагаемся на стандартное поведение (создание на том же разделе)
925              # Если не помогает, нужно явно указывать tempfile.TemporaryDirectory
926              with tempfile.NamedTemporaryFile(mode="w", delete=False, encoding="utf-8",
    dir=project_dir) as tmp:
927                  json.dump(updated, tmp, ensure_ascii=False, indent=2)
928                  tmp_path = tmp.name
929
930              try:
931                  # Копируем файл, а потом удаляем временный
932                  shutil.copy2(tmp_path, path) # copy2 сохраняет метаданные
933                  os.remove(tmp_path)
934                  print(f"[OK] updated {fname}")
935              except Exception as e:
936                  print(f"[ERROR] failed to copy/remove temporary file for {fname}:
    {e}")
937                  # Попытка очистки временного файла, если он остался
938                  if os.path.exists(tmp_path):
939                      os.remove(tmp_path)
940                  raise
941
942          except subprocess.CalledProcessError as e:
943              print(f"[ERROR] node failed for {fname}: exit code {e.returncode}")
944              if e.stderr:
945                  print(f"[STDERR]\n{e.stderr}")
946              if e.stdout:
947                  print(f"[STDOUT]\n{e.stdout}")
```

```
            raise
        except json.JSONDecodeError as e:
            print(f"[ERROR] json parse failed for {fname}: {e}")
            print(f"[STDOUT]\n{result.stdout[:500]}")
            if result.stderr:
                print(f"[STDERR]\n{result.stderr}")
            raise
        except Exception as e:
            print(f"[ERROR] failed {fname}: {e}")
            raise

    with open(HASH_PATH, "w", encoding="utf-8") as f:
        f.write(new_hash)

    print("[OK] All projects regenerated.")

    # visualizer_app.py — с поддержкой сохранения/загрузки состояния

import pandas as pd
import requests
import streamlit as st
import plotly.express as px
import numpy as np
import plotly.graph_objects as go
from typing import List
from datetime import datetime, time

from code_signal import compute_code_signal, sanitize_numeric_column
from visualizer_state import (
    create_visualizer_state,
    load_visualizer_state,
    STATE_VERSION
)

st.set_page_config(page_title="Signal Visualizer", layout="wide")
st.title("📊 Визуализация сигналов")

query_params = st.query_params
session_token = query_params.get("session", None)
api_url = query_params.get("api_url", "http://localhost:8000")

signal_codes = query_params.get("signals", [])
if isinstance(signal_codes, str):
    signal_codes = [signal_codes]

CODE = ""
INITIAL_VISUALIZER_STATE = None  # Состояние из проекта

if session_token:
    try:
        resp = requests.get(f"{api_url}/api/visualize/session/{session_token}")
        resp.raise_for_status()
        payload = resp.json()
        signal_codes = payload.get("signals", signal_codes)
        CODE = payload.get("code", CODE)
        INITIAL_VISUALIZER_STATE = payload.get("visualizer_state")  # НОВОЕ
    except Exception as e:
        st.error(f"Не удалось получить данные сессии: {e}")

# === ИНИЦИАЛИЗАЦИЯ SESSION STATE ===
if "signals_data" not in st.session_state:
    st.session_state.signals_data = None
if "selected_signals" not in st.session_state:
    st.session_state.selected_signals = set()
if "plot_areas" not in st.session_state:
```

```
    st.session_state.plot_areas = []
if "derived_signals" not in st.session_state:
    st.session_state.derived_signals = {}
if "code_signal_name" not in st.session_state:
    st.session_state.code_signal_name = None
if "synthetic_computed" not in st.session_state:
    st.session_state.synthetic_computed = {}
if "signal_groups" not in st.session_state:
    st.session_state.signal_groups = {"project": set(), "dependencies": set()}
if "global_cursor_time" not in st.session_state:
    st.session_state.global_cursor_time = None
# НОВОЕ: флаг что состояние уже загружено (чтобы не перезаписывать при rerun)
if "state_loaded" not in st.session_state:
    st.session_state.state_loaded = False
# НОВОЕ: флаг что есть несохранённые изменения
if "has_unsaved_changes" not in st.session_state:
    st.session_state.has_unsaved_changes = False


def mark_unsaved():
    """Помечает что есть несохранённые изменения"""
    st.session_state.has_unsaved_changes = True


def load_base_signals_data(signal_names: List[str]) -> pd.DataFrame | None:
    """Загружает данные базовых сигналов из архива"""
    if not signal_names:
        return None

    try:
        response = requests.post(
            f"{api_url}/api/signal-data",
            json={"signal_names": signal_names, "format": "json"},
        )
        response.raise_for_status()
        result = response.json()

        found = result.get("found", [])
        not_found = result.get("not_found", [])
        data_dict = result.get("data", {})

        if not_found:
            st.warning(f"⚠️ Базовые сигналы не найдены в архиве: {',
'.join(not_found)}")

        if not data_dict:
            return None

        frames = []
        for sig, records in data_dict.items():
            if not records:
                continue
            df = pd.DataFrame(records)
            if "datetime" not in df or "value" not in df:
                continue
            df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
            df = df.dropna(subset=["datetime"])
            df = df.set_index("datetime").sort_index()
            df = df.rename(columns={"value": sig})
            frames.append(df[[sig]])

        if not frames:
            return None

        return pd.concat(frames, axis=1).sort_index()
```

```
1077
1078      except Exception as exc:
1079          st.error(f"❌ Ошибка загрузки базовых сигналов: {exc}")
1080          return None
1081
1082
1083  def resolve_and_load_all_signals(input_signals: List[str]) -> tuple[pd.DataFrame |
      None, List[str], List[str]]:
1084      if not input_signals:
1085          return None, [], []
1086
1087      try:
1088          with st.spinner("🔍 Разворачиваем зависимости сигналов..."):
1089              resolve_resp = requests.post(
1090                  f"{api_url}/api/resolve-signals",
1091                  json={"signals": input_signals}
1092              )
1093              resolve_resp.raise_for_status()
1094              resolve_data = resolve_resp.json()
1095
1096          base_signals = resolve_data.get("base_signals", [])
1097          synthetic_signals = resolve_data.get("synthetic_signals", {})
1098          computation_order = resolve_data.get("computation_order", [])
1099
1100          project_signals = set(input_signals)
1101          dependency_signals = set()
1102          for syn_name, syn_data in synthetic_signals.items():
1103              if syn_name not in project_signals:
1104                  dependency_signals.add(syn_name)
1105              for dep in syn_data.get("dependencies", []):
1106                  if dep not in project_signals:
1107                      dependency_signals.add(dep)
1108
1109          for bs in base_signals:
1110              if bs not in project_signals:
1111                  dependency_signals.add(bs)
1112
1113          st.session_state.signal_groups = {
1114              "project": project_signals,
1115              "dependencies": dependency_signals
1116          }
1117
1118          st.info(f"📊 Сигналов проекта: {len(project_signals)} | Из зависимостей:
      {len(dependency_signals)}")
1119
1120          if synthetic_signals:
1121              with st.expander("🔗 Граф зависимостей синтетических сигналов"):
1122                  for syn_name in computation_order:
1123                      deps = synthetic_signals[syn_name].get("dependencies", [])
1124                      marker = "📌" if syn_name in project_signals else "🔗"
1125                      st.text(f"  {marker} {syn_name} ← {deps}")
1126
1127          df_all = None
1128          found_signals = []
1129          not_found_signals = []
1130
1131          if base_signals:
1132              with st.spinner(f"📥 Загружаем {len(base_signals)} базовых сигналов..."):
1133                  df_all = load_base_signals_data(base_signals)
1134                  if df_all is not None:
1135                      found_signals = list(df_all.columns)
1136                      not_found_signals = [s for s in base_signals if s not in
      df_all.columns]
1137
1138          if df_all is None:
```

```
            df_all = pd.DataFrame()

        if computation_order:
            with st.spinner(f"⚙️ Вычисляем {len(computation_order)} синтетических сигналов..."):
                progress_bar = st.progress(0)

                for idx, syn_name in enumerate(computation_order):
                    syn_data = synthetic_signals[syn_name]
                    formula = syn_data.get("formula", "")

                    if not formula:
                        st.warning(f"⚠️ Синтетический сигнал '{syn_name}' не имеет формулы")
                        continue

                    if df_all.empty:
                        st.warning(f"⚠️ Нет данных для вычисления '{syn_name}'")
                        continue

                    try:
                        syn_series = compute_code_signal(
                            formula,
                            df_all,
                            warn_callback=lambda msg, name=syn_name: st.warning(f"[{name}] {msg}", icon="⚠️")
                        )
                        syn_series.name = syn_name
                        df_all[syn_name] = syn_series
                        found_signals.append(syn_name)
                        st.session_state.synthetic_computed[syn_name] = formula

                    except Exception as e:
                        st.error(f"❌ Ошибка вычисления '{syn_name}': {e}")
                        not_found_signals.append(syn_name)

                    progress_bar.progress((idx + 1) / len(computation_order))

                progress_bar.empty()

        return df_all if not df_all.empty else None, found_signals, not_found_signals

    except requests.exceptions.HTTPError as http_err:
        error_detail = ""
        try:
            error_detail = http_err.response.json().get("detail", "")
        except:
            pass
        st.error(f"❌ Ошибка API: {error_detail or http_err}")
        return None, [], []
    except Exception as exc:
        st.error(f"❌ Ошибка загрузки данных: {exc}")
        import traceback
        st.code(traceback.format_exc())
        return None, [], []


# ========== ЗАГРУЗКА ДАННЫХ ==========
if signal_codes and st.session_state.signals_data is None:
    df_base, found_codes, not_found_codes = resolve_and_load_all_signals(signal_codes)
    st.session_state.signals_data = df_base

    if found_codes:
        st.success(f"✅ Загружено сигналов: {len(found_codes)}")
    if not_found_codes:
```

```
        st.warning(f"⚠️ Не найдены: {', '.join(not_found_codes)}")


def get_all_signals_df(exclude: set[str] | None = None):
    exclude = exclude or set()
    base = st.session_state.signals_data
    derived = st.session_state.derived_signals

    dfs = []
    if base is not None:
        dfs.append(base)
    for name, ddf in derived.items():
        if name in exclude:
            continue
        dfs.append(ddf)

    if not dfs:
        return None
    return pd.concat(dfs, axis=1).sort_index()


def compute_stats_numeric(df: pd.DataFrame) -> pd.DataFrame:
    if df is None or df.empty:
        return pd.DataFrame()

    numeric = df.apply(sanitize_numeric_column)
    valid_cols = [col for col in numeric.columns if numeric[col].count() > 0]
    if not valid_cols:
        return pd.DataFrame()

    numeric = numeric[valid_cols]
    stats = pd.DataFrame(index=numeric.columns)
    stats["count"] = numeric.count()
    stats["min"] = numeric.min()
    stats["max"] = numeric.max()
    stats["mean"] = numeric.mean()
    stats["std"] = numeric.std()
    stats["median"] = numeric.median()

    starts, ends = [], []
    for col in numeric.columns:
        series = numeric[col].dropna()
        starts.append(series.index.min() if not series.empty else pd.NaT)
        ends.append(series.index.max() if not series.empty else pd.NaT)

    stats["start"] = starts
    stats["end"] = ends
    return stats


def make_unique_name(base_name: str) -> str:
    existing = set()
    if st.session_state.signals_data is not None:
        existing |= set(st.session_state.signals_data.columns)
    existing |= set(st.session_state.derived_signals.keys())
    if base_name not in existing:
        return base_name
    idx = 2
    while f"{base_name}_{idx}" in existing:
        idx += 1
    return f"{base_name}_{idx}"


# --- синтетический сигнал из CODE ---
code_signal_name = st.session_state.code_signal_name
```

```
df_for_code = get_all_signals_df(exclude={code_signal_name} if code_signal_name else None)
code_key = (session_token, CODE)

already_have_series = (
    st.session_state.code_signal_name is not None
    and st.session_state.code_signal_name in st.session_state.derived_signals
)

if CODE and df_for_code is not None:
    need_recalc = (st.session_state.get("code_key") != code_key) or (not already_have_series)

    if need_recalc:
        try:
            synthetic_series = compute_code_signal(
                CODE,
                df_for_code,
                warn_callback=lambda msg: st.warning(msg, icon="⚠️"),
            )
            target_name = code_signal_name or make_unique_name("CODE_RESULT")
            synthetic_series.name = target_name

            st.session_state.derived_signals[target_name] = pd.DataFrame({target_name: synthetic_series})
            st.session_state.code_signal_name = target_name
            st.session_state.selected_signals.add(target_name)

            st.session_state.code_key = code_key
            st.success(f"Синтетический сигнал обновлён: {target_name}")
        except Exception as exc:
            st.warning(f"Не удалось вычислить CODE: {exc}")

elif not CODE:
    if code_signal_name:
        st.session_state.derived_signals.pop(code_signal_name, None)
        st.session_state.selected_signals.discard(code_signal_name)
        st.session_state.code_signal_name = None
    st.session_state.code_key = None


# === ЗАГРУЗКА СОХРАНЁННОГО СОСТОЯНИЯ (один раз) ===
df_all_signals = get_all_signals_df()

if not st.session_state.state_loaded and INITIAL_VISUALIZER_STATE and df_all_signals is not None:
    available_signals = set(df_all_signals.columns.tolist())

    loaded_selected, loaded_areas, load_warnings = load_visualizer_state(
        INITIAL_VISUALIZER_STATE,
        available_signals
    )

    # Применяем загруженное состояние
    if loaded_selected:
        st.session_state.selected_signals = loaded_selected
    if loaded_areas:
        st.session_state.plot_areas = loaded_areas

    # Показываем предупреждения
    for warn in load_warnings:
        st.warning(f"⚠️ {warn}")

    if loaded_selected or loaded_areas:
        st.info("📂 Загружено сохранённое состояние визуализатора")
```

```
1327
1328      st.session_state.state_loaded = True
1329      st.session_state.has_unsaved_changes = False
1330
1331
1332  # === ФУНКЦИЯ СОХРАНЕНИЯ СОСТОЯНИЯ ===
1333  def save_current_state():
1334      """Сохраняет текущее состояние на сервер"""
1335      if not session_token:
1336          st.error("Нет токена сессии для сохранения")
1337          return False
1338
1339      state = create_visualizer_state(
1340          st.session_state.selected_signals,
1341          st.session_state.plot_areas
1342      )
1343
1344      try:
1345          resp = requests.post(
1346              f"{api_url}/api/visualize/save-state",
1347              json={
1348                  "session_token": session_token,
1349                  "state": state
1350              }
1351          )
1352          resp.raise_for_status()
1353          result = resp.json()
1354
1355          if result.get("success"):
1356              st.session_state.has_unsaved_changes = False
1357              return True
1358          else:
1359              st.error(f"Ошибка сохранения: {result.get('message')}")
1360              return False
1361      except Exception as e:
1362          st.error(f"Ошибка сохранения состояния: {e}")
1363          return False
1364
1365
1366  # === SIDEBAR ===
1367  with st.sidebar:
1368      st.header("Выбор сигналов")
1369
1370      # НОВОЕ: Кнопка сохранения состояния
1371      if session_token:
1372          save_col1, save_col2 = st.columns([2, 1])
1373          with save_col1:
1374              if st.button("💾 Сохранить состояние", use_container_width=True):
1375                  if save_current_state():
1376                      st.success("✅ Состояние сохранено!")
1377                      st.info("💡 Теперь сохраните проект в редакторе")
1378          with save_col2:
1379              if st.session_state.has_unsaved_changes:
1380                  st.markdown("🔴 *Изменения*")
1381              else:
1382                  st.markdown("🟢 *Сохранено*")
1383          st.divider()
1384
1385      if df_all_signals is not None:
1386          available_signals = df_all_signals.columns.tolist()
1387
1388          signal_groups = st.session_state.get("signal_groups", {
1389              "project": set(available_signals),
1390              "dependencies": set()
1391          })
```

```
1392
1393            project_signals = [s for s in available_signals if s in
    signal_groups.get("project", set())]
1394            dependency_signals = [s for s in available_signals if s in
    signal_groups.get("dependencies", set())]
1395
1396            if project_signals:
1397                st.subheader("📌 Сигналы проекта")
1398                for signal in project_signals:
1399                    is_synthetic = signal in st.session_state.get("synthetic_computed",
    {})
1400                    label = f"⚙️ {signal}" if is_synthetic else signal
1401
1402                    checked = st.checkbox(
1403                        label,
1404                        value=(signal in st.session_state.selected_signals),
1405                        key=f"proj_{signal}"
1406                    )
1407                    if checked and signal not in st.session_state.selected_signals:
1408                        st.session_state.selected_signals.add(signal)
1409                        mark_unsaved()
1410                    elif not checked and signal in st.session_state.selected_signals:
1411                        st.session_state.selected_signals.discard(signal)
1412                        mark_unsaved()
1413
1414            if dependency_signals:
1415                st.divider()
1416                with st.expander(f"🔗 Из зависимостей ({len(dependency_signals)})",
    expanded=False):
1417                    for signal in dependency_signals:
1418                        is_synthetic = signal in
    st.session_state.get("synthetic_computed", {})
1419                        label = f"⚙️ {signal}" if is_synthetic else signal
1420
1421                        checked = st.checkbox(
1422                            label,
1423                            value=(signal in st.session_state.selected_signals),
1424                            key=f"dep_{signal}"
1425                        )
1426                        if checked and signal not in st.session_state.selected_signals:
1427                            st.session_state.selected_signals.add(signal)
1428                            mark_unsaved()
1429                        elif not checked and signal in st.session_state.selected_signals:
1430                            st.session_state.selected_signals.discard(signal)
1431                            mark_unsaved()
1432
1433            st.divider()
1434            col1, col2 = st.columns(2)
1435            with col1:
1436                if st.button("✅ Все проекта"):
1437                    st.session_state.selected_signals.update(project_signals)
1438                    mark_unsaved()
1439                    st.rerun()
1440            with col2:
1441                if st.button("❌ Снять все"):
1442                    st.session_state.selected_signals.clear()
1443                    mark_unsaved()
1444                    st.rerun()
1445
1446            st.divider()
1447            st.subheader("Создать обрезанный сигнал")
1448
1449            base_df = st.session_state.signals_data
1450            if base_df is not None and not base_df.empty:
1451                base_choice = st.selectbox("Исходный сигнал", base_df.columns)
```

```
1452                series = base_df[base_choice].dropna()
1453                if not series.empty:
1454                    col1, col2 = st.columns(2)
1455                    with col1:
1456                        start_date = st.date_input(
1457                            "Начало",
1458                            value=series.index.min().date(),
1459                        )
1460                    with col2:
1461                        end_date = st.date_input(
1462                            "Конец",
1463                            value=series.index.max().date(),
1464                        )
1465
1466                    start_ts = pd.Timestamp(start_date)
1467                    end_ts = pd.Timestamp(end_date) + pd.Timedelta(days=1) - pd.Timedelta(
1468                        microseconds=1
1469                    )
1470
1471                    default_name = f"{base_choice}__{start_ts.date()}_{end_ts.date()}"
1472                    new_name = st.text_input("Имя нового сигнала", value=default_name)
1473
1474                    col3, col4 = st.columns(2)
1475                    if col3.button("Создать"):
1476                        name_unique = make_unique_name(new_name.strip())
1477                        cut_series = series[(series.index >= start_ts) & (series.index <=
     end_ts)]
1478                        if cut_series.empty:
1479                            st.warning("В выбранном диапазоне нет точек.")
1480                        else:
1481                            st.session_state.derived_signals[name_unique] = pd.DataFrame(
1482                                {name_unique: cut_series}
1483                            )
1484                            st.success(f"Создан обрезанный сигнал: {name_unique}")
1485                            st.rerun()
1486                    if col4.button("Очистить все обрезанные"):
1487                        st.session_state.derived_signals = {
1488                            k: v
1489                            for k, v in st.session_state.derived_signals.items()
1490                            if k == st.session_state.code_signal_name
1491                        }
1492                        st.session_state.selected_signals = {
1493                            sig
1494                            for sig in st.session_state.selected_signals
1495                            if (st.session_state.signals_data is not None and sig in
     st.session_state.signals_data.columns)
1496                            or sig == st.session_state.code_signal_name
1497                        }
1498                        st.rerun()
1499
1500            if st.session_state.derived_signals:
1501                st.subheader("Удалить обрезанный/синтетический сигнал")
1502                derived_names = [name for name in st.session_state.derived_signals.keys()]
1503                delete_candidate = st.selectbox("Выберите", ["—"] + derived_names)
1504                if st.button("Удалить выбранный") and delete_candidate != "—":
1505                    st.session_state.derived_signals.pop(delete_candidate, None)
1506                    st.session_state.selected_signals.discard(delete_candidate)
1507                    if delete_candidate == st.session_state.code_signal_name:
1508                        st.session_state.code_signal_name = None
1509                    st.rerun()
1510
1511            st.divider()
1512            st.subheader("Области построения")
1513            col_a, col_b = st.columns(2)
1514            if col_a.button("➕ Добавить график"):
```

```
            new_id = max([area.get("id", 0) for area in st.session_state.plot_areas] + [0]) + 1
            st.session_state.plot_areas.append({
                "id": new_id,
                "signals": [],
                "shapes": [],
                "cursor_time": None,
                "x_range": None,
                "y_range": None
            })
            mark_unsaved()
            st.rerun()
        if col_b.button("❌ Очистить все"):
            st.session_state.plot_areas = []
            st.session_state.selected_signals = set()
            st.session_state.global_cursor_time = None
            mark_unsaved()
            st.rerun()
    else:
        st.info("📥 Данные сигналов еще не загружены.")


def find_nearest_index_in_range(valid_index, target_time, x_start, x_end):
    """Находит ближайший индекс в заданном диапазоне"""
    mask = (valid_index >= x_start) & (valid_index <= x_end)
    filtered_index = valid_index[mask]

    if len(filtered_index) == 0:
        return 0, valid_index[0] if len(valid_index) > 0 else None

    if target_time is None:
        return 0, filtered_index[0]

    diffs = abs((filtered_index - pd.to_datetime(target_time)).total_seconds())
    min_pos = diffs.argmin()
    return min_pos, filtered_index[min_pos]


# === ОСНОВНАЯ ОБЛАСТЬ ГРАФИКОВ ===
if df_all_signals is not None and st.session_state.selected_signals:
    if not st.session_state.plot_areas:
        st.session_state.plot_areas.append({
            "id": 1,
            "signals": list(st.session_state.selected_signals),
            "shapes": [],
            "cursor_time": None,
            "x_range": None,
            "y_range": None
        })

    for i, plot_area in enumerate(st.session_state.plot_areas):
        with st.container():
            col1, col2 = st.columns([3, 1])
            with col1:
                st.subheader(f"График #{plot_area['id']}")
            with col2:
                if st.button("Удалить", key=f"remove_area_{i}"):
                    st.session_state.plot_areas.pop(i)
                    mark_unsaved()
                    st.rerun()

            selected = st.multiselect(
                "Выберите сигнал(ы):",
                list(st.session_state.selected_signals),
                default=plot_area.get("signals", []),
```

```
1579                 key=f"signals_sel_{i}",
1580             )
1581 
1582             # Проверяем изменились ли сигналы
1583             if set(selected) != set(plot_area.get("signals", [])):
1584                 mark_unsaved()
1585             st.session_state.plot_areas[i]["signals"] = selected
1586 
1587             if selected:
1588                 df_plot = df_all_signals[selected].copy()
1589                 df_plot_num = df_plot.apply(sanitize_numeric_column)
1590 
1591                 valid_index = df_plot_num.dropna(how="all").index
1592                 if len(valid_index) == 0:
1593                     st.warning("Нет числовых данных для выбранных сигналов.")
1594                 else:
1595                     full_x_min = valid_index.min()
1596                     full_x_max = valid_index.max()
1597 
1598                     y_data = df_plot_num.values.flatten()
1599                     y_data = y_data[~np.isnan(y_data)]
1600                     full_y_min = float(y_data.min()) if len(y_data) > 0 else 0.0
1601                     full_y_max = float(y_data.max()) if len(y_data) > 0 else 1.0
1602 
1603                     y_padding = (full_y_max - full_y_min) * 0.05
1604                     full_y_min -= y_padding
1605                     full_y_max += y_padding
1606 
1607                     if plot_area.get('x_range') is None:
1608                         plot_area['x_range'] = [full_x_min, full_x_max]
1609 
1610                     if plot_area.get('y_range') is None:
1611                         plot_area['y_range'] = [full_y_min, full_y_max]
1612 
1613                     x_start_ts, x_end_ts = plot_area['x_range']
1614                     mask_visible = (valid_index >= x_start_ts) & (valid_index <=
     x_end_ts)
1615                     visible_index = valid_index[mask_visible]
1616 
1617                     if len(visible_index) == 0:
1618                         st.warning("В выбранном диапазоне X нет данных.")
1619                     else:
1620                         if plot_area.get('cursor_time') is None:
1621                             plot_area['cursor_time'] =
     visible_index[len(visible_index) // 2]
1622 
1623                         cursor_time = plot_area['cursor_time']
1624                         if cursor_time < x_start_ts or cursor_time > x_end_ts:
1625                             cursor_time = visible_index[len(visible_index) // 2]
1626                             plot_area['cursor_time'] = cursor_time
1627 
1628                         cursor_pos, _ = find_nearest_index_in_range(
1629                             visible_index, cursor_time, x_start_ts, x_end_ts
1630                         )
1631 
1632                         if st.session_state.global_cursor_time is not None:
1633                             global_cursor = st.session_state.global_cursor_time
1634                             if x_start_ts <= global_cursor <= x_end_ts:
1635                                 cursor_pos, cursor_time = find_nearest_index_in_range(
1636                                     visible_index, global_cursor, x_start_ts, x_end_ts
1637                                 )
1638                                 plot_area['cursor_time'] = cursor_time
1639 
1640                         ts_idx = st.slider(
1641                             "📍 Вертикальная линия (в видимом диапазоне)",
```

```
                            min_value=0,
                            max_value=len(visible_index) - 1,
                            value=min(cursor_pos, len(visible_index) - 1),
                            key=f"vline_slider_{i}",
                            help="Слайдер работает только в рамках текущего видимого диапазона X"
                        )

                        ts = visible_index[ts_idx]
                        plot_area['cursor_time'] = ts

                        col_pos, col_sync = st.columns([3, 1])
                        with col_pos:
                            st.markdown(f"**📅 Позиция линии:** `{ts.strftime('%Y-%m-%d %H:%M:%S')}`")
                        with col_sync:
                            if st.button("🔄 Синхронизировать все", key=f"sync_{i}"):
                                st.session_state.global_cursor_time = ts
                                for pa in st.session_state.plot_areas:
                                    pa['cursor_time'] = ts
                                st.rerun()

                        fig = px.line(
                            df_plot_num,
                            x=df_plot_num.index,
                            y=selected,
                            title=f"График #{plot_area['id']}",
                            render_mode="webgl"
                        )

                        fig.add_vline(x=ts, line_width=2, line_dash="dash", line_color="red")

                        shapes = plot_area.get('shapes', [])
                        for shape in shapes:
                            if shape['type'] == 'vline':
                                fig.add_vline(x=shape['x'], line_dash=shape['dash'], line_color=shape['color'], line_width=1)
                            elif shape['type'] == 'hline':
                                fig.add_hline(y=shape['y'], line_dash=shape['dash'], line_color=shape['color'], line_width=1)

                        fig.update_layout(
                            uirevision=f"plot_area_{plot_area['id']}",
                            height=600,
                            legend_title_text="Сигналы",
                            xaxis_title="Время",
                            yaxis_title="Значение",
                            margin=dict(l=20, r=20, t=40, b=20),
                            xaxis=dict(
                                range=[x_start_ts, x_end_ts],
                                rangeslider=dict(
                                    visible=True,
                                    thickness=0.08,
                                    bgcolor='#e0e0e0',
                                    range=[full_x_min, full_x_max]
                                )
                            ),
                            yaxis=dict(
                                range=plot_area['y_range'],
                                fixedrange=False
                            )
                        )

                        st.plotly_chart(fig, use_container_width=True)
```

```
1702
1703                         with st.expander(f"📍 Добавить маркеры для графика
     #{plot_area['id']}"):
1704                             col_x, col_y = st.columns(2)
1705                             with col_x:
1706                                 st.markdown("**Вертикальная линия (X)**")
1707                                 x_date = st.date_input("Дата", value=ts.date(),
     key=f"x_date_{i}")
1708                                 x_time = st.time_input("Время", value=ts.time(),
     key=f"x_time_{i}")
1709                                 x_full = pd.Timestamp.combine(x_date, x_time)
1710                                 if st.button("Добавить V-line", key=f"add_vline_{i}"):
1711                                     shapes.append({
1712                                         'type': 'vline',
1713                                         'x': x_full,
1714                                         'dash': 'dot',
1715                                         'color': 'blue'
1716                                     })
1717                                     plot_area['shapes'] = shapes
1718                                     mark_unsaved()
1719                                     st.success(f"Добавлена линия на {x_full}")
1720                                     st.rerun()
1721
1722                             with col_y:
1723                                 st.markdown("**Горизонтальная линия (Y)**")
1724                                 y_value = st.number_input("Значение Y", value=0.0,
     key=f"y_val_{i}")
1725                                 if st.button("Добавить H-line", key=f"add_hline_{i}"):
1726                                     shapes.append({
1727                                         'type': 'hline',
1728                                         'y': y_value,
1729                                         'dash': 'dash',
1730                                         'color': 'green'
1731                                     })
1732                                     plot_area['shapes'] = shapes
1733                                     mark_unsaved()
1734                                     st.success(f"Добавлена линия на Y={y_value}")
1735                                     st.rerun()
1736
1737                             if shapes:
1738                                 st.markdown("**Текущие маркеры:**")
1739                                 for j, s in enumerate(shapes):
1740                                     if s['type'] == 'vline':
1741                                         st.text(f"  V-line: {s['x']} ({s['color']})")
1742                                     else:
1743                                         st.text(f"  H-line: Y={s['y']}
     ({s['color']})")
1744                                 if st.button(f"🗑️ Очистить маркеры",
     key=f"clear_shapes_{i}"):
1745                                     plot_area['shapes'] = []
1746                                     mark_unsaved()
1747                                     st.rerun()
1748
1749                         nearest =
     df_plot_num.reindex(df_plot_num.index.union([ts])).sort_index()
1750                         nearest = nearest.ffill().loc[ts]
1751
1752                         st.markdown("**📊 Статистика:**")
1753                         stats_df = compute_stats_numeric(df_plot)
1754                         if stats_df.empty:
1755                             st.info("Нет данных для статистики.")
1756                         else:
1757                             stats_view = stats_df.copy()
1758                             stats_view["value"] = nearest.reindex(stats_view.index)
1759                             stats_view["start"] = pd.to_datetime(stats_view["start"],
```

```
    errors="coerce").dt.strftime("%Y-%m-%d %H:%M:%S")
                                stats_view["end"] = pd.to_datetime(stats_view["end"],
    errors="coerce").dt.strftime("%Y-%m-%d %H:%M:%S")
                                st.dataframe(
                                    stats_view.style.format(
                                        {
                                            "count": "{:.0f}",
                                            "min": "{:.6g}",
                                            "max": "{:.6g}",
                                            "mean": "{:.6g}",
                                            "std": "{:.6g}",
                                            "median": "{:.6g}",
                                            "value": "{:.6g}",
                                        },
                                        na_rep="",
                                    ),
                                    use_container_width=True,
                                )
            else:
                st.info("Выберите сигналы для отображения.")
        st.divider()

elif df_all_signals is None:
    st.info("📥 Данные сигналов ещё не загружены.")
else:
    st.info("👈 Выберите сигналы слева для визуализации.")

if df_all_signals is not None:
    with st.expander("ℹ️ Информация о данных"):
        col1, col2, col3 = st.columns(3)
        with col1:
            st.metric("Всего сигналов", len(df_all_signals.columns))
        with col2:
            st.metric("Количество записей", len(df_all_signals))
        with col3:
            try:
                dt_range = df_all_signals.index.max() - df_all_signals.index.min()
                st.metric("Диапазон времени", str(dt_range).split(".")[0])
            except Exception:
                st.metric("Диапазон времени", "—")

if CODE:
    with st.expander("🧩 Сгенерированный код"):
        st.code(CODE, language="text")

# visualizer_state.py — модуль для сериализации/десериализации состояния визуализатора

import json
from datetime import datetime
from typing import Any, Dict, List, Optional, Set
import pandas as pd


# Версия формата состояния (для обратной совместимости в будущем)
STATE_VERSION = 1


def serialize_timestamp(ts) -> Optional[str]:
    """Конвертирует Timestamp/datetime в ISO строку"""
    if ts is None:
        return None
    if isinstance(ts, str):
        return ts
    if isinstance(ts, (datetime, pd.Timestamp)):
        return ts.isoformat()
```

```
    return str(ts)


def deserialize_timestamp(ts_str: Optional[str]) -> Optional[pd.Timestamp]:
    """Конвертирует ISO строку в Timestamp"""
    if ts_str is None:
        return None
    try:
        return pd.Timestamp(ts_str)
    except Exception:
        return None


def serialize_shape(shape: Dict[str, Any]) -> Dict[str, Any]:
    """Сериализует один маркер (shape) для JSON"""
    result = {
        'type': shape.get('type'),
        'dash': shape.get('dash', 'solid'),
        'color': shape.get('color', 'gray')
    }

    if shape.get('type') == 'vline':
        result['x'] = serialize_timestamp(shape.get('x'))
    elif shape.get('type') == 'hline':
        result['y'] = shape.get('y')

    return result


def deserialize_shape(shape_data: Dict[str, Any]) -> Optional[Dict[str, Any]]:
    """Десериализует маркер из JSON"""
    shape_type = shape_data.get('type')

    if shape_type not in ('vline', 'hline'):
        return None

    result = {
        'type': shape_type,
        'dash': shape_data.get('dash', 'solid'),
        'color': shape_data.get('color', 'gray')
    }

    if shape_type == 'vline':
        ts = deserialize_timestamp(shape_data.get('x'))
        if ts is None:
            return None
        result['x'] = ts
    elif shape_type == 'hline':
        y_val = shape_data.get('y')
        if y_val is None:
            return None
        result['y'] = float(y_val)

    return result


def serialize_plot_area(plot_area: Dict[str, Any]) -> Dict[str, Any]:
    """Сериализует одну область графика"""
    return {
        'id': plot_area.get('id', 1),
        'signals': list(plot_area.get('signals', [])),
        'shapes': [serialize_shape(s) for s in plot_area.get('shapes', [])],
        # cursor_time и диапазоны НЕ сохраняем — они пересчитываются
    }

```

```
1888
1889  def deserialize_plot_area(
1890      area_data: Dict[str, Any],
1891      available_signals: Set[str]
1892  ) -> Optional[Dict[str, Any]]:
1893      """
1894      Десериализует область графика.
1895      Фильтрует сигналы, которых нет в available_signals.
1896      """
1897      area_id = area_data.get('id', 1)
1898
1899      # Фильтруем сигналы — оставляем только существующие
1900      raw_signals = area_data.get('signals', [])
1901      valid_signals = [s for s in raw_signals if s in available_signals]
1902
1903      # Если после фильтрации не осталось сигналов — пропускаем область
1904      # (но можно оставить пустую, если хотите)
1905
1906      # Десериализуем маркеры
1907      shapes = []
1908      for shape_data in area_data.get('shapes', []):
1909          shape = deserialize_shape(shape_data)
1910          if shape is not None:
1911              shapes.append(shape)
1912
1913      return {
1914          'id': area_id,
1915          'signals': valid_signals,
1916          'shapes': shapes,
1917          'cursor_time': None,  # Пересчитывается при загрузке
1918          'x_range': None,      # Пересчитывается при загрузке
1919          'y_range': None       # Пересчитывается при загрузке
1920      }
1921
1922
1923  def create_visualizer_state(
1924      selected_signals: Set[str],
1925      plot_areas: List[Dict[str, Any]]
1926  ) -> Dict[str, Any]:
1927      """
1928      Создаёт объект состояния визуализатора для сохранения.
1929
1930      Args:
1931          selected_signals: Набор выбранных сигналов
1932          plot_areas: Список областей графиков
1933
1934      Returns:
1935          Словарь, готовый для JSON сериализации
1936      """
1937      return {
1938          'version': STATE_VERSION,
1939          'selected_signals': sorted(list(selected_signals)),
1940          'plot_areas': [serialize_plot_area(pa) for pa in plot_areas]
1941      }
1942
1943
1944  def load_visualizer_state(
1945      state_data: Optional[Dict[str, Any]],
1946      available_signals: Set[str]
1947  ) -> tuple[Set[str], List[Dict[str, Any]], List[str]]:
1948      """
1949      Загружает и валидирует состояние визуализатора.
1950      Мягкая загрузка: если сигнала из состояния нет в проекте, он просто игнорируется
     без предупреждений.
1951      """
```

```
    warnings = []

    # Если состояния нет — возвращаем пустые значения
    if state_data is None:
        return set(), [], []

    # Проверяем версию
    version = state_data.get('version', 1)
    if version > STATE_VERSION:
        warnings.append(f"Версия состояния ({version}) новее текущей")

    # --- Мягкая загрузка выбранных сигналов ---
    raw_selected = state_data.get('selected_signals', [])
    selected_signals = set()

    for sig in raw_selected:
        # Добавляем сигнал только если он реально существует в проекте сейчас
        if sig in available_signals:
            selected_signals.add(sig)
        # Если сигнала нет — просто молчим (никаких missing_signals и warnings)

    # --- Мягкая загрузка областей графиков ---
    plot_areas = []
    for area_data in state_data.get('plot_areas', []):
        area = deserialize_plot_area(area_data, available_signals)
        # Если область валидна (в ней есть сигналы или маркеры), добавляем её
        if area is not None:
            # Если в области были сигналы, которые удалили из проекта,
            # deserialize_plot_area их уже отфильтровал внутри.
            # Если в области вообще не осталось сигналов — мы всё равно её создадим,
            # но она будет пустой (пользователь сам решит, что с ней делать).
            plot_areas.append(area)

    return selected_signals, plot_areas, warnings


def state_to_json(state: Dict[str, Any]) -> str:
    """Конвертирует состояние в JSON строку"""
    return json.dumps(state, ensure_ascii=False, indent=2)


def state_from_json(json_str: str) -> Optional[Dict[str, Any]]:
    """Парсит JSON строку в состояние"""
    try:
        return json.loads(json_str)
    except (json.JSONDecodeError, TypeError):
        return None

#code_signal.py

import re
from typing import List, Tuple, Dict

import numpy as np
import pandas as pd


class CodeEvaluationError(Exception):
    """Ошибка во время вычисления выражения CODE."""


def sanitize_numeric_column(series: pd.Series) -> pd.Series:
    if series.dtype.kind in ("i", "u", "f"):
        return series
    text = series.astype(str).str.replace(",", ".", regex=False)
```

```
    return pd.to_numeric(text, errors="coerce")


def evaluate_code_expression(code_str: str, df_all: pd.DataFrame) -> Tuple[pd.Series, List[str]]:
    if df_all is None or df_all.empty:
        raise CodeEvaluationError("Нет данных для расчёта синтетического сигнала.")
    if not code_str or not code_str.strip():
        raise CodeEvaluationError("Строка CODE пуста.")

    index = df_all.index
    numeric_df = df_all.apply(sanitize_numeric_column)
    series_map = {col: numeric_df[col] for col in numeric_df.columns}
    warnings: List[str] = []

    # ---------- обработка «неправильных» имён сигналов ----------
    safe_name_map: Dict[str, str] = {}
    used_safe_names = set()

    def _make_safe_name(original: str, idx: int) -> str:
        base = re.sub(r"\W", "_", original)
        if not base or not re.match(r"[A-Za-z_]", base):
            base = f"SIG_{idx}"
        while base in used_safe_names:
            base += "_"
        used_safe_names.add(base)
        return base

    sorted_signals = sorted(series_map.keys(), key=len, reverse=True)
    for idx, sig_name in enumerate(sorted_signals):
        safe = _make_safe_name(sig_name, idx)
        safe_name_map[sig_name] = safe

    def _replace_signal_names(expr: str) -> str:
        result = []
        i = 0
        in_string = False
        string_char = ""

        while i < len(expr):
            ch = expr[i]
            if in_string:
                result.append(ch)
                if ch == string_char and expr[i - 1] != "\\":
                    in_string = False
                i += 1
                continue

            if ch in ("'", '"'):
                in_string = True
                string_char = ch
                result.append(ch)
                i += 1
                continue

            matched = None
            for name in sorted_signals:
                if expr.startswith(name, i):
                    matched = name
                    break
            if matched:
                result.append(safe_name_map[matched])
                i += len(matched)
            else:
                result.append(ch)
```

```
2081                 i += 1
2082
2083         return "".join(result)
2084
2085     # ---------- вспомогательные функции ----------
2086     def _ensure_series(value) -> pd.Series:
2087         if isinstance(value, pd.Series):
2088             return value.reindex(index)
2089         if isinstance(value, pd.DataFrame):
2090             if value.shape[1] == 1:
2091                 return value.iloc[:, 0].reindex(index)
2092             raise CodeEvaluationError("Невозможно привести DataFrame с несколькими
      колонками к Series.")
2093         if isinstance(value, (list, tuple, np.ndarray)):
2094             arr = np.asarray(value, dtype=float)
2095             if arr.size == 1:
2096                 arr = np.full(len(index), arr.item())
2097             elif arr.shape[0] != len(index):
2098                 return pd.Series(np.nan, index=index)
2099             return pd.Series(arr, index=index)
2100         if value is None or np.isscalar(value):
2101             return pd.Series(value, index=index)
2102         try:
2103             return pd.Series(value, index=index)
2104         except Exception as exc:
2105             raise CodeEvaluationError(f"Невозможно преобразовать значение '{value}' к
      Series.") from exc
2106
2107     def _aggregate_nanfunc(func, args, empty_value=np.nan):
2108         if not args:
2109             return pd.Series(empty_value, index=index)
2110         stacked = np.vstack([_ensure_series(arg).values for arg in args])
2111         return pd.Series(func(stacked, axis=0), index=index)
2112
2113     def GETPOINT(*_):
2114         if "GETPOINT" not in warnings:
2115             warnings.append("GETPOINT пока не поддержан — возвращается NaN.")
2116         return pd.Series(np.nan, index=index)
2117
2118     def PREV(param):
2119         s = _history_series(param)
2120         if s is None:
2121             return pd.Series(np.nan, index=index)
2122         return s.shift(1)
2123
2124     def _history_series(param):
2125         # 1) Если уже Series — используем её
2126         if isinstance(param, pd.Series):
2127             return sanitize_numeric_column(param).reindex(index)
2128
2129         # 2) Если пришло "безопасное имя" (SIG_...) — оно уже есть в env как Series.
2130         # Но сюда оно попадёт только если пользователь передал строку "SIG_0".
2131         if isinstance(param, str):
2132             # сначала пробуем как исходное имя сигнала
2133             if param in series_map:
2134                 return series_map[param]
2135
2136             # потом пробуем как safe-name
2137             for orig, safe in safe_name_map.items():
2138                 if param == safe:
2139                     return series_map.get(orig)
2140
2141         return None
2142
2143     def _history_window(period):
```

```
2144          try:
2145              minutes = int(period)
2146          except (TypeError, ValueError):
2147              return None
2148          if minutes <= 0:
2149              return None
2150          return f"{minutes}min"
2151
2152      def _history_apply(param, period, fn):
2153          s = _history_series(param)
2154          window = _history_window(period)
2155          if s is None or window is None:
2156              return pd.Series(np.nan, index=index)
2157
2158          # 1) Если datetime-индекс — используем time-based rolling
2159          if isinstance(s.index, (pd.DatetimeIndex, pd.TimedeltaIndex, pd.PeriodIndex)):
2160              return fn(s.rolling(window, min_periods=1))
2161
2162          # 2) Иначе пробуем интерпретировать period как "кол-во точек"
2163          try:
2164              n = int(period)
2165              if n <= 0:
2166                  return pd.Series(np.nan, index=index)
2167              return fn(s.rolling(window=n, min_periods=1))
2168          except Exception:
2169              return pd.Series(np.nan, index=index)
2170
2171      HISTORYAVG = lambda n, p: _history_apply(n, p, lambda r: r.mean())
2172      HISTORYCOUNT = lambda n, p: _history_apply(n, p, lambda r: r.count())
2173      HISTORYSUM = lambda n, p: _history_apply(n, p, lambda r: r.sum())
2174      HISTORYMAX = lambda n, p: _history_apply(n, p, lambda r: r.max())
2175      HISTORYMIN = lambda n, p: _history_apply(n, p, lambda r: r.min())
2176      HISTORYDIFF = lambda n, p: _history_apply(n, p, lambda r: r.max() - r.min())
2177
2178      def HISTORYGRADIENT(param_name, period):
2179          s = _history_series(param_name)
2180          window = _history_window(period)
2181          if s is None or window is None:
2182              return pd.Series(np.nan, index=index)
2183
2184          def slope(window_series: pd.Series):
2185              valid = window_series.dropna()
2186              if len(valid) < 2:
2187                  return np.nan
2188              x = valid.index.view(np.int64).astype(float) / 1e9
2189              y = valid.values.astype(float)
2190              x_mean = x.mean()
2191              y_mean = y.mean()
2192              denom = np.sum((x - x_mean) ** 2)
2193              if denom == 0:
2194                  return np.nan
2195              return np.sum((x - x_mean) * (y - y_mean)) / denom
2196
2197          return s.rolling(window).apply(slope, raw=False)
2198
2199      def ROUND(a, b=0):
2200          a_values = _ensure_series(a).values
2201          b_values = _ensure_series(b).values
2202          decimals = [
2203              0 if np.isnan(dec) else int(round(dec))
2204              for dec in b_values
2205          ]
2206          rounded = np.array([
2207              np.round(val, dec) if not np.isnan(val) else np.nan
2208              for val, dec in zip(a_values, decimals)
```

```
2209          ])
2210          return pd.Series(rounded, index=index)
2211
2212      # ---------- окружение eval ----------
2213      env = {
2214          "np": np,
2215          "ABS": lambda a: pd.Series(np.abs(_ensure_series(a).values), index=index),
2216          "EXP": lambda a: pd.Series(np.exp(_ensure_series(a).values), index=index),
2217          "POW": lambda a, b: pd.Series(np.power(_ensure_series(a).values,
      _ensure_series(b).values), index=index),
2218          "MIN": lambda *args: _aggregate_nanfunc(np.nanmin, args),
2219          "MAX": lambda *args: _aggregate_nanfunc(np.nanmax, args),
2220          "AVG": lambda *args: _aggregate_nanfunc(np.nanmean, args, empty_value=0.0),
2221          "MED": lambda *args: _aggregate_nanfunc(np.nanmedian, args),
2222          "ROUND": ROUND,
2223          "WHEN": lambda cond, t_val, f_val: pd.Series(
2224              np.where(_ensure_series(cond).astype(bool).values,
2225                       _ensure_series(t_val).values,
2226                       _ensure_series(f_val).values),
2227              index=index,
2228          ),
2229          "LOG": lambda x: pd.Series(np.log(_ensure_series(x).values), index=index),
2230          # Логарифм по основанию 10 (если нужен)
2231          "LOG10": lambda x: pd.Series(np.log10(_ensure_series(x).values), index=index),
2232          "PREV": PREV,
2233          "HISTORYAVG": HISTORYAVG,
2234          "HISTORYCOUNT": HISTORYCOUNT,
2235          "HISTORYSUM": HISTORYSUM,
2236          "HISTORYMAX": HISTORYMAX,
2237          "HISTORYMIN": HISTORYMIN,
2238          "HISTORYDIFF": HISTORYDIFF,
2239          "HISTORYGRADIENT": HISTORYGRADIENT,
2240          "GETPOINT": GETPOINT,
2241      }
2242
2243      for original_name, safe_name in safe_name_map.items():
2244          env[safe_name] = series_map[original_name]
2245
2246      def _normalize_expression(expr: str) -> str:
2247          expr = re.sub(r"\bAND\b", "&", expr, flags=re.IGNORECASE)
2248          expr = re.sub(r"\bOR\b", "|", expr, flags=re.IGNORECASE)
2249          expr = re.sub(r"\bNOT\b", "~", expr, flags=re.IGNORECASE)
2250          expr = expr.replace("<>", "!=")
2251          expr = re.sub(r"(?<![<>=!])=(?![<>=])", "==", expr)
2252          return expr
2253
2254      normalized_code = _normalize_expression(code_str)
2255      normalized_code = _replace_signal_names(normalized_code)
2256
2257      try:
2258          raw_result = eval(normalized_code, {"__builtins__": {}}, env)
2259      except Exception as exc:
2260          raise CodeEvaluationError(str(exc)) from exc
2261
2262      result_series = _ensure_series(raw_result)
2263      result_series.name = result_series.name or "CODE_RESULT"
2264      return result_series, warnings
2265
2266  def compute_code_signal(
2267      code_str: str,
2268      df_all: pd.DataFrame,
2269      warn_callback=lambda msg: None,
2270  ) -> pd.Series:
2271      """
2272      Совместимость с визуализатором: считает синтетический сигнал по CODE
```

```
    и прокидывает предупреждения через колбэк.
    """
    series, warnings = evaluate_code_expression(code_str, df_all)
    for message in warnings:
        warn_callback(message)
    return series
```

```
  /**
 * Главный модуль приложения
 * app.js
 */

const App = {
    /**
     * Инициализация приложения
     */
    init() {
        Settings.init().catch(console.error);
        //Settings.init().then(() => {
        //    // если хочешь — можно обновить UI (например, статус “Сигналы загружены”)
        //    console.log('Settings loaded, signals:', Settings.signals.length);
        //    }).catch(err => console.error(err));
        //console.log('signals loaded:', Settings.signals.slice(0, 5));
        this.setupPaletteDragDrop();
        this.setupGlobalMouseHandlers();
        this.setupContextMenu();
        this.setupWorkspaceClick();
        this.setupOutputCounter();
        this.setupMultiSelection();

        // Инициализация модулей
        Viewport.init();
        Modal.init();
        Project.init();

        // Первоначальное определение выходов (только если модуль загружен)
        if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
            Outputs.updateOutputStatus();
        }

        console.log('Logic Scheme Editor initialized');
        document.getElementById('btn-generate-code').addEventListener('click', () => {
            const code = CodeGen.generate();
            document.getElementById('code-output').value = code;
            document.getElementById('code-modal-overlay').style.display = 'flex';
        });

        document.getElementById('code-modal-close').addEventListener('click', () => {
            document.getElementById('code-modal-overlay').style.display = 'none';
        });
        document.getElementById('btn-visualize').addEventListener('click', () => {
            App.openSignalVisualizer();
});
    },

openSignalVisualizer() {
    try {
        // 1) Собираем входные сигналы
        const signals = Object.values(AppState.elements)
            .filter(e => e && e.type === 'input-signal')
            .map(e => e.props?.name || e.id);
        const uniqSignals = [...new Set(signals)];

        if (uniqSignals.length === 0) {
```

```
2337             alert('Нет входных сигналов в схеме.');
2338             return;
2339         }
2340 
2341         // 2) Генерируем код
2342         let codeStr = '';
2343         if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
     {
2344             codeStr = CodeGen.generate() || '';
2345         }
2346 
2347         // 3) Определяем URL-ы динамически
2348         const currentHost = window.location.hostname;
2349         const apiPort = window.location.port || 8000;
2350         const visualizerPort = Settings.config?.visualizerPort || 8501;
2351 
2352         const apiUrl = `http://${currentHost}:${apiPort}`;
2353         const visualizerBase = `http://${currentHost}:${visualizerPort}`;
2354 
2355         console.log('API URL:', apiUrl);
2356         console.log('Visualizer URL:', visualizerBase);
2357 
2358         // 4) Получаем сохранённое состояние визуализатора из проекта
2359         const visualizerState = AppState.project?.visualizer_state || null;
2360 
2361         if (visualizerState) {
2362             console.log('Передаём сохранённое состояние визуализатора:',
     visualizerState);
2363         }
2364 
2365         // 5) Создаём сессию на backend (с передачей состояния)
2366         fetch('/api/visualize/session', {
2367             method: 'POST',
2368             headers: { 'Content-Type': 'application/json' },
2369             body: JSON.stringify({
2370                 signals: uniqSignals,
2371                 code: codeStr,
2372                 visualizer_state: visualizerState  // НОВОЕ: передаём состояние
2373             })
2374         })
2375         .then(r => {
2376             if (!r.ok) throw new Error('Failed to create visualize session');
2377             return r.json();
2378         })
2379         .then(data => {
2380             const token = data.token;
2381 
2382             // НОВОЕ: сохраняем токен для последующего получения состояния
2383             AppState.currentVisualizerToken = token;
2384 
2385             const params = new URLSearchParams();
2386             params.set('session', token);
2387             params.set('api_url', apiUrl);
2388 
2389             const visualizerUrl = `${visualizerBase}/?${params.toString()}`;
2390             console.log('Opening visualizer:', visualizerUrl);
2391             window.open(visualizerUrl, '_blank');
2392         })
2393         .catch(err => {
2394             console.error(err);
2395             alert('Не удалось открыть визуализатор: ' + err.message);
2396         });
2397 
2398     } catch (e) {
2399         console.error(e);
```

```
2400          alert('Ошибка при подготовке визуализации: ' + e.message);
2401      }
2402  },
2403
2404  /**
2405   * Получает состояние визуализатора с сервера
2406   * Вызывается перед сохранением проекта
2407   */
2408  async fetchVisualizerState() {
2409      if (!AppState.currentVisualizerToken) {
2410          console.log('Нет активной сессии визуализатора');
2411          return null;
2412      }
2413
2414      try {
2415          const response = await fetch(`/api/visualize/get-state/$
      {AppState.currentVisualizerToken}`);
2416
2417          if (!response.ok) {
2418              console.warn('Не удалось получить состояние визуализатора:',
      response.status);
2419              return null;
2420          }
2421
2422          const result = await response.json();
2423
2424          if (result.success && result.state) {
2425              console.log('Получено состояние визуализатора:', result.state);
2426              return result.state;
2427          }
2428
2429          return null;
2430      } catch (error) {
2431          console.error('Ошибка получения состояния визуализатора:', error);
2432          return null;
2433      }
2434  },
2435
2436      /**
2437       * Отмена состояния drag из палитры (helper)
2438       */
2439      cancelPaletteDrag() {
2440          if (AppState.dragPreview) {
2441              try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
2442              AppState.dragPreview = null;
2443          }
2444          AppState.isDraggingFromPalette = false;
2445          AppState.dragType = null;
2446      },
2447
2448      /**
2449       * Настройка счётчика выходов в меню
2450       */
2451      setupOutputCounter() {
2452          // Не создавать повторно, если уже есть
2453          if (document.getElementById('btn-outputs')) return;
2454
2455          const menu = document.getElementById('menu');
2456
2457          // Создаём кнопку с счётчиком выходов
2458          const outputBtn = document.createElement('button');
2459          outputBtn.className = 'menu-btn output-btn';
2460          outputBtn.id = 'btn-outputs';
2461          outputBtn.innerHTML = `
2462              📤 Выходы
```

```
2463              <span id="output-counter" class="output-counter">0</span>
2464          `;
2465
2466          // Вставляем после кнопки свойств проекта
2467          const projectBtn = document.getElementById('btn-project-settings');
2468          if (projectBtn) {
2469              projectBtn.after(outputBtn);
2470          } else {
2471              menu.appendChild(outputBtn);
2472          }
2473
2474          outputBtn.addEventListener('click', () => {
2475              Modal.showProjectPropertiesModal();
2476          });
2477      },
2478
2479      /**
2480       * Настройка drag & drop из палитры
2481       */
2482      setupPaletteDragDrop() {
2483          document.querySelectorAll('.palette-item').forEach(item => {
2484              item.addEventListener('mousedown', (e) => {
2485                  // Только левая кнопка мыши должна запускать drag из палитры
2486                  if (e.button !== 0) return;
2487                  e.preventDefault();
2488
2489                  AppState.isDraggingFromPalette = true;
2490                  AppState.dragType = item.dataset.type;
2491
2492                  AppState.dragPreview = document.createElement('div');
2493                  AppState.dragPreview.className = 'drag-preview';
2494                  AppState.dragPreview.textContent =
      ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
2495                  AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2496                  AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2497                  document.body.appendChild(AppState.dragPreview);
2498              });
2499          });
2500      },
2501
2502      /**
2503       * Глобальные обработчики мыши
2504       */
2505  /**
2506   * Глобальные обработчики мыши
2507   */
2508  setupGlobalMouseHandlers() {
2509      document.addEventListener('mousemove', (e) => {
2510          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
2511              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2512              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2513          }
2514          if (AppState.resizing) {
2515              Elements.handleResize(e);
2516              return;
2517          }
2518          if (AppState.draggingElement) {
2519              Elements.handleDrag(e);
2520          }
2521          if (AppState.tempLine && AppState.connectingFrom) {
2522              Connections.drawTempConnection(e);
2523          }
2524      });
2525
2526      document.addEventListener('mouseup', (e) => {
```

```
2527          if (AppState.resizing) {
2528              AppState.resizing = null;
2529              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
2530          }
2531
2532          if (AppState.isDraggingFromPalette) {
2533              try {
2534                  if (AppState.dragPreview) {
2535                      AppState.dragPreview.remove();
2536                      AppState.dragPreview = null;
2537                  }
2538
2539                  const container = document.getElementById('workspace-container');
2540                  const rect = container.getBoundingClientRect();
2541
2542                  if (e.clientX >= rect.left && e.clientX <= rect.right &&
2543                      e.clientY >= rect.top && e.clientY <= rect.bottom) {
2544
2545                      const canvasPos = screenToCanvas(e.clientX, e.clientY);
2546                      const config = ELEMENT_TYPES[AppState.dragType];
2547                      if (config) {
2548                          const defaultWidth = config.minWidth || 120;
2549                          const defaultHeight = config.minHeight || 60;
2550
2551                          // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
      обработать
2552                          const newElement = Elements.addElement(
2553                              AppState.dragType,
2554                              canvasPos.x - defaultWidth / 2,
2555                              canvasPos.y - defaultHeight / 2
2556                          );
2557
2558                          if (newElement && typeof Outputs !== 'undefined') {
2559                              Outputs.updateOutputStatus();
2560                          }
2561                      } else {
2562                          console.error('Неизвестный тип элемента при drop:',
      AppState.dragType);
2563                      }
2564                  }
2565              } finally {
2566                  App.cancelPaletteDrag();
2567              }
2568          }
2569
2570          if (AppState.draggingElement) {
2571              AppState.draggingElement = null;
2572          }
2573
2574          Connections.clearConnectionState();
2575      });
2576
2577      document.addEventListener('keydown', (e) => {
2578      // 1. Проверяем, не печатает ли пользователь текст
2579          const target = e.target;
2580          const isInput = target.tagName === 'INPUT' ||
2581                          target.tagName === 'TEXTAREA' ||
2582                          target.isContentEditable;
2583
2584          if (isInput) return; // Если печатаем - игнорируем глобальные хоткеи
2585
2586          // 2. Проверяем, не открыто ли модальное окно
2587          const modal = document.getElementById('modal-overlay');
2588          const projectModal = document.getElementById('project-modal-overlay');
2589          const isModalOpen = (modal && modal.style.display !== 'none') ||
```

```
                            (projectModal && projectModal.style.display !== 'none');

        if (isModalOpen) return; // Если открыто окно - игнорируем

        // --- Дальше старая логика ---

        if (e.key === 'Delete' && AppState.selectedElement) {
            Elements.deleteElement(AppState.selectedElement);
            if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
        }

        if (e.key === 'Escape') {
            Elements.deselectAll();
            Connections.clearConnectionState();
            if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
        }
    });
},

    /**
     * Настройка контекстного меню
     */
    setupContextMenu() {
        document.addEventListener('click', (e) => {
            const menu = document.getElementById('context-menu');
            if (!menu.contains(e.target)) {
                menu.style.display = 'none';
            }
        });

        document.getElementById('ctx-properties').addEventListener('click', () => {
            const elemId = document.getElementById('context-menu').dataset.elementId;
            document.getElementById('context-menu').style.display = 'none';
            const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
            if (config?.hasProperties) {
                Modal.showPropertiesModal(elemId);
            }
        });

        document.getElementById('ctx-delete').addEventListener('click', () => {
            document.getElementById('context-menu').style.display = 'none';

            // Используем новую функцию для удаления всех выделенных
            Elements.deleteSelectedElements();

            if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
                Outputs.updateOutputStatus();
            }
        });
        document.getElementById('ctx-copy').addEventListener('click', () => {
            document.getElementById('context-menu').style.display = 'none';
            Elements.copySelectedElements();
        });
    },

    /**
     * Клик по рабочей области
     */
    // app.js
    // app.js
    setupWorkspaceClick() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('click', (e) => {
            // Если мы только что закончили тянуть РАМКУ (реальное выделение), не
```

```
    сбрасываем
            if (AppState.marqueeJustEnded) return;

            // Если кликнули ЛЕВОЙ кнопкой мыши НЕ по элементу и НЕ по порту
            if (e.button === 0 && !e.target.closest('.element') && !
    e.target.closest('.port')) {
                Elements.deselectAll();
            }
        });
    },
    /**
     * --- Выделение рамкой и множественное перемещение ---
     */
    // app.js
    setupMultiSelection() {
        const container = document.getElementById('workspace-container');
        const rectEl = document.getElementById('selection-rect');

        container.addEventListener('mousedown', (e) => {
            // РАМКА: только ЛЕВАЯ кнопка (0) и клик НЕ по элементу
            if (e.button !== 0 || e.target.closest('.element') ||
    e.target.closest('#minimap')) return;

            const pos = screenToCanvas(e.clientX, e.clientY);
            AppState.multiSelecting = true;
            AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
    pos.y, w: 0, h: 0 };

            rectEl.style.left = e.clientX + 'px';
            rectEl.style.top = e.clientY + 'px';
            rectEl.style.width = '0px';
            rectEl.style.height = '0px';
            rectEl.style.display = 'block';
        });

        document.addEventListener('mousemove', (e) => {
            if (!AppState.multiSelecting) return;

            const pos = screenToCanvas(e.clientX, e.clientY);
            const sx = AppState.selectionRect.startX;
            const sy = AppState.selectionRect.startY;

            const x = Math.min(sx, pos.x);
            const y = Math.min(sy, pos.y);
            const w = Math.abs(pos.x - sx);
            const h = Math.abs(pos.y - sy);

            // Обновляем визуальную рамку
            rectEl.style.left = (x * AppState.viewport.zoom + AppState.viewport.panX)
    + 'px';
            rectEl.style.top = (y * AppState.viewport.zoom + AppState.viewport.panY) +
    'px';
            rectEl.style.width = (w * AppState.viewport.zoom) + 'px';
            rectEl.style.height = (h * AppState.viewport.zoom) + 'px';

            // Ищем элементы внутри
            const selected = [];
            for (const [id, elData] of Object.entries(AppState.elements)) {
                if (!elData || elData.type === 'output-frame') continue;
                if (elData.x >= x && elData.x + elData.width <= x + w &&
                    elData.y >= y && elData.y + elData.height <= y + h) {
                    selected.push(id);
                }
            }

```

```
            AppState.selectedElements = selected;
            AppState.selectedElement = selected.length > 0 ? selected[selected.length - 1] : null;

            document.querySelectorAll('.element').forEach(el => {
                el.classList.toggle('selected', selected.includes(el.id));
            });
        });

        document.addEventListener('mouseup', () => {
            if (AppState.multiSelecting) {
                AppState.multiSelecting = false;
                const rectEl = document.getElementById('selection-rect');
                const w = parseInt(rectEl.style.width) || 0;
                const h = parseInt(rectEl.style.height) || 0;
                rectEl.style.display = 'none';

                // Флаг, чтобы setupWorkspaceClick не сбросил выделение сразу
                if (w > 2 || h > 2) {
                    AppState.marqueeJustEnded = true;
                    setTimeout(() => { AppState.marqueeJustEnded = false; }, 50);
                }
            }
        });
    },
};

// Запуск приложения при загрузке страницы
document.addEventListener('DOMContentLoaded', () => {
    App.init();
});

// js/codegen_graph.js


const CodeGenGraph = {
        /**
     * Собрать все условия вверх по цепочке cond-портов (до корня).
     * Возвращает null или объединённое через AND условие.
     */
/**
 * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
 */
   collectAllCond(graph) {
        if (!graph) return null;

        let c = null;
        const elem = graph.elem;

        // 1. Собираем условия через cond-порт (как было)
        if (graph.condInput) {
            const condConn = graph.condInput.conn;
            const fromGraph = graph.condInput.fromGraph;
            const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
            c = oneCond;

            // Рекурсивно идём вверх по cond-цепочке
            const upCond = this.collectAllCond(fromGraph);
            if (upCond) {
                c = c ? Optimizer.And(c, upCond) : upCond;
            }
        }

        // 2. НОВОЕ: если это separator — учитываем контекст его входа
        if (elem.type === 'separator' && graph.inputs.length > 0) {
```

```
            const inputGraph = graph.inputs[0].fromGraph;
            const inputContext = this.collectAllCond(inputGraph);
            if (inputContext) {
                c = c ? Optimizer.And(c, inputContext) : inputContext;
            }
        }

        return c;
    },
    buildDependencyGraph(elementId) {
        const graph = {
            nodeId: elementId,
            elem: AppState.elements[elementId],
            inputs: [],
            condInput: null,
        };

        if (!graph.elem) return null;

        const inConns = AppState.connections
        .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
        .sort((a, b) => {
            const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
            const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
            return ai - bi;
        });

        inConns.forEach(conn => {
        graph.inputs.push({
            conn,
            fromGraph: this.buildDependencyGraph(conn.fromElement)
        });
        });

        const condConn = AppState.connections.find(c =>
            c.toElement === elementId && c.toPort === 'cond-0'
        );
        if (condConn) {
            graph.condInput = {
                conn: condConn,
                fromGraph: this.buildDependencyGraph(condConn.fromElement)
            };
        }

        return graph;
    },

    /**
     * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
     */
    evalLogic(graph) {
        if (!graph) return Optimizer.TrueCond;
        const elem = graph.elem;

        switch (elem.type) {
            case 'if': {
                const left = graph.inputs[0]?.fromGraph;
                const right = graph.inputs[1]?.fromGraph;

                const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
                const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);

                const op = elem.props.operator || '=';
                return this.buildIfLogic(leftVal, op, rightVal);
            }
```

```
2843
2844                case 'and': {
2845                    let result = null;
2846                    for (const inp of graph.inputs) {
2847                        const inLogic = this.evalLogic(inp.fromGraph);
2848                        result = result ? Optimizer.And(result, inLogic) : inLogic;
2849                    }
2850                    return result || Optimizer.TrueCond;
2851                }
2852
2853                case 'or': {
2854                    let result = null;
2855                    for (const inp of graph.inputs) {
2856                        const inLogic = this.evalLogic(inp.fromGraph);
2857                        result = result ? Optimizer.Or(result, inLogic) : inLogic;
2858                    }
2859                    return result || Optimizer.FalseCond;
2860                }
2861
2862                case 'not': {
2863                    const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
2864                    return Optimizer.Not(inLogic);
2865                }
2866
2867                case 'separator': {
2868                    return this.evalLogic(graph.inputs[0]?.fromGraph);
2869                }
2870
2871                default:
2872                    return Optimizer.TrueCond;
2873            }
2874        },
2875
2876        /**
2877         * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
2878         */
2879        evalValue(graph) {
2880            if (!graph) return Optimizer.Const(0);
2881            const elem = graph.elem;
2882
2883            switch (elem.type) {
2884                case 'input-signal':
2885                    return Optimizer.Var(elem.props.name || graph.nodeId);
2886
2887                case 'const':
2888                    return Optimizer.Const(Number(elem.props.value) || 0);
2889
2890                case 'formula': {
2891                    const expr = this.buildFormulaExpr(elem);
2892                    return Optimizer.Var(expr);
2893                }
2894
2895                case 'separator':
2896                    return this.evalValue(graph.inputs[0]?.fromGraph);
2897
2898                default:
2899                    return Optimizer.Const(0);
2900            }
2901        },
2902
2903        // js/codegen_graph.js
2904
2905        /**
2906         * Рекурсивно собрать полный контекст условий для элемента
2907         * через всю цепочку cond-портов вверх
```

```
2908        */
2909    // В codegen_graph.js, в evalFullContext добавь:
2910
2911        evalFullContext(graph) {
2912            if (!graph) return null;
2913
2914            let context = null;
2915            const elem = graph.elem;
2916
2917            console.log(`evalFullContext для ${elem.id} (${elem.type})`);
2918
2919            // 1. Если сам элемент имеет cond-порт — собираем его условие
2920            if (graph.condInput) {
2921                const condConn = graph.condInput.conn;
2922                console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
      {condConn.fromPort}`);
2923
2924                const condLogic = this.evalConditionFromPort(
2925                    graph.condInput.fromGraph,
2926                    condConn.fromPort
2927                );
2928                console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
2929                context = condLogic;
2930
2931                // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
2932                const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
2933                if (upstreamContext) {
2934                    console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
2935                    context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
2936                }
2937            } else {
2938                console.log(`  → нет cond-0`);
2939            }
2940
2941            console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
2942            return context;
2943        },
2944
2945        /**
2946         * Получить УСЛОВИЕ для cond-порта элемента
2947         * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
2948         */
2949        evalConditionFromPort(graph, fromPort) {
2950            if (!graph) return null;
2951            const elem = graph.elem;
2952
2953            // Если это сепаратор — вычисляем его вход и применяем ветвление
2954            if (elem.type === 'separator') {
2955                const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
2956
2957                if (fromPort === 'out-0') {
2958                    return inputLogic;
2959                } else if (fromPort === 'out-1') {
2960                    return Optimizer.Not(inputLogic);
2961                }
2962            }
2963
2964            // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
2965            if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
2966                return this.evalLogic(graph);
2967            }
```

```
2968
2969          return null;
2970      },
2971
2972      /**
2973       * Главная функция: получить {cond, expr} для элемента
2974       */
2975      evalGraphValue(graph) {
2976
2977          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
2978
2979          const elem = graph.elem;
2980          //let cond = null;
2981
2982          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
2983          let cond = this.collectAllCond(graph);
2984
2985          let expr = null;
2986
2987          switch (elem.type) {
2988              case 'input-signal':
2989                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
2990                  break;
2991
2992              case 'const':
2993                  expr = Optimizer.Const(Number(elem.props.value) || 0);
2994                  break;
2995
2996              case 'formula': {
2997                  // Для формулы также собираем условия от всех входных элементов
2998                  const inputConds = graph.inputs.map(inp => {
2999                      const inResult = this.evalGraphValue(inp.fromGraph);
3000                      return inResult.cond;
3001                  }).filter(c => c);
3002
3003                  // Объединяем cond-порт с условиями от входов
3004                  for (const inCond of inputConds) {
3005                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
3006                  }
3007
3008                  expr = Optimizer.Var(this.buildFormulaExpr(elem));
3009                  break;
3010              }
3011
3012              case 'separator':
3013                  // Сепаратор — просто пробрасываем значение дальше
3014                  return this.evalGraphValue(graph.inputs[0]?.fromGraph);
3015
3016              // Логические элементы не должны здесь быть
3017              case 'and':
3018              case 'or':
3019              case 'not':
3020              case 'if':
3021              default:
3022                  expr = Optimizer.Const(0);
3023          }
3024
3025          return { cond, expr };
3026      },
3027
3028      buildIfLogic(leftVal, op, rightVal) {
3029          const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
3030          const rightName = rightVal.type === 'var' ? rightVal.name :
      String(rightVal.n);
3031
```

```
3032          const leftZero = leftVal.type === 'const' && leftVal.n === 0;
3033          const rightZero = rightVal.type === 'const' && rightVal.n === 0;
3034
3035          switch (op) {
3036              case '=':
3037                  if (rightZero) return Optimizer.Eq0(leftName);
3038                  if (leftZero) return Optimizer.Eq0(rightName);
3039                  return Optimizer.Cmp(leftName, '=', rightName);
3040              case '!=':
3041                  if (rightZero) return Optimizer.Ne0(leftName);
3042                  if (leftZero) return Optimizer.Ne0(rightName);
3043                  return Optimizer.Cmp(leftName, '!=', rightName);
3044              case '>':
3045              case '<':
3046              case '>=':
3047              case '<=':
3048                  return Optimizer.Cmp(leftName, op, rightName);
3049              default:
3050                  return Optimizer.TrueCond;
3051          }
3052      },
3053
3054
3055      buildFormulaExpr(elem) {
3056          let result = elem.props.expression || '0';
3057
3058          // 1) Сначала раскрываем шаблоны (h и др.)
3059          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
3060              ? Settings.getTemplatesMap()
3061              : null;
3062          result = expandFormulaTemplates(result, map);
3063
3064          // 2) Потом раскрываем ссылки на формулы
3065          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
3066          for (const ref of formulaRefs) {
3067              const refElem = AppState.elements[ref];
3068              if (refElem && refElem.type === 'formula') {
3069                  const refExpr = this.buildFormulaExpr(refElem);
3070                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
3071              }
3072          }
3073
3074          return result;
3075      }
3076  };
3077
3078  window.CodeGenGraph = CodeGenGraph;
3079
3080
3081  // js/codegen_optimizer.js
3082
3083  let _depth = 0;
3084  const MAX_DEPTH = 200;
3085
3086  // === Конструкторы ===
3087  function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
3088  function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
3089  function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
3090  function And(a, b) {
3091      if (!a) return b;
3092      if (!b) return a;
3093      return { kind: 'cond', type: 'and', a, b };
3094  }
3095  function Or(a, b) {
3096      if (!a) return b;
```

```
    if (!b) return a;
    return { kind: 'cond', type: 'or', a, b };
}
function Not(x) {
    if (!x) return null;
    return { kind: 'cond', type: 'not', x };
}
const TrueCond = { kind: 'cond', type: 'true' };
const FalseCond = { kind: 'cond', type: 'false' };

function Const(n) { return { kind: 'expr', type: 'const', n }; }
function Var(name) { return { kind: 'expr', type: 'var', name }; }
function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }

// === Утилиты ===
function atomKey(c) {
    if (!c) return null;
    switch (c.type) {
        case 'eq0': return `eq0:${c.v}`;
        case 'ne0': return `ne0:${c.v}`;
        case 'cmp': return `cmp:${c.l}:${c.op}:${c.r}`;
        case 'true': return 'true';
        case 'false': return 'false';
        default: return null;
    }
}

function splitAndCond(c) {
    if (!c || c.type !== 'and') return null;
    return [c.a, c.b];
}

function findSharedAndComplement(c1, c2) {
    const p1 = splitAndCond(c1);
    const p2 = splitAndCond(c2);
    if (!p1 || !p2) return null;

    const combos = [
        [p1[0], p1[1], p2[0], p2[1]],
        [p1[0], p1[1], p2[1], p2[0]],
        [p1[1], p1[0], p2[0], p2[1]],
        [p1[1], p1[0], p2[1], p2[0]],
    ];

    for (const [s1, x1, s2, x2] of combos) {
        if (condEq(s1, s2) && condNegationEq(x1, x2)) {
            return { shared: s1 };
        }
    }
    return null;
}

function negateOp(op) {
    switch (op) {
        case '=': return '!=';
        case '!=': return '=';
        case '>': return '<=';
        case '<': return '>=';
        case '>=': return '<';
        case '<=': return '>';
        default: return null;
    }
}

```

```
3162  // Преобразует cmp-условие в интервал по одной переменной
3163  // Возвращает { varName, min, minInc, max, maxInc } или null
3164  function cmpToInterval(c) {
3165      if (!c || c.type !== 'cmp') return null;
3166
3167      const lNum = parseNumberLiteral(c.l);
3168      const rNum = parseNumberLiteral(c.r);
3169
3170      let varName, op, val;
3171
3172      if (lNum == null && rNum != null) {
3173          // var OP const
3174          varName = c.l;
3175          op = c.op;
3176          val = rNum;
3177      } else if (lNum != null && rNum == null) {
3178          // const OP var  ->  var (OP') const
3179          varName = c.r;
3180          op = reverseOp(c.op);
3181          if (!op) return null;
3182          val = lNum;
3183      } else {
3184          // Либо обе стороны числа, либо обе не числа — не трогаем
3185          return null;
3186      }
3187
3188      // Интересуют только упорядочивающие операторы
3189      switch (op) {
3190          case '<':
3191          case '<=':
3192          case '>':
3193          case '>=':
3194          case '=':
3195              break;
3196          default:
3197              return null;
3198      }
3199
3200      let min = Number.NEGATIVE_INFINITY;
3201      let max = Number.POSITIVE_INFINITY;
3202      let minInc = false;
3203      let maxInc = false;
3204
3205      switch (op) {
3206          case '<':
3207              max = val; maxInc = false; break;
3208          case '<=':
3209              max = val; maxInc = true; break;
3210          case '>':
3211              min = val; minInc = false; break;
3212          case '>=':
3213              min = val; minInc = true; break;
3214          case '=':
3215              min = val; minInc = true;
3216              max = val; maxInc = true;
3217              break;
3218      }
3219
3220      return { varName, min, minInc, max, maxInc };
3221  }
3222
3223  function intervalSubset(a, b) {
3224      if (!a || !b) return false;
3225
3226      // Нижняя граница: a.min >= b.min
```

```
3227      const amin = a.min, bmin = b.min;
3228      if (amin === Number.NEGATIVE_INFINITY) {
3229          if (bmin !== Number.NEGATIVE_INFINITY) return false;
3230          // оба -∞ — ок
3231      } else if (bmin === Number.NEGATIVE_INFINITY) {
3232          // b начинается “раньше” — ок
3233      } else if (amin > bmin) {
3234          // a стартует правее b — ок
3235      } else if (amin < bmin) {
3236          // a захватывает меньшее значение — не подмножество
3237          return false;
3238      } else {
3239          // amin === bmin
3240          if (a.minInc && !b.minInc) {
3241              // a включает границу, а b — нет → в a есть точка, не входящая в b
3242              return false;
3243          }
3244      }
3245
3246      // Верхняя граница: a.max <= b.max
3247      const amax = a.max, bmax = b.max;
3248      if (amax === Number.POSITIVE_INFINITY) {
3249          if (bmax !== Number.POSITIVE_INFINITY) return false;
3250      } else if (bmax === Number.POSITIVE_INFINITY) {
3251          // b идёт дальше — ок
3252      } else if (amax < bmax) {
3253          // a заканчивается раньше — ок
3254      } else if (amax > bmax) {
3255          return false;
3256      } else {
3257          // amax === bmax
3258          if (a.maxInc && !b.maxInc) {
3259              return false;
3260          }
3261      }
3262
3263      return true;
3264  }
3265
3266  // Удаляет избыточные cmp-условия в массиве атомов
3267  // mode: 'and' | 'or'
3268  function removeRedundantCmpAtoms(atoms, mode) {
3269      if (!atoms || atoms.length < 2) return atoms;
3270
3271      const keep = new Array(atoms.length).fill(true);
3272
3273      for (let i = 0; i < atoms.length; i++) {
3274          if (!keep[i]) continue;
3275          const a = atoms[i];
3276          if (!a || a.type !== 'cmp') continue;
3277
3278          for (let j = 0; j < atoms.length; j++) {
3279              if (i === j || !keep[j]) continue;
3280              const b = atoms[j];
3281              if (!b || b.type !== 'cmp') continue;
3282
3283              const rel = cmpImplicationRelation(a, b);
3284              if (!rel) continue;
3285
3286              if (rel === 'a_in_b') {
3287                  if (mode === 'or') {
3288                      // A ⊆ B → A OR B = B  → A лишнее
3289                      keep[i] = false;
3290                      break;
3291                  } else if (mode === 'and') {
```

```
                    // A ⊆ B → A AND B = A → B лишнее
                    keep[j] = false;
                }
            } else if (rel === 'b_in_a') {
                if (mode === 'or') {
                    // B ⊆ A → A OR B = A → B лишнее
                    keep[j] = false;
                } else if (mode === 'and') {
                    // B ⊆ A → A AND B = B → A лишнее
                    keep[i] = false;
                    break;
                }
            }
        }
    }

    return atoms.filter((_, idx) => keep[idx]);
}

// Отношение между двумя cmp-условиями через интервалы
// 'a_in_b'  — A ⊆ B
// 'b_in_a'  — B ⊆ A
// 'equal'   — одинаковые интервалы (редко используем)
// null      — не можем определить
function cmpImplicationRelation(c1, c2) {
    const i1 = cmpToInterval(c1);
    const i2 = cmpToInterval(c2);
    if (!i1 || !i2) return null;
    if (i1.varName !== i2.varName) return null;

    const aInB = intervalSubset(i1, i2);
    const bInA = intervalSubset(i2, i1);

    if (aInB && bInA) return 'equal';
    if (aInB) return 'a_in_b';
    if (bInA) return 'b_in_a';
    return null;
}

// Разворот оператора при перестановке аргументов (левый/правый)
function reverseOp(op) {
    switch (op) {
        case '<':  return '>';
        case '>':  return '<';
        case '<=': return '>=';
        case '>=': return '<=';
        case '=':
        case '!=':
            return op;
        default:
            return null;
    }
}

// Аккуратный парсер числового литерала.
// Возвращает число или null, если строка не чисто числовая.
function parseNumberLiteral(s) {
    if (typeof s !== 'string') return null;
    const trimmed = s.trim().replace(',', '.');

    // Только простые вещи: -123, 45, 3.14
    if (!/^-?\d+(\.\d+)?$/.test(trimmed)) return null;

    const n = Number(trimmed);
    return Number.isFinite(n) ? n : null;
```

```
}


function negateAtomKey(key) {
    if (!key) return null;
    if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
    if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
    if (key.startsWith('cmp:')) {
        const parts = key.slice(4).split(':');
        if (parts.length === 3) {
            const negOp = negateOp(parts[1]);
            if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
        }
    }
    return null;
}

function isNegation(a, b) {
    if (!a || !b) return false;
    if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
    if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
    if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
        return a.op === negateOp(b.op);
    }
    if (a.type === 'not' && condEq(a.x, b)) return true;
    if (b.type === 'not' && condEq(b.x, a)) return true;
    return false;
}

function isAtomCond(t) {
    return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
}

function pruneOrByContext(orTerm, contextAtoms) {
    const branches = flattenOr(orTerm);
    const kept = [];

    for (const br of branches) {
        let contradicts = false;

        for (const ctx of contextAtoms) {
            if (isNegation(br, ctx)) {
                contradicts = true;
                break;
            }
        }

        if (!contradicts) kept.push(br);
    }

    if (kept.length === 0) return FalseCond;
    if (kept.length === 1) return kept[0];
    return buildOr(kept);
}

function condNegationEq(a, b) {
    if (!a || !b) return false;

    // Простая проверка: a == NOT(b)
    if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;

    // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
    // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
    if (a.type === 'or' && b.type === 'and') {
        return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
```

```
3422                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3423     }
3424     // Симметрично
3425     if (a.type === 'and' && b.type === 'or') {
3426         return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
3427                condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3428     }
3429
3430     // Проверка атомов: (X = 0) vs (X != 0)
3431     if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
3432     if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
3433
3434     // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
3435     if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
3436         return a.op === negateOp(b.op);
3437     }
3438
3439     return false;
3440 }
3441
3442
3443
3444 function condEq(a, b) {
3445     if (a === b) return true;
3446     if (!a || !b) return false;
3447     if (a.type !== b.type) return false;
3448
3449     switch (a.type) {
3450         case 'eq0':
3451         case 'ne0':
3452             return a.v === b.v;
3453         case 'cmp':
3454             return a.l === b.l && a.op === b.op && a.r === b.r;
3455         case 'true':
3456         case 'false':
3457             return true;
3458         case 'not':
3459             return condEq(a.x, b.x);
3460         case 'and':
3461         case 'or':
3462             return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
3463                    (condEq(a.a, b.b) && condEq(a.b, b.a));
3464         default:
3465             return false;
3466     }
3467 }
3468
3469 function flattenAnd(c) {
3470     if (!c) return [];
3471     if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
3472     return [c];
3473 }
3474
3475 function flattenOr(c) {
3476     if (!c) return [];
3477     if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
3478     return [c];
3479 }
3480
3481 function buildAnd(terms) {
3482     if (terms.length === 0) return TrueCond;
3483     let result = terms[0];
3484     for (let i = 1; i < terms.length; i++) {
3485         result = And(result, terms[i]);
3486     }
```

```
    return result;
}

function buildOr(terms) {
    if (terms.length === 0) return FalseCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = Or(result, terms[i]);
    }
    return result;
}

// Поглощение для AND: X AND (X OR Y) = X
function applyAndAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'or') continue;

        const orParts = flattenOr(ti);
        let drop = false;

        outer:
        for (const part of orParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
    }

    return terms.filter((_, idx) => keep[idx]);
}

// Поглощение для OR: X OR (X AND Y) = X
function applyOrAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'and') continue;

        const andParts = flattenAnd(ti);
        let drop = false;

        outer:
        for (const part of andParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
```

```
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
    }

    return terms.filter((_, idx) => keep[idx]);
}

// === Упрощение условий ===
function simplifyCond(c) {
    _depth++;
    if (_depth > MAX_DEPTH) {
        _depth--;
        return c;
    }

    try {
        return simplifyCondCore(c);
    } finally {
        _depth--;
    }
}

function simplifyCondCore(c) {
    if (!c || c.kind !== 'cond') return c;

    switch (c.type) {
        case 'true':
        case 'false':
        case 'eq0':
        case 'ne0':
        case 'cmp':
            return c;

        case 'not': {
            const x = simplifyCondCore(c.x);
            if (!x) return TrueCond;
            if (x.type === 'true') return FalseCond;
            if (x.type === 'false') return TrueCond;
            if (x.type === 'not') return simplifyCondCore(x.x);
            if (x.type === 'eq0') return Ne0(x.v);
            if (x.type === 'ne0') return Eq0(x.v);
            if (x.type === 'cmp') {
                const negOp = negateOp(x.op);
                if (negOp) return Cmp(x.l, negOp, x.r);
            }
            if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
            if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
            return Not(x);
        }

case 'and': {
    const a = simplifyCondCore(c.a);
    const b = simplifyCondCore(c.b);

    if (!a) return b;
    if (!b) return a;
    if (a.type === 'false' || b.type === 'false') return FalseCond;
    if (a.type === 'true') return b;
    if (b.type === 'true') return a;
```

```
3617
3618      const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
3619
3620      // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
3621      const eq0Vars = new Map(); // var -> term
3622      const ne0Vars = new Map(); // var -> term
3623      const cmpTerms = [];
3624      const otherTerms = [];
3625
3626      for (const t of allTerms) {
3627          if (t.type === 'true') continue;
3628          if (t.type === 'false') return FalseCond;
3629
3630          if (t.type === 'eq0') {
3631              // Проверка на противоречие сразу
3632              if (ne0Vars.has(t.v)) {
3633                  console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
3634                  return FalseCond;
3635              }
3636              eq0Vars.set(t.v, t);
3637          } else if (t.type === 'ne0') {
3638              // Проверка на противоречие сразу
3639              if (eq0Vars.has(t.v)) {
3640                  console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
3641                  return FalseCond;
3642              }
3643              ne0Vars.set(t.v, t);
3644          } else if (t.type === 'cmp') {
3645              cmpTerms.push(t);
3646          } else if (t.type === 'or') {
3647              // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
3648              const orTerms = flattenOr(t);
3649              const validBranches = [];
3650
3651              for (const branch of orTerms) {
3652                  let branchValid = true;
3653
3654                  if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
3655                      console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
      {branch.v} = 0`);
3656                      branchValid = false;
3657                  } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
3658                      console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
      {branch.v} != 0`);
3659                      branchValid = false;
3660                  }
3661
3662                  if (branchValid) {
3663                      validBranches.push(branch);
3664                  }
3665              }
3666
3667              if (validBranches.length === 0) {
3668                  console.log(`Все ветки OR противоречат контексту → FALSE`);
3669                  return FalseCond;
3670              } else if (validBranches.length === 1) {
3671                  // Если осталась только одна ветка OR, добавляем её напрямую
3672                  const singleBranch = validBranches[0];
3673                  if (singleBranch.type === 'eq0') {
3674                      if (ne0Vars.has(singleBranch.v)) return FalseCond;
3675                      eq0Vars.set(singleBranch.v, singleBranch);
3676                  } else if (singleBranch.type === 'ne0') {
3677                      if (eq0Vars.has(singleBranch.v)) return FalseCond;
3678                      ne0Vars.set(singleBranch.v, singleBranch);
3679                  } else {
```

```
                    otherTerms.push(singleBranch);
                }
            } else {
                // Перестраиваем OR только с валидными ветками
                otherTerms.push(buildOr(validBranches));
            }
        } else {
            otherTerms.push(t);
        }
    }

    // Собираем уникальные атомы
    const atomMap = new Map();

    for (const [v, term] of eq0Vars) {
        const key = atomKey(term);
        if (key) atomMap.set(key, term);
    }

    for (const [v, term] of ne0Vars) {
        const key = atomKey(term);
        if (key) atomMap.set(key, term);
    }

    for (const term of cmpTerms) {
        const key = atomKey(term);
        if (key) {
            const negKey = negateAtomKey(key);
            if (negKey && atomMap.has(negKey)) {
                return FalseCond;
            }
            if (!atomMap.has(key)) {
                atomMap.set(key, term);
            }
        }
    }

    let uniqueAtoms = Array.from(atomMap.values());
    uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');

    let result = [...uniqueAtoms, ...otherTerms];

    // Поглощение: X AND (X OR Y) = X
    // === НОВОЕ: выбрасываем из OR ветки, противоречащие контексту AND ===
    const contextAtoms = result.filter(t => isAtomCond(t));
    result = result.map(t => {
        if (t.type !== 'or') return t;
        return pruneOrByContext(t, contextAtoms);
    }).filter(t => t.type !== 'true'); // на всякий случай

    result = applyAndAbsorption(result);

    if (result.length === 0) return TrueCond;
    if (result.length === 1) return result[0];

    return buildAnd(result);
}

        case 'or': {
            const a = simplifyCondCore(c.a);
            const b = simplifyCondCore(c.b);

            if (!a) return b;
            if (!b) return a;
            if (a.type === 'true' || b.type === 'true') return TrueCond;
```

```
            if (a.type === 'false') return b;
            if (b.type === 'false') return a;

            const allTerms = [...flattenOr(a), ...flattenOr(b)];
            const atomMap = new Map();
            const otherTerms = [];

            for (const t of allTerms) {
                if (t.type === 'true') return TrueCond;
                if (t.type === 'false') continue;

                const key = atomKey(t);
                if (key) {
                    const negKey = negateAtomKey(key);
                    if (negKey && atomMap.has(negKey)) {
                        return TrueCond;
                    }
                    if (!atomMap.has(key)) {
                        atomMap.set(key, t);
                    }
                } else {
                    otherTerms.push(t);
                }
            }

            let uniqueAtoms = Array.from(atomMap.values());
            uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');

            let result = [...uniqueAtoms, ...otherTerms];

            // Поглощение: X OR (X AND Y) = X
            result = applyOrAbsorption(result);

            if (result.length === 0) return FalseCond;
            if (result.length === 1) return result[0];

            return buildOr(result);
        }

        default:
            return c;
    }
}

// === Сравнение выражений ===
function exprEq(a, b) {
    if (a === b) return true;
    if (!a && !b) return true;
    if (!a || !b) return false;
    if (a.type !== b.type) return false;

    switch (a.type) {
        case 'const': return a.n === b.n;
        case 'var': return a.name === b.name;
        case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
        case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
        default: return false;
    }
}

// === Упрощение выражений ===
function simplifyExpr(expr) {
    _depth++;
    if (_depth > MAX_DEPTH) {
        _depth--;
```

```
3810          return expr;
3811      }
3812
3813      try {
3814          return simplifyExprCore(expr);
3815      } finally {
3816          _depth--;
3817      }
3818  }
3819
3820  function simplifyExprCore(expr) {
3821      if (!expr || expr.kind !== 'expr') return expr;
3822
3823      switch (expr.type) {
3824          case 'const':
3825          case 'var':
3826              return expr;
3827
3828          case 'op': {
3829              const l = simplifyExprCore(expr.l);
3830              const r = simplifyExprCore(expr.r);
3831
3832              if (expr.op === '+') {
3833                  if (r?.type === 'const' && r.n === 0) return l;
3834                  if (l?.type === 'const' && l.n === 0) return r;
3835              }
3836              if (expr.op === '*') {
3837                  if (l?.type === 'const' && l.n === 0) return Const(0);
3838                  if (r?.type === 'const' && r.n === 0) return Const(0);
3839                  if (l?.type === 'const' && l.n === 1) return r;
3840                  if (r?.type === 'const' && r.n === 1) return l;
3841              }
3842              return Op(expr.op, l, r);
3843          }
3844
3845          case 'when': {
3846              const c = simplifyCond(expr.c);
3847              const t = simplifyExprCore(expr.t);
3848              const e = simplifyExprCore(expr.e);
3849
3850              if (c?.type === 'true') return t;
3851              if (c?.type === 'false') return e;
3852              if (exprEq(t, e)) return t;
3853              // ✅ НОВОЕ: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
3854              if (e && e.type === 'when') {
3855                  const c2 = simplifyCond(e.c);
3856                  const t2 = simplifyExprCore(e.t);
3857                  const e2 = simplifyExprCore(e.e);
3858
3859                  if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
3860                      return When(c, t, t2);
3861                  }
3862              }
3863              // Узкое правило: WHEN(A∧B, t1, WHEN(A∧¬B, t2, WHEN(¬A, t3, e3))) -> ...
      t3
3864              if (e && e.type === 'when') {
3865                  const c2 = e.c, t2 = e.t, e2 = e.e;
3866
3867                  if (e2 && e2.type === 'when') {
3868                      const c3 = e2.c, t3 = e2.t;
3869
3870                      const shared = findSharedAndComplement(c, c2);
3871                      if (shared && condNegationEq(c3, shared.shared)) {
3872                          return When(c, t, When(c2, t2, t3));
3873                      }
```

```
                }
            }

            return When(c, t, e);
        }

        default:
            return expr;
    }
}

// === Печать ===
function printCond(c) {
    if (!c) return 'TRUE';

    switch (c.type) {
        case 'eq0': return `(${c.v} = 0)`;
        case 'ne0': return `(${c.v} != 0)`;
        case 'cmp': return `(${c.l} ${c.op} ${c.r})`;
        case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
        case 'or':  return `(${printCond(c.a)} OR ${printCond(c.b)})`;
        case 'not': return `NOT(${printCond(c.x)})`;
        case 'true': return 'TRUE';
        case 'false': return 'FALSE';
        default: return '?';
    }
}

function printExpr(e) {
    if (!e) return '0';

    switch (e.type) {
        case 'const': return String(e.n);
        case 'var': return e.name;
        case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
        case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, ${printExpr(e.e)})`;
        default: return '?';
    }
}

window.Optimizer = {
    Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
    Const, Var, Op, When,
    simplifyCond, simplifyExpr,
    printCond, printExpr,
    condEq, exprEq
};

// js/codegen.js

const CodeGen = {
    _cache: {},
    _branchCache: {},
    _resolveCache: {},
    _visiting: new Set(),

    reset() {
        this._cache = {};
        this._branchCache = {};
        this._resolveCache = {};
        this._visiting = new Set();
    },

    toExpr(valueStr) {
```

```
3938          const s = String(valueStr).trim();
3939          if (s === '0') return Optimizer.Const(0);
3940          const num = parseFloat(s);
3941          if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
3942          return Optimizer.Var(s);
3943      },
3944
3945      exprToName(exprAst) {
3946          if (!exprAst) return '0';
3947          if (exprAst.type === 'var') return exprAst.name;
3948          if (exprAst.type === 'const') return String(exprAst.n);
3949          return Optimizer.printExpr(exprAst);
3950      },
3951
3952      mergeCond(a, b) {
3953          if (!a && !b) return null;
3954          if (!a) return b;
3955          if (!b) return a;
3956          if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
3957          return Optimizer.And(a, b);
3958      },
3959
3960      getConn(toId, toPort) {
3961          return AppState.connections.find(c => c.toElement === toId && c.toPort ===
      toPort);
3962      },
3963
3964      getConns(toId, prefix) {
3965          return AppState.connections.filter(c => c.toElement === toId &&
      c.toPort.startsWith(prefix));
3966      },
3967
3968      buildFormulaExpr(elem) {
3969          let result = elem.props.expression || '0';
3970
3971          // 1) Сначала раскрываем шаблоны (h и др.)
3972          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
3973              ? Settings.getTemplatesMap()
3974              : null;
3975          result = expandFormulaTemplates(result, map);
3976
3977          // 2) Потом раскрываем ссылки на формулы
3978          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
3979          for (const ref of formulaRefs) {
3980              const refElem = AppState.elements[ref];
3981              if (refElem && refElem.type === 'formula') {
3982                  const refExpr = this.buildFormulaExpr(refElem);
3983                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
3984              }
3985          }
3986
3987          return result;
3988      },
3989
3990      // === Получить ЧИСТУЮ логику элемента ===
3991      getPureLogic(id) {
3992          const cacheKey = `logic:${id}`;
3993          if (cacheKey in this._cache) {
3994              return this._cache[cacheKey];
3995          }
3996
3997          const elem = AppState.elements[id];
3998          if (!elem) return null;
3999
4000          let logic = null;
```

```
4001
4002          switch (elem.type) {
4003              case 'if': {
4004                  const leftConn = this.getConn(id, 'in-0');
4005                  const rightConn = this.getConn(id, 'in-1');
4006
4007                  const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
      Optimizer.Const(0);
4008                  const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
      Optimizer.Const(0);
4009
4010                  const op = (elem.props.operator || '=').trim();
4011                  const leftName = this.exprToName(leftVal);
4012                  const rightName = this.exprToName(rightVal);
4013
4014                  const leftZero = leftVal.type === 'const' && leftVal.n === 0;
4015                  const rightZero = rightVal.type === 'const' && rightVal.n === 0;
4016
4017                  switch (op) {
4018                      case '=':
4019                          if (rightZero) {
4020                              logic = Optimizer.Eq0(leftName);
4021                          } else if (leftZero) {
4022                              logic = Optimizer.Eq0(rightName);
4023                          } else {
4024                              logic = Optimizer.Cmp(leftName, '=', rightName);
4025                          }
4026                          break;
4027                      case '!=':
4028                          if (rightZero) {
4029                              logic = Optimizer.Ne0(leftName);
4030                          } else if (leftZero) {
4031                              logic = Optimizer.Ne0(rightName);
4032                          } else {
4033                              logic = Optimizer.Cmp(leftName, '!=', rightName);
4034                          }
4035                          break;
4036                      case '>':
4037                      case '<':
4038                      case '>=':
4039                      case '<=':
4040                          logic = Optimizer.Cmp(leftName, op, rightName);
4041                          break;
4042                      default:
4043                          logic = Optimizer.TrueCond;
4044                  }
4045                  break;
4046              }
4047
4048              case 'and':
4049              case 'or': {
4050                  const isAnd = elem.type === 'and';
4051                  const count = elem.props.inputCount || 2;
4052                  let result = null;
4053
4054                  for (let i = 0; i < count; i++) {
4055                      const conn = this.getConn(id, `in-${i}`);
4056                      if (!conn) continue;
4057
4058                      const val = this.getPureLogic(conn.fromElement);
4059                      if (!val) continue;
4060
4061                      if (result === null) {
4062                          result = val;
4063                      } else {
```

```
4064                         result = isAnd ? Optimizer.And(result, val) :
      Optimizer.Or(result, val);
4065                     }
4066                 }
4067                 logic = result || Optimizer.FalseCond;
4068                 break;
4069             }
4070 
4071             case 'not': {
4072                 const conn = this.getConn(id, 'in-0');
4073                 const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
4074                 logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
4075                 break;
4076             }
4077 
4078             case 'separator': {
4079                 const conn = this.getConn(id, 'in-0');
4080                 logic = conn ? this.getPureLogic(conn.fromElement) :
      Optimizer.FalseCond;
4081                 break;
4082             }
4083 
4084             default:
4085                 logic = null;
4086         }
4087 
4088         // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
4089         if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
      elem.type === 'not') {
4090             const ctx = this.getConditionFromPort(id);
4091             if (ctx) {
4092                 if (logic) {
4093                     logic = Optimizer.And(ctx, logic);
4094                 } else {
4095                     logic = ctx;
4096                 }
4097             }
4098         }
4099 
4100         this._cache[cacheKey] = logic;
4101         return logic;
4102     },
4103 
4104     // === Получить значение ===
4105     getValue(id) {
4106         const elem = AppState.elements[id];
4107         if (!elem) return Optimizer.Const(0);
4108 
4109         switch (elem.type) {
4110             case 'input-signal':
4111                 // Имя сигнала или id как Var(...)
4112                 return this.toExpr(elem.props.name || id);
4113 
4114             case 'const':
4115                 return Optimizer.Const(Number(elem.props.value) || 0);
4116 
4117             case 'formula': {
4118                 // Используем текст формулы как выражение
4119                 const exprStr = this.buildFormulaExpr(elem) || '0';
4120                 return this.toExpr(exprStr);
4121             }
4122 
4123             default:
4124                 // На всякий случай — даём символическое имя, а не 0
4125                 if (elem.props && typeof elem.props.name === 'string') {
```

```
                    return this.toExpr(elem.props.name);
                }
                return this.toExpr(id);
            }
        },

        // === Получить ПОЛНОЕ условие для ветки сепаратора ===
        getBranchCondition(sepId, fromPort) {
            const cacheKey = `${sepId}:${fromPort}`;
            if (cacheKey in this._branchCache) {
                return this._branchCache[cacheKey];
            }

            const sep = AppState.elements[sepId];
            if (!sep || sep.type !== 'separator') return null;

            const inputLogic = this.getPureLogic(sepId);
            const sepContext = this.getConditionFromPort(sepId);

            let branchLogic;
            if (fromPort === 'out-1') {
                branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
            } else {
                branchLogic = inputLogic || Optimizer.TrueCond;
            }

            let result;
            if (sepContext) {
                result = Optimizer.And(sepContext, branchLogic);
            } else {
                result = branchLogic;
            }

            this._branchCache[cacheKey] = result;
            return result;
        },

        // === Получить условие от cond-порта ===
        getConditionFromPort(id) {
            const conn = this.getConn(id, 'cond-0');
            if (!conn) return null;

            const sourceElem = AppState.elements[conn.fromElement];
            if (!sourceElem) return null;

            if (sourceElem.type === 'separator') {
                return this.getBranchCondition(conn.fromElement, conn.fromPort);
            }

            return this.getPureLogic(conn.fromElement);
        },

        // === Основная функция разрешения ===
        resolve(id) {
            if (id in this._resolveCache) {
                return this._resolveCache[id];
            }

            if (this._visiting.has(id)) {
                return null;
            }
            this._visiting.add(id);

            const elem = AppState.elements[id];
            if (!elem) {
```

```
4191              this._visiting.delete(id);
4192              return null;
4193          }
4194
4195          let result = null;
4196
4197          try {
4198              switch (elem.type) {
4199                  case 'input-signal':
4200                      result = {
4201                          isValue: true,
4202                          cond: null,
4203                          expr: this.toExpr(elem.props.name || id)
4204                      };
4205                      break;
4206
4207                  case 'const': {
4208                      const cond = this.getConditionFromPort(id);
4209                      result = {
4210                          isValue: true,
4211                          cond: cond,
4212                          expr: Optimizer.Const(Number(elem.props.value) || 0)
4213                      };
4214                      break;
4215                  }
4216
4217                  case 'formula': {
4218                      let cond = this.getConditionFromPort(id);
4219
4220                      const inConns = this.getConns(id, 'in-');
4221                      for (const conn of inConns) {
4222                          const inputNode = this.resolve(conn.fromElement);
4223                          if (inputNode && inputNode.cond) {
4224                              cond = this.mergeCond(cond, inputNode.cond);
4225                          }
4226                      }
4227
4228                      const fullExpr = this.buildFormulaExpr(elem);
4229                      result = {
4230                          isValue: true,
4231                          cond: cond,
4232                          expr: Optimizer.Var(fullExpr)
4233                      };
4234                      break;
4235                  }
4236
4237                  default:
4238                      result = null;
4239              }
4240          } finally {
4241              this._visiting.delete(id);
4242          }
4243
4244          this._resolveCache[id] = result;
4245          return result;
4246      },
4247
4248      generate() {
4249          console.log('=== Генерация кода (граф) ===');
4250          this.reset();
4251
4252          try {
4253              const outputs = Object.values(AppState.elements).filter(e => e.type ===
      'output');
4254
```

```
4255             if (outputs.length === 0) {
4256                 return '/* Нет выходов */';
4257             }
4258 
4259             const allVariants = [];
4260 
4261             for (const out of outputs) {
4262                 const conns = this.getConns(out.id, 'in-');
4263 
4264                 for (const conn of conns) {
4265                     console.log(`\n=== Обработка выхода ${out.id}, вход от $
     {conn.fromElement} ===`);
4266                     const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
4267                     const result = CodeGenGraph.evalGraphValue(graph);
4268                     console.log(`Результат: cond=${Optimizer.printCond(result.cond)},
     expr=${Optimizer.printExpr(result.expr)}`);
4269 
4270                     if (!result || !result.expr) continue;
4271 
4272                     const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
     null;
4273                     const isZero = result.expr.type === 'const' && result.expr.n ===
     0;
4274 
4275                     if (isZero && !cond) continue;
4276 
4277                     allVariants.push({
4278                         cond,
4279                         expr: result.expr,
4280                         isZero
4281                     });
4282                 }
4283             }
4284 
4285             console.log('Варианты:', allVariants.map(v => ({
4286                 cond: Optimizer.printCond(v.cond),
4287                 expr: Optimizer.printExpr(v.expr)
4288             })));
4289 
4290             if (allVariants.length === 0) return '0';
4291 
4292             const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
4293             if (valueVariants.length === 0) return '0';
4294 
4295             let result = Optimizer.Const(0);
4296 
4297             for (let i = valueVariants.length - 1; i >= 0; i--) {
4298                 const v = valueVariants[i];
4299                 if (v.cond) {
4300                     result = Optimizer.When(v.cond, v.expr, result);
4301                 } else {
4302                     result = v.expr;
4303                 }
4304             }
4305 
4306             const simplified = Optimizer.simplifyExpr(result);
4307             return Optimizer.printExpr(simplified);
4308 
4309         } catch (err) {
4310             console.error('Ошибка:', err);
4311             return `/* Ошибка: ${err.message} */`;
4312         }
4313     }
4314 };
4315 
```

```
window.CodeGen = CodeGen;

/**
 * Конфигурация приложения
 * config.js
 */

// Типы сигналов
const SIGNAL_TYPE = {
    NUMERIC: 'numeric',   // Числовой сигнал
    LOGIC: 'logic',       // Логический (может быть TRUE или FALSE)
    TRUE: 'true',         // Явно ИСТИНА
    FALSE: 'false',       // Явно ЛОЖЬ
    ANY: 'any'            // Любой тип
};

// Типы проекта
const PROJECT_TYPE = {
    PARAMETER: 'parameter',
    RULE: 'rule'
};

// Конфигурация элементов
const ELEMENT_TYPES = {
    'input-signal': {
        name: 'Вход',
        inputs: 0,
        outputs: 1,
        outputLabels: ['out'],
        outputTypes: [SIGNAL_TYPE.NUMERIC],
        color: '#4a90d9',
        hasProperties: true,
        defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
        resizable: true,
        minWidth: 150,
        minHeight: 50
    },
    'and': {
        name: 'И',
        inputs: 2,  // По умолчанию 2, но может быть изменено
        outputs: 1,
        inputLabels: ['A', 'B'],
        inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
        outputLabels: ['результат'],
        outputTypes: [SIGNAL_TYPE.LOGIC],
        color: '#a855f7',
        hasProperties: true,  // ← Теперь есть свойства (для изменения количества входов)
        resizable: true,
        minWidth: 120,
        minHeight: 80,
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC,
        defaultProps: {
            inputCount: 2  // ← Новое свойство
        }
    },
    'or': {
        name: 'ИЛИ',
        inputs: 2,  // По умолчанию 2
        outputs: 1,
        inputLabels: ['A', 'B'],
        inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
        outputLabels: ['результат'],
        outputTypes: [SIGNAL_TYPE.LOGIC],
```

```
4380            color: '#a855f7',
4381            hasProperties: true,  // ← Теперь есть свойства
4382            resizable: true,
4383            minWidth: 120,
4384            minHeight: 80,
4385            hasConditionPort: true,
4386            conditionPortType: SIGNAL_TYPE.LOGIC,
4387            defaultProps: {
4388                inputCount: 2  // ← Новое свойство
4389            }
4390        },
4391        'not': {
4392            name: 'НЕ',
4393            inputs: 1,
4394            outputs: 1,
4395            inputLabels: ['A'],
4396            inputTypes: [SIGNAL_TYPE.LOGIC],
4397            outputLabels: ['¬A'],
4398            outputTypes: [SIGNAL_TYPE.LOGIC],
4399            color: '#a855f7',
4400            hasProperties: true,
4401            resizable: true,
4402            minWidth: 100,
4403            minHeight: 60,
4404            hasConditionPort: true,
4405            conditionPortType: SIGNAL_TYPE.LOGIC
4406        },
4407        'if': {
4408            name: 'ЕСЛИ',
4409            inputs: 2,
4410            outputs: 1,  // ← Только один выход!
4411            inputLabels: ['A', 'B'],
4412            inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
4413            outputLabels: ['результат'],  // ← Просто результат
4414            outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
4415            color: '#e94560',
4416            hasProperties: true,
4417            defaultProps: { operator: '=' },
4418            resizable: true,
4419            minWidth: 120,
4420            minHeight: 80,
4421            hasConditionPort: true,
4422            conditionPortType: SIGNAL_TYPE.LOGIC
4423        },
4424        'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
4425            name: 'Сепаратор',
4426            inputs: 1,
4427            outputs: 2,
4428            inputLabels: ['сигнал'],
4429            inputTypes: [SIGNAL_TYPE.LOGIC],
4430            outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
4431            outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
4432            color: '#f59e0b',
4433            hasProperties: true,
4434            resizable: true,
4435            minWidth: 120,
4436            minHeight: 80,
4437            hasConditionPort: true,
4438            conditionPortType: SIGNAL_TYPE.LOGIC
4439        },
4440        'const': {
4441            name: 'Константа',
4442            inputs: 0,
4443            outputs: 1,
4444            outputLabels: ['out'],
```

```
        outputTypes: [SIGNAL_TYPE.NUMERIC],
        color: '#3b82f6',
        hasProperties: true,
        defaultProps: { value: 0 },
        resizable: true,
        minWidth: 120,
        minHeight: 60,
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC
    },
    'formula': {
        name: 'Формула',
        inputs: 2,
        outputs: 1,
        inputLabels: ['in₁', 'in₂'],
        inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
        outputLabels: ['результат'],
        outputTypes: [SIGNAL_TYPE.NUMERIC],
        color: '#f59e0b',
        hasProperties: true,
        resizable: true,
        minWidth: 140,
        minHeight: 80,
        defaultProps: {
            expression: '',
            inputCount: 2
        },
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC
    },
    'output': {
        name: 'Выход',
        inputs: 1,
        outputs: 0,
        inputLabels: ['сигнал'],
        inputTypes: [SIGNAL_TYPE.ANY],
        color: '#10b981',
        hasProperties: true,
        defaultProps: { label: 'Выход', outputGroup: '' },
        resizable: true,
        minWidth: 150,
        minHeight: 60,
    },  // ← важно, если предыдущий элемент не заканчивается запятой
    'group': {
        name: 'Группа',
        inputs: 0,
        outputs: 0,
        color: '#6b7280',
        resizable: true,
        minWidth: 200,
        minHeight: 120,
        hasProperties: true,
        defaultProps: { title: 'Группа' }
    }
};

const VIEWPORT_CONFIG = {
    minZoom: 0.1,
    maxZoom: 3,
    zoomStep: 0.1,
    panSpeed: 1,
    canvasWidth: 5000,
    canvasHeight: 5000
};

```

```
const MINIMAP_CONFIG = {
    width: 200,
    height: 150,
    padding: 10
};

/**
 * Модуль работы с соединениями
 * connections.js
 */

const Connections = {
    /**
     * Настройка обработчиков порта
     */
    setupPortHandlers(port) {
        port.addEventListener('mousedown', (e) => {
            e.stopPropagation();

            if (port.classList.contains('output')) {
                const elemId = port.dataset.element;
                const portName = port.dataset.port;
                const signalType = getOutputPortType(elemId, portName);

                AppState.connectingFrom = {
                    element: elemId,
                    port: portName
                };
                AppState.connectingFromType = signalType;

                this.highlightCompatiblePorts(signalType);

                const svg = document.getElementById('connections-svg');
                const startPos = this._getPortCanvasCenter(port);

                AppState.tempLine = document.createElementNS('http://www.w3.org/2000/svg', 'path');
                AppState.tempLine.setAttribute('class', 'temp-connection');
                AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L ${startPos.x} ${startPos.y}`);
                svg.appendChild(AppState.tempLine);
            }
        });

        port.addEventListener('mouseup', (e) => {
            e.stopPropagation();
            e.preventDefault();

            if (AppState.connectingFrom && port.classList.contains('input')) {
                const toElement = port.dataset.element;
                const toPortName = port.dataset.port;
                const inputType = getInputPortType(toElement, toPortName);

                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
                    this.clearConnectionState();
                    return;
                }

                if (AppState.connectingFrom.element !== toElement) {
                    const targetElem = AppState.elements[toElement];
                    const allowMultipleInputs = targetElem?.type === 'output';

                    const exists = AppState.connections.some(c =>
                        c.toElement === toElement && c.toPort === toPortName
                    );
```

```
4573
4574                    if (!exists || allowMultipleInputs) {
4575                        AppState.connections.push({
4576                            fromElement: AppState.connectingFrom.element,
4577                            fromPort: AppState.connectingFrom.port,
4578                            toElement,
4579                            toPort: toPortName,
4580                            signalType: AppState.connectingFromType
4581                        });
4582
4583                        port.classList.add('connected');
4584                        this.drawConnections();
4585                        this.clearConnectionState();
4586                        return;
4587                    }
4588                }
4589            }
4590
4591            this.clearConnectionState();
4592        });
4593
4594        port.addEventListener('mouseenter', () => {
4595            if (AppState.connectingFrom && port.classList.contains('input')) {
4596                const toPortName = port.dataset.port;
4597                const inputType = getInputPortType(port.dataset.element, toPortName);
4598
4599                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
4600                    if (AppState.tempLine) {
4601                        AppState.tempLine.classList.add('invalid');
4602                    }
4603                }
4604            }
4605        });
4606
4607        port.addEventListener('mouseleave', () => {
4608            if (AppState.tempLine) {
4609                AppState.tempLine.classList.remove('invalid');
4610            }
4611        });
4612    },
4613
4614    /**
4615     * Подсветка совместимых портов
4616     */
4617    highlightCompatiblePorts(signalType) {
4618        document.querySelectorAll('.port.input').forEach(port => {
4619            const inputType = getInputPortType(port.dataset.element,
     port.dataset.port);
4620
4621            if (areTypesCompatible(signalType, inputType)) {
4622                port.classList.add('compatible-highlight');
4623            } else {
4624                port.classList.add('incompatible');
4625            }
4626        });
4627    },
4628
4629    /**
4630     * Очистка состояния соединения
4631     */
4632    clearConnectionState() {
4633        if (AppState.tempLine) {
4634            AppState.tempLine.remove();
4635            AppState.tempLine = null;
4636        }
```

```
4637          AppState.connectingFrom = null;
4638          AppState.connectingFromType = null;
4639
4640          document.querySelectorAll('.port').forEach(port => {
4641              port.classList.remove('compatible-highlight', 'incompatible');
4642          });
4643      },
4644
4645      /**
4646       * Отрисовка временной линии соединения
4647       */
4648      drawTempConnection(e) {
4649      if (!AppState.tempLine || !AppState.connectingFrom) return;
4650
4651      const fromElem = document.getElementById(AppState.connectingFrom.element);
4652      if (!fromElem) return;
4653
4654      const fromPort = fromElem.querySelector(`[data-port="$
     {AppState.connectingFrom.port}"]`);
4655      if (!fromPort) return;
4656
4657      const startPos = this._getPortCanvasCenter(fromPort);
4658      const endPos = screenToCanvas(e.clientX, e.clientY);
4659
4660      const horizontalDist = Math.abs(endPos.x - startPos.x);
4661      const controlDist = Math.max(horizontalDist * 0.4, 50);
4662
4663      // Тянем всегда от выхода (вектор 1, 0)
4664      const cx1 = startPos.x + controlDist;
4665      const cy1 = startPos.y;
4666
4667      // Вторая точка контроля для плавности за курсором
4668      const cx2 = endPos.x - controlDist;
4669      const cy2 = endPos.y;
4670
4671      AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} $
     {cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
4672      AppState.tempLine.setAttribute('fill', 'none');
4673  },
4674
4675      /**
4676       * Отрисовка всех соединений
4677       */
4678      drawConnections() {
4679      const svg = document.getElementById('connections-svg');
4680
4681      // 1. Очистка старых линий
4682      svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
4683
4684      // 2. Сброс визуального состояния портов
4685      document.querySelectorAll('.port.connected').forEach(port => {
4686          port.classList.remove('connected');
4687      });
4688
4689      // 3. Перебор всех соединений из AppState
4690      AppState.connections.forEach(conn => {
4691          const fromElem = document.getElementById(conn.fromElement);
4692          const toElem = document.getElementById(conn.toElement);
4693
4694          if (!fromElem || !toElem) return;
4695
4696          const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
4697          const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
4698
4699          if (!fromPort || !toPort) return;
```

```
4700
4701          fromPort.classList.add('connected');
4702          toPort.classList.add('connected');
4703
4704          const startPos = this._getPortCanvasCenter(fromPort);
4705          const endPos = this._getPortCanvasCenter(toPort);
4706
4707          if (!startPos || !endPos) return;
4708
4709          // Расстояние для изгиба кривой
4710          const horizontalDist = Math.abs(endPos.x - startPos.x);
4711          const verticalDist = Math.abs(endPos.y - startPos.y);
4712          const controlDist = Math.max(horizontalDist * 0.4, 50);
4713
4714          // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
4715          let d;
4716          let cx1 = startPos.x;
4717          let cy1 = startPos.y;
4718          let cx2 = endPos.x;
4719          let cy2 = endPos.y;
4720
4721          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
4722          cx1 = startPos.x + controlDist;
4723          cy1 = startPos.y;
4724
4725          // ВХОД (Target):
4726          if (conn.toPort === 'cond-0') {
4727              // Технический порт: Касательная (0, 1) в декартовой (вверх)
4728              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
4729              cx2 = endPos.x;
4730              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
4731          } else {
4732              // Обычный вход: Касательная (-1, 0) -> Слева направо
4733              cx2 = endPos.x - controlDist;
4734              cy2 = endPos.y;
4735          }
4736
4737          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
      ${endPos.y}`;
4738
4739          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
4740          path.setAttribute('d', d);
4741          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
4742
4743          // --- ЛОГИКА ЦВЕТА (Классы) ---
4744          let cssClass = 'connection';
4745          const type = conn.signalType;
4746
4747          // Приоритет новым типам сигналов
4748          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
4749          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
4750          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
4751          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
4752          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
4753
4754          path.setAttribute('class', cssClass);
4755
4756          // Обработчики событий
4757          path.style.pointerEvents = 'stroke';
4758          path.style.cursor = 'pointer';
4759          path.addEventListener('click', () => this.handleConnectionClick(conn));
4760
4761          svg.appendChild(path);
4762      });
4763
```

```
4764      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
4765          Outputs.updateOutputStatus();
4766      }
4767      Viewport.updateMinimap();
4768  },
4769      /**
4770       * Обработка клика по соединению (удаление)
4771       */
4772      handleConnectionClick(conn) {
4773          if (confirm('Удалить соединение?')) {
4774              AppState.connections = AppState.connections.filter(c =>
4775                  !(c.fromElement === conn.fromElement &&
4776                    c.fromPort === conn.fromPort &&
4777                    c.toElement === conn.toElement &&
4778                    c.toPort === conn.toPort)
4779              );
4780
4781              this.drawConnections();
4782          }
4783      },
4784
4785      /**
4786       * Получение центра порта в координатах Canvas
4787       */
4788      _getPortCanvasCenter(portEl) {
4789          if (!portEl) return null;
4790
4791          const rect = portEl.getBoundingClientRect();
4792          return screenToCanvas(
4793              rect.left + rect.width / 2,
4794              rect.top + rect.height / 2
4795          );
4796      }
4797  };
4798
4799  /**
4800   * Модуль работы с элементами схемы
4801   * elements.js
4802   */
4803
4804  const Elements = {
4805      /**
4806       * Генерация HTML для элемента
4807       */
4808          createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
4809              const config = ELEMENT_TYPES[elemType];
4810              if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
4811
4812              const safe = (value, fallback = '') => (value === null || value ===
      undefined) ? fallback : String(value);
4813              const w = width ?? config.minWidth ?? 120;
4814              const h = height ?? config.minHeight ?? 60;
4815
4816              const getPortClass = (signalType, direction) => {
4817                  const base = direction === 'output' ? 'port output' : 'port input';
4818                  if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
4819                  if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
4820                  return `${base} any-port`;
4821              };
4822
4823              // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
      innerHTML
4824              // Она тут остается, но ее результат не встраивается в HTML-строку
      напрямую, кроме формулы
4825              const buildConditionPortHTML = () => {
```

```
4826                 return `
4827                     <div class="condition-port-wrapper">
4828                         <div class="condition-port-label">условие</div>
4829                         <div class="port input condition-port"
4830                             data-port="cond-0"
4831                             data-element="${elemId}"
4832                             data-signal-type="${SIGNAL_TYPE.LOGIC}"
4833                             title="Техническое условие">
4834                         </div>
4835                     </div>`;
4836             };
4837 
4838 
4839             const buildInputPorts = (count, types = [], labels = []) => {
4840                 let html = '';
4841                 for (let i = 0; i < count; i++) {
4842                     const type = types[i] ?? types[types.length - 1] ??
    SIGNAL_TYPE.ANY;
4843                     html += `<div class="${getPortClass(type, 'input')}" data-
    port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
    || `Вход ${i+1}`}"></div>`;
4844                 }
4845                 return html;
4846             };
4847 
4848             const buildOutputPorts = (count, types = [], labels = []) => {
4849                 let html = '';
4850                 for (let i = 0; i < count; i++) {
4851                     const type = types[i] ?? types[types.length - 1] ??
    SIGNAL_TYPE.ANY;
4852                     html += `<div class="${getPortClass(type, 'output')}" data-
    port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
    || `Выход ${i+1}`}"></div>`;
4853                 }
4854                 return html;
4855             };
4856 
4857             const resizeHandles = config.resizable ? `<div class="resize-handle
    handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
    direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
    '';
4858             // hasCondClass будет добавляться в addElement
4859             // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
    '';
4860 
4861             let innerHTML = '';
4862 
4863             if (elemType === 'input-signal') {
4864                 const name = safe(props.name, 'Сигнал');
4865                 const type = props.signalType || SIGNAL_TYPE.NUMBER;
4866                 const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
4867                 innerHTML = `
4868                     <div class="element-header" style="background:$
    {config.color};">Источник</div>
4869                     <div class="element-body">
4870                         <div class="element-symbol">
4871                             <span class="input-signal-icon">${symbol}</span>
4872                             <span class="input-signal-name">${name}</span>
4873                         </div>
4874                         <div class="ports-right">
4875                             ${buildOutputPorts(1, [type], ['Выход'])}
4876                         </div>
4877                     </div>`;
4878             }
4879             else if (elemType === 'const') {
```

```
4880                 innerHTML = `
4881                     <div class="element-header" style="background:$
     {config.color};">Константа</div>
4882                     <div class="element-body">
4883                         <div class="element-symbol">${props.value ?? 0}</div>
4884                         <div class="ports-right">
4885                             ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
4886                         </div>
4887                     </div>`;
4888             }
4889             else if (elemType === 'separator') {
4890                 innerHTML = `
4891                     <div class="element-header" style="background:$
     {config.color};">Сепаратор</div>
4892                     <div class="element-body">
4893                         <div class="ports-left">${buildInputPorts(1,
     config.inputTypes, config.inputLabels)}</div>
4894                         <div class="element-symbol">✓/✗</div>
4895                         <div class="ports-right">
4896                             <div class="port output logic-port true-port" data-
     port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
     title="ИСТИНА"></div>
4897                             <div class="port output logic-port false-port" data-
     port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
     title="ЛОЖЬ"></div>
4898                         </div>
4899                     </div>`;
4900             }
4901             else if (elemType === 'and' || elemType === 'or') {
4902                 const gateSymbol = elemType === 'and' ? '∧' : '∨';
4903                 const inputCount = props.inputCount || config.defaultProps?.inputCount
     || 2;
4904 
4905                 // Генерируем динамические входы
4906                 let inputsHTML = '';
4907                 for (let i = 0; i < inputCount; i++) {
4908                     inputsHTML += `<div class="port input logic-port" data-port="in-$
     {i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход $
     {i+1}"></div>`;
4909                 }
4910 
4911                 innerHTML = `
4912                     <div class="element-header" style="background:${config.color};">$
     {config.name}</div>
4913                     <div class="element-body">
4914                         <div class="ports-left">
4915                             ${inputsHTML}
4916                         </div>
4917                         <div class="element-symbol">${gateSymbol}</div>
4918                         <div class="ports-right">
4919                             <div class="port output logic-port" data-port="out-0"
     data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></
     div>
4920                         </div>
4921                     </div>`;
4922             }
4923             else if (elemType === 'if') {
4924                 const op = safe(props.operator, '=');
4925                 innerHTML = `
4926                     <div class="element-header" style="background:$
     {config.color};">Условие</div>
4927                     <div class="element-body">
4928                         <div class="ports-left">${buildInputPorts(2,
     config.inputTypes, config.inputLabels)}</div>
4929                         <div class="element-symbol">${op}</div>
```

```
                        <div class="ports-right">
                            ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'not') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">НЕ</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(1, [SIGNAL_TYPE.LOGIC], ['A'])}</div>
                        <div class="element-symbol">¬</div>
                        <div class="ports-right">
                            ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'formula') {
                const inputCount = props.inputCount || config.defaultProps?.inputCount || config.inputs || 2;
                const expression = safe(props.expression);
                const displayExpression = expression
                    ? (expression.length > 12 ? `${expression.slice(0, 12)}…` : expression)
                    : 'f(x)';

                innerHTML = `
                    ${buildConditionPortHTML()}
                    <div class="element-header" style="background:${config.color};">Формула</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(inputCount, config.inputTypes, config.inputLabels)}</div>
                        <div class="element-symbol">${displayExpression}</div>
                        <div class="ports-right">
                            ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Результат'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'output') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Выход</div>
                    <div class="element-body">
                        <div class="ports-left">
                            ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
                        </div>
                        <div class="element-symbol">${safe(props.label, 'Выход')}</div>
                        <div class="ports-right"></div>
                    </div>`;

            }
            else if (elemType === 'group') {
                const title = props.title || 'Группа';
                innerHTML = `
                    <div class="group-content">
                    <div class="group-title">${title}</div>
                    </div>`;
            }

            else { // Для любых других (fallback)
                innerHTML = `
```

```
                    <div class="element-header" style="background:${config.color};">${config.name}</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(config.inputs || 0, config.inputTypes, config.inputLabels)}</div>
                        <div class="element-symbol">${config.name}</div>
                        <div class="ports-right">
                            ${buildOutputPorts(config.outputs || 0, config.outputTypes, config.outputLabels)}
                        </div>
                    </div>`;
            }
            const commentHtml = `<div class="element-comment">${safe(props.comment, '')}</div>`;


            const html = `
                <div class="element ${elemType}" id="${elemId}"
                    style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;" data-type="${elemType}">
                    ${innerHTML}
                    ${commentHtml}
                    ${resizeHandles}
                </div>`;

            return { html, width: w, height: h };
        },

    /**
     * Добавление элемента
     */
        addElement(elemType, x, y, props = {}, elemId = null, customWidth = null, customHeight = null) {
            const config = ELEMENT_TYPES[elemType];
            if (!config) {
                console.error(`Неизвестный тип элемента: ${elemType}`);
                return null;
            }

            if (!elemId) {
                elemId = `${elemType}_${++AppState.elementCounter}`;
            }

            let width = customWidth;
            let height = customHeight;

            if (width === null || width === undefined) {
                width = config.minWidth || 140;
            }
            if (height === null || height === undefined) {
                height = config.minHeight || 70;
            }

            try {
                const result = this.createElementHTML(elemType, elemId, x, y, props, width, height);
                if (!result || !result.html) {
                    console.error('createElementHTML вернул пустой результат');
                    return null;
                }

                const workspace = document.getElementById('workspace');
                const wrapper = document.createElement('div');
                wrapper.innerHTML = result.html.trim();
                const element = wrapper.firstElementChild;
```

```
5044                 if (!element) {
5045                     console.error('Не удалось создать DOM элемент из HTML');
5046                     return null;
5047                 }
5048 
5049                 // Добавляем класс для отступа
5050                 if (config.hasConditionPort) {
5051                     element.classList.add('has-condition-port');
5052                 }
5053 
5054                 workspace.appendChild(element);
5055 
5056                 AppState.elements[elemId] = {
5057                     id: elemId,
5058                     type: elemType,
5059                     x,
5060                     y,
5061                     width: result.width || width,
5062                     height: result.height || height,
5063                     props: { ...(config.defaultProps || {}), ...(props || {}) }
5064                 };
5065 
5066                 // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ ЕГО В HTML)
5067                 if (config.hasConditionPort && elemType !== 'formula') {
5068                     const condPortWrapper = document.createElement('div');
5069                     condPortWrapper.innerHTML = `
5070                         <div class="condition-port-wrapper">
5071                             <div class="condition-port-label">условие</div>
5072                             <div class="port input condition-port"
5073                                 data-port="cond-0"
5074                                 data-element="${elemId}"
5075                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"
5076                                 title="Техническое условие">
5077                             </div>
5078                         </div>`;
5079                     element.prepend(condPortWrapper.firstElementChild); // Вставляем в самое начало элемента
5080                 }
5081 
5082 
5083                 this.setupElementHandlers(elemId); // Передаем ID элемента
5084 
5085                 // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
5086                 // element.querySelectorAll('.port').forEach(port => {
5087                 //     Connections.setupPortHandlers(port);
5088                 // });
5089 
5090                 Connections.drawConnections(); // Перерисовываем соединения, чтобы учесть новые порты
5091                 Viewport.updateMinimap();
5092                 return elemId;
5093             } catch (err) {
5094                 console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
5095                 return null;
5096             }
5097         },
5098 
5099     /**
5100      * Обновление входов логического элемента (AND, OR)
5101      */
5102     updateLogicGateInputs(elemId, inputCount) {
5103         const elem = document.getElementById(elemId);
5104         if (!elem) return;
5105 
```

```
        const portsLeft = elem.querySelector('.ports-left');
        if (!portsLeft) return;

        // Удаляем соединения к портам, которые больше не существуют
        AppState.connections = AppState.connections.filter(c => {
            if (c.toElement === elemId && c.toPort.startsWith('in-')) {
                const portNum = parseInt(c.toPort.split('-')[1], 10);
                return portNum < inputCount;
            }
            return true;
        });

        // Генерируем новые входы
        let inputsHTML = '';
        for (let i = 0; i < inputCount; i++) {
            inputsHTML += `
                <div class="port input logic-port"
                    data-port="in-${i}"
                    data-element="${elemId}"
                    data-signal-type="${SIGNAL_TYPE.LOGIC}"
                    title="Вход ${i+1}">
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

        // Переподключаем обработчики
        portsLeft.querySelectorAll('.port').forEach(port =>
            Connections.setupPortHandlers(port)
        );

        Connections.drawConnections();
    },

    /**
     * Удаление элемента
     */
    deleteElement(elemId) {
        AppState.connections = AppState.connections.filter(c =>
            c.fromElement !== elemId && c.toElement !== elemId
        );

        const elem = document.getElementById(elemId);
        if (elem) elem.remove();

        delete AppState.elements[elemId];

        if (AppState.selectedElement === elemId) {
            AppState.selectedElement = null;
        }

        Connections.drawConnections();
        Viewport.updateMinimap();
    },

    /**
     * Выделение элемента
     */
// elements.js
    selectElement(elemId) {
        // Снимаем старое выделение со всех
        this.deselectAll();

        AppState.selectedElement = elemId;
        AppState.selectedElements = [elemId];
```

```
5171
5172          const elem = document.getElementById(elemId);
5173          if (elem) elem.classList.add('selected');
5174
5175          const elemData = AppState.elements[elemId];
5176          if (elemData) {
5177              document.getElementById('selection-info').textContent =
5178                  `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
5179          }
5180      },
5181
5182      deselectAll() {
5183          // Снимаем класс со всех элементов на странице
5184          document.querySelectorAll('.element.selected').forEach(el =>
      el.classList.remove('selected'));
5185
5186          AppState.selectedElement = null;
5187          AppState.selectedElements = [];
5188          if (document.getElementById('selection-info')) {
5189              document.getElementById('selection-info').textContent = '';
5190          }
5191      },
5192      /**
5193       * Настройка обработчиков элемента
5194       */
5195      setupElementHandlers(elemId) {
5196          try {
5197              const elem = document.getElementById(elemId);
5198              if (!elem) return;
5199
5200              // elements.js -> setupElementHandlers
5201              elem.addEventListener('mousedown', (e) => {
5202                  if (e.target.classList.contains('port')) return;
5203                  if (e.target.classList.contains('resize-handle')) return;
5204
5205                  e.preventDefault();
5206                  e.stopPropagation();
5207
5208                  // ПРАВКА ТУТ:
5209                  // Если элемент НЕ в группе — выделяем только его.
5210                  // Если элемент УЖЕ в группе — не трогаем группу, чтобы можно было
      тянуть всех.
5211                  if (!AppState.selectedElements.includes(elemId)) {
5212                      this.selectElement(elemId);
5213                  }
5214
5215                  AppState.draggingElement = elemId;
5216                  const canvasPos = screenToCanvas(e.clientX, e.clientY);
5217                  const elemData = AppState.elements[elemId];
5218                  AppState.dragOffset.x = canvasPos.x - elemData.x;
5219                  AppState.dragOffset.y = canvasPos.y - elemData.y;
5220              });
5221
5222              elem.addEventListener('dblclick', (e) => {
5223                  if (e.target.classList.contains('port')) return;
5224                  const config = ELEMENT_TYPES[AppState.elements[elemId].type];
5225                  if (config?.hasProperties) {
5226                      Modal.showPropertiesModal(elemId);
5227                  }
5228              });
5229
5230              elem.addEventListener('contextmenu', (e) => {
5231                  e.preventDefault();
5232                  this.showContextMenu(e.clientX, e.clientY, elemId);
5233              });
```

```
5234
5235                const handles = elem.querySelectorAll('.resize-handle');
5236                handles.forEach(handle => this.setupResizeHandlers(handle, elemId));
5237
5238                const ports = elem.querySelectorAll('.port');
5239                ports.forEach(port => Connections.setupPortHandlers(port));
5240
5241            } catch (err) {
5242                console.error('setupElementHandlers error for', elemId, err);
5243            }
5244        },
5245
5246        /**
5247         * Контекстное меню
5248         */
5249        showContextMenu(x, y, elemId) {
5250            const menu = document.getElementById('context-menu');
5251            menu.style.left = `${x}px`;
5252            menu.style.top = `${y}px`;
5253            menu.style.display = 'block';
5254            menu.dataset.elementId = elemId;
5255        },
5256
5257        /**
5258         * Настройка resize
5259         */
5260        setupResizeHandlers(handle, elemId) {
5261            handle.addEventListener('mousedown', (e) => {
5262                e.stopPropagation();
5263                e.preventDefault();
5264
5265                const elemData = AppState.elements[elemId];
5266
5267                AppState.resizing = {
5268                    elemId: elemId,
5269                    handle: handle.dataset.direction,
5270                    startX: e.clientX,
5271                    startY: e.clientY,
5272                    startWidth: elemData.width,
5273                    startHeight: elemData.height,
5274                    startLeft: elemData.x,
5275                    startTop: elemData.y
5276                };
5277            });
5278        },
5279    // elements.js — ЗАМЕНИ функцию copySelectedElements
5280        copySelectedElements() {
5281            const ids = (AppState.selectedElements && AppState.selectedElements.length >
      0)
5282                ? [...AppState.selectedElements]
5283                : (AppState.selectedElement ? [AppState.selectedElement] : []);
5284
5285            if (ids.length === 0) {
5286                console.log('Нечего копировать');
5287                return;
5288            }
5289
5290            const originals = ids
5291                .map(id => AppState.elements[id])
5292                .filter(Boolean);
5293
5294            if (originals.length === 0) return;
5295
5296            const offsetX = 50;
5297            const offsetY = 50;
```

```
5298
5299          const idMap = {};
5300          const newIds = [];
5301
5302          originals.forEach(el => {
5303              // Копируем свойства элемента (глубокое копирование props)
5304              const newProps = JSON.parse(JSON.stringify(el.props || {}));
5305
5306              // Используем существующую функцию addElement
5307              // Она сама сгенерирует ID и создаст DOM
5308              const createdId = this.addElement(
5309                  el.type,                    // тип элемента
5310                  el.x + offsetX,             // новая позиция X
5311                  el.y + offsetY,             // новая позиция Y
5312                  newProps,                   // скопированные свойства
5313                  null,                       // ID = null, чтобы addElement
      сгенерировал сам
5314                  el.width,                   // ширина
5315                  el.height                   // высота
5316              );
5317
5318              if (createdId) {
5319                  idMap[el.id] = createdId;
5320                  newIds.push(createdId);
5321              }
5322          });
5323
5324          // Копируем связи ТОЛЬКО между скопированными элементами
5325          const newConnections = [];
5326          AppState.connections.forEach(conn => {
5327              if (idMap[conn.fromElement] && idMap[conn.toElement]) {
5328                  newConnections.push({
5329                      fromElement: idMap[conn.fromElement],
5330                      fromPort: conn.fromPort,
5331                      toElement: idMap[conn.toElement],
5332                      toPort: conn.toPort,
5333                      signalType: conn.signalType || 'Boolean'
5334                  });
5335              }
5336          });
5337
5338          AppState.connections.push(...newConnections);
5339          Connections.drawConnections();
5340
5341          // Выделяем новые элементы
5342          this.deselectAll();
5343          AppState.selectedElements = newIds;
5344          AppState.selectedElement = newIds[newIds.length - 1];
5345
5346          newIds.forEach(id => {
5347              const el = document.getElementById(id);
5348              if (el) el.classList.add('selected');
5349          });
5350
5351          document.getElementById('selection-info').textContent =
5352              `Скопировано: ${newIds.length} элемент(ов)`;
5353
5354          Viewport.updateMinimap();
5355          console.log(`Скопировано ${newIds.length} элементов`);
5356      },
5357
5358          // elements.js — добавь в объект Elements
5359      deleteSelectedElements() {
5360          const ids = (AppState.selectedElements && AppState.selectedElements.length >
      0)
```

```
5361              ? [...AppState.selectedElements]
5362              : (AppState.selectedElement ? [AppState.selectedElement] : []);
5363
5364          if (ids.length === 0) {
5365              console.log('Нечего удалять');
5366              return;
5367          }
5368
5369          // Удаляем каждый элемент
5370          ids.forEach(id => {
5371              this.deleteElement(id);
5372          });
5373
5374          // Сбрасываем выделение
5375          AppState.selectedElement = null;
5376          AppState.selectedElements = [];
5377          document.getElementById('selection-info').textContent = '';
5378
5379          console.log(`Удалено ${ids.length} элементов`);
5380      },
5381
5382      /**
5383       * Обработка resize
5384       */
5385      handleResize(e) {
5386          if (!AppState.resizing) return;
5387
5388          const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
      startTop } = AppState.resizing;
5389          const elem = document.getElementById(elemId);
5390          const elemData = AppState.elements[elemId];
5391          const config = ELEMENT_TYPES[elemData.type];
5392
5393          const dx = (e.clientX - startX) / AppState.viewport.zoom;
5394          const dy = (e.clientY - startY) / AppState.viewport.zoom;
5395
5396          let newWidth = startWidth;
5397          let newHeight = startHeight;
5398          let newLeft = startLeft;
5399          let newTop = startTop;
5400
5401          if (handle.includes('e')) {
5402              newWidth = Math.max(config.minWidth, startWidth + dx);
5403          }
5404          if (handle.includes('w')) {
5405              newWidth = Math.max(config.minWidth, startWidth - dx);
5406              newLeft = startLeft + (startWidth - newWidth);
5407          }
5408          if (handle.includes('s')) {
5409              newHeight = Math.max(config.minHeight, startHeight + dy);
5410          }
5411          if (handle.includes('n')) {
5412              newHeight = Math.max(config.minHeight, startHeight - dy);
5413              newTop = startTop + (startHeight - newHeight);
5414          }
5415
5416          elem.style.width = `${newWidth}px`;
5417          elem.style.height = `${newHeight}px`;
5418          elem.style.left = `${newLeft}px`;
5419          elem.style.top = `${newTop}px`;
5420
5421          elemData.width = newWidth;
5422          elemData.height = newHeight;
5423          elemData.x = newLeft;
5424          elemData.y = newTop;
```

```
5425
5426          Connections.drawConnections();
5427      },
5428
5429      /**
5430       * Обработка перетаскивания элемента
5431       */
5432      handleDrag(e) {
5433          if (!AppState.draggingElement) return;
5434
5435          const canvasPos = screenToCanvas(e.clientX, e.clientY);
5436          const elemId = AppState.draggingElement;
5437          const elemData = AppState.elements[elemId];
5438          if (!elemData) return;
5439
5440          const newX = canvasPos.x - AppState.dragOffset.x;
5441          const newY = canvasPos.y - AppState.dragOffset.y;
5442          const dx = newX - elemData.x;
5443          const dy = newY - elemData.y;
5444
5445          // если выделено несколько
5446          const group = AppState.selectedElements && AppState.selectedElements.length >
     1
5447              ? AppState.selectedElements
5448              : [elemId];
5449
5450          for (const id of group) {
5451              const elData = AppState.elements[id];
5452              if (!elData) continue;
5453              elData.x += dx;
5454              elData.y += dy;
5455              const el = document.getElementById(id);
5456              if (el) {
5457              el.style.left = elData.x + 'px';
5458              el.style.top = elData.y + 'px';
5459              }
5460          }
5461
5462          Connections.drawConnections();
5463      },
5464
5465      /**
5466       * Обновление входов формулы
5467       */
5468      updateFormulaInputs(elemId, inputCount) {
5469          const elem = document.getElementById(elemId);
5470          if (!elem) return;
5471
5472          const portsLeft = elem.querySelector('.ports-left');
5473          if (!portsLeft) return;
5474
5475          AppState.connections = AppState.connections.filter(c => {
5476              if (c.toElement === elemId && c.toPort.startsWith('in-')) {
5477                  const portNum = parseInt(c.toPort.split('-')[1], 10);
5478                  return portNum < inputCount;
5479              }
5480              return true;
5481          });
5482
5483          let inputsHTML = '';
5484          for (let i = 0; i < inputCount; i++) {
5485              inputsHTML += `
5486                  <div class="port input any-port"
5487                       data-port="in-${i}"
5488                       data-element="${elemId}"
```

```
                    data-signal-type="${SIGNAL_TYPE.ANY}"
                    title="in${i} (Любой)">
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

        portsLeft.querySelectorAll('.port').forEach(port =>
            Connections.setupPortHandlers(port)
        );

        Connections.drawConnections();
    },

    /**
     * Рассчитать оптимальный размер элемента на основе количества портов
     */
    calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
        const elem = AppState.elements[elemId];
        if (!elem) return null;

        const config = ELEMENT_TYPES[elem.type];
        if (!config || !config.resizable) return null;

        // Базовая высота
        let baseHeight = config.minHeight || 60;

        // Каждый порт требует примерно 25-30px высоты
        const portSpacing = 28;
        const maxPorts = Math.max(inputCount, outputCount);

        // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
        const additionalHeight = (maxPorts - 1) * portSpacing;
        const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);

        return newHeight;
    },

    /**
     * Обновление размера элемента при изменении портов
     */
    updateElementSize(elemId) {
        const elem = document.getElementById(elemId);
        const elemData = AppState.elements[elemId];

        if (!elem || !elemData) return;

        const config = ELEMENT_TYPES[elemData.type];
        if (!config || !config.resizable) return;

        let inputCount = 0;
        let outputCount = config.outputs || 1;

        // Определяем количество входов
        if (elemData.type === 'and' || elemData.type === 'or' || elemData.type === 'formula') {
            inputCount = elemData.props.inputCount || config.inputs || 2;
        } else {
            inputCount = config.inputs || 0;
        }

        // Рассчитываем новую высоту
        const newHeight = this.calculateOptimalHeight(elemId, inputCount, outputCount);

```

```
5552          if (newHeight && newHeight !== elemData.height) {
5553              elemData.height = newHeight;
5554              elem.style.height = `${newHeight}px`;
5555
5556              // Перерисовываем соединения, т.к. изменился размер элемента
5557              Connections.drawConnections();
5558              Viewport.updateMinimap();
5559          }
5560      }
5561
5562
5563  };
5564
5565  /**
5566   * Модуль модальных окон
5567   * modal.js
5568   */
5569
5570  const Modal = {
5571      /**
5572       * Инициализация модальных окон
5573       */
5574      init() {
5575          // Модальное окно свойств элемента
5576          document.getElementById('modal-save').addEventListener('click', () => {
5577              this.saveElementProperties();
5578          });
5579
5580          document.getElementById('modal-cancel').addEventListener('click', () => {
5581              this.hideModal('modal-overlay');
5582          });
5583
5584          document.getElementById('modal-overlay').addEventListener('click', (e) => {
5585              if (e.target.id === 'modal-overlay') {
5586                  this.hideModal('modal-overlay');
5587              }
5588          });
5589
5590          // Модальное окно свойств проекта
5591          document.getElementById('project-modal-save').addEventListener('click', () =>
      {
5592              this.saveProjectProperties();
5593          });
5594
5595          document.getElementById('project-modal-cancel').addEventListener('click', ()
      => {
5596              this.hideModal('project-modal-overlay');
5597          });
5598
5599          document.getElementById('project-modal-overlay').addEventListener('click', (e)
      => {
5600              if (e.target.id === 'project-modal-overlay') {
5601                  this.hideModal('project-modal-overlay');
5602              }
5603          });
5604      },
5605
5606      /**
5607       * Показать модальное окно
5608       */
5609      showModal(modalId) {
5610          document.getElementById(modalId).style.display = 'flex';
5611      },
5612
5613      /**
```

```
5614        * Скрыть модальное окно
5615        */
5616       hideModal(modalId) {
5617           document.getElementById(modalId).style.display = 'none';
5618           // Скрываем tooltip если он есть
5619           const tooltip = document.getElementById('template-tooltip');
5620           if (tooltip) {
5621               tooltip.classList.remove('visible');
5622           }
5623       },
5624
5625       /**
5626        * Показать свойства элемента
5627        */
5628       showPropertiesModal(elemId) {
5629           const elemData = AppState.elements[elemId];
5630           const elemType = elemData.type;
5631           const props = elemData.props;
5632           const config = ELEMENT_TYPES[elemType];
5633
5634           const modalOverlay = document.getElementById('modal-overlay');
5635           const modalTitle = document.getElementById('modal-title');
5636           const modalContent = document.getElementById('modal-content');
5637
5638           modalTitle.textContent = `Свойства: ${config.name}`;
5639
5640           let contentHTML = '';
5641
5642    if (elemType === 'input-signal') {
5643     const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
5644
5645     contentHTML = `
5646       <div class="modal-row">
5647         <label>Название сигнала:</label>
5648         <input type="text" id="prop-name" value="${props.name || ''}"
      placeholder="Например: 10LBA..." />
5649         <small style="color:#999;">
5650           Поиск по маске через * (например: *MAA*CP*)
5651         </small>
5652         <div id="signal-filter-results"
5653              style="max-height:160px; overflow-y:auto; background:#0f3460; border-
      radius:5px; margin-top:6px; display:none;">
5654         </div>
5655       </div>
5656
5657       <div class="modal-row">
5658         <label>Описание сигнала:</label>
5659         <textarea id="prop-description" readonly>${props.description || ''}</textarea>
5660       </div>
5661
5662       // modal.js в блоке input-signal
5663       <div class="modal-row">
5664           <label>Размерность:</label>
5665           <input type="text" id="prop-dimension" value="${props.dimension || ''}" />
5666       </div>
5667
5668       <div class="modal-row">
5669         <label>Тип сигнала:</label>
5670         <select id="prop-signal-type">
5671           <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
      'selected' : ''}>Числовой</option>
5672           <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
      'selected' : ''}>Логический</option>
5673         </select>
5674       </div>
```

```
5675    `;
5676
5677    // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
      DOM.
5678    // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
5679    // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
5680  } else if (elemType === 'if') {
5681              contentHTML = `
5682                  <div class="modal-row">
5683                      <label>Оператор сравнения:</label>
5684                      <select id="prop-operator">
5685                          <option value="=" ${props.operator === '=' ? 'selected' : ''}
      >=  (равно)</option>
5686                          <option value=">" ${props.operator === '>' ? 'selected' : ''}
      >>  (больше)</option>
5687                          <option value="<" ${props.operator === '<' ? 'selected' : ''}
      ><  (меньше)</option>
5688                          <option value=">=" ${props.operator === '>=' ? 'selected' :
      ''}>= (больше или равно)</option>
5689                          <option value="<=" ${props.operator === '<=' ? 'selected' :
      ''}>= (меньше или равно)</option>
5690                          <option value="!=" ${props.operator === '!=' ? 'selected' :
      ''}>!= (не равно)</option>
5691                      </select>
5692                  </div>
5693              `;
5694          } else if (elemType === 'and' || elemType === 'or') {
5695              contentHTML = `
5696                  <div class="modal-row">
5697                      <label>Количество входов:</label>
5698                      <input type="number" id="prop-input-count" value="$
      {props.inputCount || 2}" min="2" max="10">
5699                  </div>
5700                  <div class="modal-row">
5701                      <p style="color: #aaa; font-size: 12px;">
5702                          Измените количество входных портов для этого логического
      элемента.
5703                          Лишние соединения будут автоматически удалены.
5704                      </p>
5705                  </div>
5706              `;
5707          } else if (elemType === 'const') {
5708              contentHTML = `
5709                  <div class="modal-row">
5710                      <label>Значение:</label>
5711                      <input type="number" id="prop-value" value="${props.value ?? 0}"
      step="any">
5712                  </div>
5713              `;
5714          }
5715          else if (elemType === 'group') {
5716              contentHTML = `
5717              <div class="modal-row">
5718                  <label>Название группы:</label>
5719                  <input type="text" id="prop-title" value="${props.title || 'Группа'}">
5720              </div>`;
5721          }
5722
5723          else if (elemType === 'formula') {
5724              let signalsHTML = '';
5725              AppState.connections.forEach(conn => {
5726                  if (conn.toElement === elemId) {
5727                      const fromElem = AppState.elements[conn.fromElement];
5728                      if (fromElem) {
5729                          const signalName = fromElem.props?.name || fromElem.id;
```

```
5730                         signalsHTML += `<div class="signal-item" data-signal="$
    {signalName}">${signalName} (${conn.toPort})</div>`;
5731                     }
5732                 }
5733             });
5734 
5735             // ... (где-то выше код сбора signalsHTML) ...
5736 
5737             contentHTML = `
5738                 <div class="modal-row">
5739                     <label>Количество входов:</label>
5740                     <input type="number" id="prop-input-count" value="$
    {props.inputCount || 2}" min="1" max="10">
5741                 </div>
5742 
5743                 <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->
5744                 <div style="display: flex; gap: 15px; margin-bottom: 15px; height:
    140px;">
5745                     <!-- Левая колонка: Сигналы -->
5746                     <div style="flex: 1; display: flex; flex-direction: column;">
5747                         <label style="margin-bottom: 5px; display:block;">Входные
    сигналы:</label>
5748                         <div class="signal-list" id="signal-list" style="flex: 1;
    overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px
    solid #4a90d9;">
5749                             ${signalsHTML || '<div style="color:#888;padding:5px;">Нет
    сигналов</div>'}
5750                         </div>
5751                     </div>
5752 
5753                     <!-- Правая колонка: Шаблоны -->
5754                     <div style="flex: 1; display: flex; flex-direction: column;">
5755                         <label style="margin-bottom: 5px; display:block;">Шаблоны:</
    label>
5756                         <div class="signal-list" id="template-list" style="flex: 1;
    overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px
    solid #4a90d9;">
5757                             <div style="color:#888;padding:5px;">Загрузка…</div>
5758                         </div>
5759                     </div>
5760                 </div>
5761 
5762                 <!-- Нижний блок: Поле формулы (во всю ширину) -->
5763                 <div class="modal-row">
5764                     <label>Выражение формулы:</label>
5765                     <textarea id="prop-expression"
5766                               style="width: 100%; min-height: 80px; font-family:
    monospace; font-size: 14px; line-height: 1.4;"
5767                               spellcheck="false">${props.expression || ''}</textarea>
5768                     <small style="color:#999; display:block; margin-top:4px;">
5769                         Двойной клик по сигналу или шаблону вставит его в позицию
    курсора (или заменит выделенный текст).
5770                     </small>
5771                 </div>
5772             `;
5773         }
5774         if (!contentHTML) {
5775             contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных
    свойств.</div>`;
5776             }
5777         contentHTML += `
5778             <div class="modal-row">
5779                 <label>Комментарий:</label>
5780                 <textarea id="prop-comment" placeholder="Комментарий к элементу...">$
    {props.comment || ''}</textarea>
```

```
5781             </div>
5782             `;
5783 
5784 
5785         modalContent.innerHTML = contentHTML;
5786         // modal.js — внутри showPropertiesModal, блок if (elemType === 'formula')
5787         if (elemType === 'formula') {
5788             const listEl = document.getElementById('template-list');
5789 
5790             // Создаём tooltip элемент (один на всю страницу)
5791             let tooltip = document.getElementById('template-tooltip');
5792             if (!tooltip) {
5793                 tooltip = document.createElement('div');
5794                 tooltip.id = 'template-tooltip';
5795                 tooltip.className = 'template-tooltip';
5796                 document.body.appendChild(tooltip);
5797             }
5798 
5799             (async () => {
5800                 try {
5801                     const data = await Settings.fetchFormulaTemplates();
5802                     const items = data.templates || [];
5803 
5804                     if (!items.length) {
5805                         listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет
      шаблонов</div>';
5806                         return;
5807                     }
5808 
5809                     // Новый код
5810                     listEl.innerHTML = items.map(t => {
5811                         // --- НАЧАЛО ИЗМЕНЕНИЙ ---
5812                         let argList = [];
5813 
5814                         if (Array.isArray(t.args)) {
5815                             // Если пришел старый формат (массив): ["p", "t"]
5816                             argList = t.args;
5817                         } else if (t.args && typeof t.args === 'object') {
5818                             // Если пришел новый формат (объект): {"p": {...}, "t":
      {...}}
5819                             // Берем только ключи (имена переменных)
5820                             argList = Object.keys(t.args);
5821                         }
5822 
5823                         // Формируем подпись функции: h(p, t)
5824                         const sig = `${t.name}(${argList.join(', ')})`;
5825                         // --- КОНЕЦ ИЗМЕНЕНИЙ ---
5826 
5827                         const desc = (t.description || '').replace(/"/g, '&quot;');
5828                         return `<div class="signal-item template-item"
5829                                 data-insert="${sig}"
5830                                 data-name="${t.name}"
5831                                 data-description="${desc}">${sig}</div>`;
5832                     }).join('');
5833 
5834                     // Обработчики для каждого шаблона
5835                     listEl.querySelectorAll('.template-item').forEach(div => {
5836                         // Двойной клик — вставка
5837                         div.addEventListener('dblclick', () => {
5838                             const insert = div.dataset.insert;
5839                             const textarea = document.getElementById('prop-
      expression');
5840                             insertAtCursor(textarea, insert);
5841                         });
5842 
```

```
                        // Наведение — показать tooltip
                        div.addEventListener('mouseenter', (e) => {
                            const description = div.dataset.description;
                            const name = div.dataset.name;

                            if (!description) return;

                            tooltip.innerHTML = `
                                <div class="template-tooltip-title">${name}</div>
                                <div>${description}</div>
                            `;

                            // Позиционируем tooltip
                            const rect = div.getBoundingClientRect();
                            tooltip.style.left = rect.left + 'px';
                            tooltip.style.top = (rect.bottom + 8) + 'px';
                            tooltip.classList.add('visible');
                        });

                        // Уход мыши — скрыть tooltip
                        div.addEventListener('mouseleave', () => {
                            tooltip.classList.remove('visible');
                        });
                    });

                } catch (e) {
                    console.error(e);
                    listEl.innerHTML = '<div style="color:#888;padding:5px;">Ошибка загрузки</div>';
                }
            })();
        }



        // --- post init handlers (когда DOM модалки уже существует) ---
        if (elemType === 'input-signal') {
            const input = document.getElementById('prop-name');
            const results = document.getElementById('signal-filter-results');
            const descField = document.getElementById('prop-description');

            let timer = null;

            const renderList = (items) => {
                if (!items || items.length === 0) {
                results.innerHTML = '<div style="color:#666;padding:6px;">Нет совпадений</div>';
                results.style.display = 'block';
                return;
                }

                results.innerHTML = items.map(s => `
                <div class="signal-result-item"
                    style="padding:6px 8px; cursor:pointer; border-bottom:1px solid rgba(255,255,255,0.08);">
                    <div style="font-weight:600;">${s.Tagname}</div>
                    <div style="color:#aaa; font-size:11px;">${s.Description || ''}</div>
                </div>
                `).join('');

                results.style.display = 'block';

                results.querySelectorAll('.signal-result-item').forEach((div, i) => {
                div.addEventListener('click', () => {
```

```
                    const chosen = items[i];
                    input.value = chosen.Tagname;
                    descField.value = chosen.Description || '';
                    const dimField = document.getElementById('prop-dimension');
                    if (dimField) dimField.value = chosen.EngineeringUnit ||
    chosen.Dimension || '';
                    results.style.display = 'none';
                });
                });
            };

            const search = async () => {
                const mask = (input.value || '').trim();

                // Показываем список только если пользователь реально использует маску
                if (!mask.includes('*')) {
                results.style.display = 'none';
                return;
                }

                results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</
    div>';
                results.style.display = 'block';

                try {
                // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
                const data = await Settings.fetchSignals(mask, 50);
                renderList(data.items || []);
                } catch (e) {
                results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка
    загрузки сигналов</div>';
                results.style.display = 'block';
                console.error(e);
                }
            };

            input.addEventListener('input', () => {
                clearTimeout(timer);
                timer = setTimeout(search, 200); // debounce
            });

            // опционально: закрывать список кликом вне
            document.addEventListener('mousedown', (e) => {
                if (!results.contains(e.target) && e.target !== input) {
                results.style.display = 'none';
                }
            }, { once: true });
        }
        modalOverlay.dataset.elementId = elemId;
        this.showModal('modal-overlay');

        // Функция для умной вставки текста в позицию курсора
        const insertAtCursor = (field, text) => {
            if (!field) return;

            // Получаем позиции выделения
            const startPos = field.selectionStart;
            const endPos = field.selectionEnd;
            const currentValue = field.value;

            // Вставляем текст: (текст до) + (новый текст) + (текст после)
            field.value = currentValue.substring(0, startPos) +
                        text +
                        currentValue.substring(endPos, currentValue.length);

```

```
            // Возвращаем фокус и ставим курсор сразу после вставленного текста
            field.focus();
            const newCursorPos = startPos + text.length;
            field.setSelectionRange(newCursorPos, newCursorPos);
        };

        // Обработчик вставки сигналов для формулы
        if (elemType === 'formula') {
            document.querySelectorAll('.signal-item').forEach(item => {
            item.addEventListener('dblclick', () => {
                const signal = item.dataset.signal;
                const textarea = document.getElementById('prop-expression');

                // БЫЛО: textarea.value += signal;
                // СТАЛО:
                insertAtCursor(textarea, signal);
                });
            });
        }
    },

    /**
     * Сохранить свойства элемента
     */
/**
 * Сохранить свойства элемента
 */
    saveElementProperties() {
        try {
            const modalOverlay = document.getElementById('modal-overlay');
            const elemId = modalOverlay.dataset.elementId;
            const elemData = AppState.elements[elemId];
            const elem = document.getElementById(elemId);
            if (!elemData) {
                alert('⚠ Элемент не найден — возможно, он был удалён или
переименован.');
                console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
                this.hideModal('modal-overlay');
                return;
            }

            const elemType = elemData.type;

            if (elemType === 'input-signal') {
                const name = document.getElementById('prop-name').value || 'Сигнал';
                const description = document.getElementById('prop-description').value
|| '';
                const signalType = document.getElementById('prop-signal-type').value;
                const dimension = document.getElementById('prop-dimension').value ||
'';
                elemData.props.dimension = dimension;

                const oldSignalType = elemData.props.signalType;
                elemData.props.name = name;
                elemData.props.description = description;
                elemData.props.signalType = signalType;

                if (oldSignalType !== signalType) {
                    AppState.connections = AppState.connections.filter(conn => {
                    if (conn.fromElement === elemId) {
                        const toPortIndex = parseInt(conn.toPort.split('-')[1]);
                        const inputType = getInputPortType(conn.toElement,
toPortIndex);
                        return areTypesCompatible(signalType, inputType);
                    }
```

```
6027                     return true;
6028                     });
6029                 }
6030 
6031                 const { html } = Elements.createElementHTML(
6032                     elemType, elemId, elemData.x, elemData.y, elemData.props,
      elemData.width, elemData.height
6033                 );
6034                 elem.outerHTML = html;
6035 
6036                 Elements.setupElementHandlers(elemId);
6037                 Connections.drawConnections();
6038             } else if (elemType === 'if') {
6039                 const operator = document.getElementById('prop-operator').value;
6040                 elemData.props.operator = operator;
6041                 const symbol = elem.querySelector('.element-symbol');
6042                 if (symbol) symbol.textContent = operator;
6043 
6044             } else if (elemType === 'const') {
6045                 const value = parseFloat(document.getElementById('prop-value').value)
      || 0;
6046                 elemData.props.value = value;
6047                 const symbol = elem.querySelector('.element-symbol');
6048                 if (symbol) symbol.textContent = String(value);
6049 
6050             } else if (elemType === 'formula') {
6051                 const expression = document.getElementById('prop-expression').value;
6052                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
6053 
6054                 elemData.props.expression = expression;
6055                 elemData.props.inputCount = inputCount;
6056 
6057                 const symbol = elem.querySelector('.element-symbol');
6058                 if (symbol) {
6059                     symbol.textContent = expression.length > 12 ? `$
      {expression.slice(0, 12)}…` : (expression || 'f(x)');
6060                 }
6061 
6062                 Elements.updateFormulaInputs(elemId, inputCount);
6063                 Elements.updateElementSize(elemId);  // ← Добавляем это
6064             } else if (elemType === 'and' || elemType === 'or') {
6065                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
6066                 elemData.props.inputCount = inputCount;
6067 
6068                 Elements.updateLogicGateInputs(elemId, inputCount);
6069                 Elements.updateElementSize(elemId);  // ← Добавляем это
6070 
6071                 const symbol = elem.querySelector('.element-symbol');
6072                 if (symbol) {
6073                     symbol.textContent = elemType === 'and' ? '∧' : '∨';
6074                 }
6075 
6076         } else if (elemType === 'output') {
6077                 const label = document.getElementById('prop-label').value || 'Выход';
6078                 const outputGroup = document.getElementById('prop-output-group').value
      || '';
6079 
6080                 elemData.props.label = label;
6081                 elemData.props.outputGroup = outputGroup;
6082 
6083                 const symbol = elem.querySelector('.element-symbol');
6084                 if (symbol) symbol.textContent = label;
6085             }
```

```
6086              else if (elemType === 'group') {
6087                  const title = document.getElementById('prop-title').value || 'Группа';
6088                  elemData.props.title = title;
6089                  const titleEl = elem.querySelector('.group-title');
6090                  if (titleEl) titleEl.textContent = title;
6091                  }
6092              const commentEl = document.getElementById('prop-comment');
6093              if (commentEl) elemData.props.comment = commentEl.value || '';
6094
6095              this.hideModal('modal-overlay');
6096
6097          } catch (error) {
6098              console.error('❌ Ошибка при сохранении свойств:', error);
6099              alert('Ошибка сохранения: ' + error.message);
6100          }
6101      },
6102
6103      /**
6104       * Показать свойства проекта
6105       */
6106      showProjectPropertiesModal() {
6107          const content = document.getElementById('project-modal-content');
6108          const project = AppState.project;
6109
6110          // Генерируем HTML для списка выходов только если модуль загружен
6111          let outputsHtml = '';
6112          if (typeof Outputs !== 'undefined' && AppState.outputs) {
6113              const logicalOutputsHtml = AppState.outputs.logical.length > 0
6114                  ? AppState.outputs.logical.map(output => `
6115                      <div class="output-item"
6116                          data-element-id="${output.elementId}"
6117                          onmouseenter="Outputs.highlightOutput('${output.elementId}',
      true)"
6118                          onmouseleave="Outputs.highlightOutput('${output.elementId}',
      false)"
6119                          onclick="Outputs.navigateToOutput('${output.elementId}');
      Modal.hideModal('project-modal-overlay');">
6120                          <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
      '❌'}</span>
6121                          <span class="output-name">${output.elementName}</span>
6122                          <span class="output-port">→ ${output.portLabel}</span>
6123                      </div>
6124                  `).join('')
6125                  : '<div class="no-outputs">Нет логических выходов</div>';
6126
6127              const numericOutputsHtml = AppState.outputs.numeric.length > 0
6128                  ? AppState.outputs.numeric.map(output => `
6129                      <div class="output-item numeric"
6130                          data-element-id="${output.elementId}"
6131                          onmouseenter="Outputs.highlightOutput('${output.elementId}',
      true)"
6132                          onmouseleave="Outputs.highlightOutput('${output.elementId}',
      false)"
6133                          onclick="Outputs.navigateToOutput('${output.elementId}');
      Modal.hideModal('project-modal-overlay');">
6134                          <span class="output-icon">🔢</span>
6135                          <span class="output-name">${output.elementName}</span>
6136                          <span class="output-port">→ значение</span>
6137                      </div>
6138                  `).join('')
6139                  : '<div class="no-outputs">Нет числовых выходов</div>';
6140
6141              outputsHtml = `
6142                  <div class="modal-row">
6143                      <label>Выходные сигналы схемы:</label>
```

```
                    <div class="outputs-container">
                        <div class="outputs-section">
                            <div class="outputs-section-title">
                                <span class="section-icon">🔀</span>
                                Логические выходы (${AppState.outputs.logical.length})
                            </div>
                            <div class="outputs-list">
                                ${logicalOutputsHtml}
                            </div>
                        </div>
                        <div class="outputs-section">
                            <div class="outputs-section-title">
                                <span class="section-icon">📐</span>
                                Числовые выходы (${AppState.outputs.numeric.length})
                            </div>
                            <div class="outputs-list">
                                ${numericOutputsHtml}
                            </div>
                        </div>
                    </div>
                    <div class="outputs-hint">
                        💡 Выходами автоматически становятся элементы, чьи выходные порты не подключены к другим элементам.
                        Кликните на выход, чтобы перейти к нему на схеме.
                    </div>
                </div>
            `;
        }

        content.innerHTML = `
            <div class="modal-row">
                <label>Код проекта:</label>
                <input type="text" id="project-code" value="${project.code || ''}" placeholder="Уникальный идентификатор">
            </div>

            <div class="modal-row">
                <label>Тип проекта:</label>
                <div class="project-type-selector">
                    <div class="project-type-btn ${project.type === PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
                        <div class="type-icon">📊</div>
                        <div class="type-name">Параметр</div>
                        <div class="type-desc">Вычисляемое значение</div>
                    </div>
                    <div class="project-type-btn ${project.type === PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
                        <div class="type-icon">📋</div>
                        <div class="type-name">Правило</div>
                        <div class="type-desc">Логическое условие</div>
                    </div>
                </div>
            </div>

            <div id="parameter-fields" class="conditional-fields ${project.type === PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
                <div class="modal-row">
                    <label>Описание:</label>
                    <textarea id="project-description" placeholder="Описание сигнала">${project.description || ''}</textarea>
                </div>
                <div class="modal-row">
                    <label>Размерность:</label>
                    <input type="text" id="project-dimension" value="${project.dimension || ''}" placeholder="Например: м/с, кг, °C">
```

```
                </div>
            </div>

            <div id="rule-fields" class="conditional-fields ${project.type === PROJECT_TYPE.RULE ? 'visible' : ''}">
                <div class="modal-row">
                    <label>Возможная причина:</label>
                    <textarea id="project-possible-cause" placeholder="Описание возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
                </div>
                <div class="modal-row">
                    <label>Методические указания:</label>
                    <textarea id="project-guidelines" placeholder="Инструкции и рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
                </div>
            </div>

            ${outputsHtml}
        `;

        // Обработчики переключения типа
        content.querySelectorAll('.project-type-btn').forEach(btn => {
            btn.addEventListener('click', () => {
                content.querySelectorAll('.project-type-btn').forEach(b => b.classList.remove('active'));
                btn.classList.add('active');

                const type = btn.dataset.type;
                document.getElementById('parameter-fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
                document.getElementById('rule-fields').classList.toggle('visible', type === PROJECT_TYPE.RULE);
            });
        });

        this.showModal('project-modal-overlay');
    },

    /**
     * Сохранить свойства проекта
     */
    saveProjectProperties() {
        const activeTypeBtn = document.querySelector('.project-type-btn.active');
        const type = activeTypeBtn ? activeTypeBtn.dataset.type : PROJECT_TYPE.PARAMETER;

        AppState.project.code = document.getElementById('project-code').value;
        AppState.project.type = type;

        if (type === PROJECT_TYPE.PARAMETER) {
            AppState.project.dimension = document.getElementById('project-dimension').value;
            AppState.project.description = document.getElementById('project-description').value || '';
            AppState.project.possibleCause = '';
            AppState.project.guidelines = '';
        } else {
            AppState.project.dimension = '';
            AppState.project.description = '';
            AppState.project.possibleCause = document.getElementById('project-possible-cause').value;
            AppState.project.guidelines = document.getElementById('project-guidelines').value;
        }

```

```
6256            this.hideModal('project-modal-overlay');
6257        }
6258    };
6259
6260    /**
6261     * Модуль управления выходными сигналами
6262     */
6263
6264    const Outputs = {
6265        /**
6266         * Обновление статуса выходных элементов
6267         * Вызывается при каждом изменении схемы
6268         */
6269        updateOutputStatus() {
6270            this.clearAllOutputHighlights();
6271            AppState.outputs.logical = [];
6272            AppState.outputs.numeric = [];
6273            updateFrameChildren();
6274
6275            // Обработка элементов-выходов
6276            Object.values(AppState.elements).forEach(elem => {
6277                if (!elem || elem.type !== 'output') return;
6278
6279                // Проверяем, к чему подключен вход этого выхода
6280                const inputConns = AppState.connections.filter(c =>
6281                    c.toElement === elem.id && c.toPort === 'in-0'
6282                );
6283
6284                // Каждое соединение к выходу — это отдельный выход
6285                inputConns.forEach((conn, index) => {
6286                    const fromElem = AppState.elements[conn.fromElement];
6287                    if (!fromElem) return;
6288
6289                    const outputType = conn.signalType;
6290                    const outputInfo = {
6291                        id: `${elem.id}_conn_${index}`,
6292                        elementId: elem.id,
6293                        sourceElementId: conn.fromElement,
6294                        sourcePort: conn.fromPort,
6295                        portIndex: 0,
6296                        portId: 'in-0',
6297                        type: outputType,
6298                        label: elem.props?.label || 'Выход',
6299                        elementType: 'output',
6300                        elementName: elem.props?.label || 'Выход',
6301                        name: elem.props?.label || 'Выход'
6302                    };
6303
6304                    if (outputType === SIGNAL_TYPE.LOGIC) {
6305                        AppState.outputs.logical.push(outputInfo);
6306                    } else if (outputType === SIGNAL_TYPE.NUMBER) {
6307                        AppState.outputs.numeric.push(outputInfo);
6308                    }
6309
6310                    // Подсветим входной порт
6311                    this.highlightOutputPort(elem.id, 0, outputType);
6312                });
6313            });
6314
6315            this.updateOutputCounter();
6316        },
6317
6318        /**
6319         * Очистка всех выделений выходов
6320         */
```

```
6321      clearAllOutputHighlights() {
6322          document.querySelectorAll('.port.output-active').forEach(port => {
6323              port.classList.remove('output-active');
6324          });
6325
6326          document.querySelectorAll('.element.has-output').forEach(elem => {
6327              elem.classList.remove('has-output');
6328          });
6329
6330          document.querySelectorAll('.element.output-ambiguous').forEach(el =>
      el.classList.remove('output-ambiguous'));
6331          document.querySelectorAll('.element.output-missing').forEach(el =>
      el.classList.remove('output-missing'));
6332      },
6333
6334      /**
6335       * Выделение выходного порта
6336       */
6337      highlightOutputPort(elemId, portIndex, portType) {
6338          const elem = document.getElementById(elemId);
6339          if (!elem) return;
6340
6341          const port = elem.querySelector(`.port.output[data-port="out-${portIndex}"]`);
6342          if (port) {
6343              port.classList.add('output-active');
6344          }
6345
6346          // Добавляем класс элементу (даёт общий визуал)
6347          elem.classList.add('has-output');
6348      },
6349
6350      /**
6351       * Обновление счётчика выходов в меню
6352       */
6353      updateOutputCounter() {
6354          const counter = document.getElementById('output-counter');
6355          if (counter) {
6356              const total = AppState.outputs.logical.length +
      AppState.outputs.numeric.length;
6357              counter.textContent = total;
6358              counter.style.display = total > 0 ? 'inline-block' : 'none';
6359          }
6360      },
6361
6362      /**
6363       * Получить все выходы для сохранения в проект
6364       */
6365      getOutputsForSave() {
6366          // Сохраняем информацию о frame/inner для рамок
6367          return {
6368              logical: AppState.outputs.logical.map(o => ({
6369                  id: o.id,
6370                  elementId: o.elementId,
6371                  frameId: o.frameId || null,
6372                  innerElementId: o.innerElementId || null,
6373                  portIndex: o.portIndex ?? o.innerPortIndex ?? null,
6374                  portLabel: o.label
6375              })),
6376              numeric: AppState.outputs.numeric.map(o => ({
6377                  id: o.id,
6378                  elementId: o.elementId,
6379                  frameId: o.frameId || null,
6380                  innerElementId: o.innerElementId || null,
6381                  portIndex: o.portIndex ?? o.innerPortIndex ?? null,
6382                  portLabel: o.label
```

```
6383              }))
6384          };
6385      },
6386
6387      /**
6388       * Подсветить конкретный выход (при наведении в списке)
6389       */
6390      highlightOutput(elementId, highlight = true) {
6391          const elem = document.getElementById(elementId);
6392          if (elem) {
6393              if (highlight) {
6394                  elem.classList.add('output-highlighted');
6395              } else {
6396                  elem.classList.remove('output-highlighted');
6397              }
6398          }
6399      },
6400
6401      /**
6402       * Перейти к элементу выхода на схеме (elementId — фокусируемый элемент; для рамок
      это id рамки)
6403       */
6404      navigateToOutput(elementId) {
6405          const elemData = AppState.elements[elementId];
6406          if (!elemData) return;
6407
6408          // Центрируем viewport на элементе
6409          const container = document.getElementById('workspace-container');
6410          const rect = container.getBoundingClientRect();
6411
6412          const centerX = elemData.x + elemData.width / 2;
6413          const centerY = elemData.y + elemData.height / 2;
6414
6415          AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
6416          AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;
6417
6418          Viewport.updateTransform();
6419
6420          // Выделяем элемент
6421          Elements.selectElement(elementId);
6422
6423          // Временная подсветка
6424          this.highlightOutput(elementId, true);
6425          setTimeout(() => this.highlightOutput(elementId, false), 2000);
6426      }
6427  };
6428
6429  /**
6430   * Модуль управления проектом (сохранение, загрузка)
6431   * project.js
6432   */
6433
6434  // --- миграция id: '-' -> '_' с обновлением всех ссылок ---
6435  function migrateIdsDashToUnderscore() {
6436    const map = {};
6437
6438    // 1) собрать map старых id → новых
6439    Object.values(AppState.elements).forEach(el => {
6440      if (typeof el.id === 'string' && el.id.includes('-')) {
6441        map[el.id] = el.id.replace(/-/g, '_');
6442      }
6443    });
6444
6445    if (!Object.keys(map).length) return;
6446
```

```
6447    // 2) DOM id + data-element
6448    Object.entries(map).forEach(([oldId, newId]) => {
6449      const dom = document.getElementById(oldId);
6450      if (dom) dom.id = newId;
6451
6452      if (dom) {
6453        dom.querySelectorAll('[data-element]').forEach(p => {
6454          if (p.dataset.element === oldId) p.dataset.element = newId;
6455        });
6456      }
6457    });
6458
6459    // 3) AppState.elements ключи
6460    Object.entries(map).forEach(([oldId, newId]) => {
6461      const el = AppState.elements[oldId];
6462      if (!el) return;
6463      el.id = newId;
6464      AppState.elements[newId] = el;
6465      delete AppState.elements[oldId];
6466    });
6467
6468    // 4) connections
6469    AppState.connections.forEach(c => {
6470      if (map[c.fromElement]) c.fromElement = map[c.fromElement];
6471      if (map[c.toElement]) c.toElement = map[c.toElement];
6472    });
6473
6474    // 5) формулы
6475    const escapeRegex = s => s.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
6476    Object.values(AppState.elements).forEach(el => {
6477      if (el.type === 'formula' && el.props?.expression) {
6478        let expr = el.props.expression;
6479        Object.entries(map).forEach(([oldId, newId]) => {
6480          const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
      z0-9_])`, 'g');
6481          expr = expr.replace(re, (m, p1) => `${p1}${newId}`);
6482        });
6483        el.props.expression = expr;
6484      }
6485    });
6486
6487    // 6) selected + modal
6488    if (map[AppState.selectedElement]) AppState.selectedElement =
      map[AppState.selectedElement];
6489    const modal = document.getElementById('modal-overlay');
6490    if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
      map[modal.dataset.elementId];
6491  }
6492
6493  const Project = {
6494      /**
6495       * Инициализация
6496       */
6497          /**
6498       * Инициализация
6499       */
6500  init() {
6501      document.getElementById('btn-new').addEventListener('click', () =>
      this.newProject());
6502      document.getElementById('btn-save').addEventListener('click', () =>
      this.saveProject());
6503      document.getElementById('btn-load').addEventListener('click', () =>
      this.openProjectListModal());
6504      document.getElementById('btn-project-settings').addEventListener('click', () => {
6505          Modal.showProjectPropertiesModal();
```

```
6506      });
6507
6508      // Работа с модалкой выбора проекта
6509      this.projectList = [];
6510      this.filteredProjectList = [];
6511      this.selectedProjectFilename = null;
6512
6513      document.getElementById('project-cancel').addEventListener('click', () =>
      this.closeProjectListModal());
6514      document.getElementById('project-refresh').addEventListener('click', () =>
      this.refreshProjectList());
6515
6516      document.getElementById('project-load').addEventListener('click', () => {
6517          if (this.selectedProjectFilename) {
6518              this.loadProjectFromList(this.selectedProjectFilename);
6519          }
6520      });
6521
6522      document.getElementById('project-search').addEventListener('input', (event) => {
6523          this.filterProjectList(event.target.value);
6524      });
6525  },
6526
6527      /**
6528       * Новый проект
6529       */
6530      newProject() {
6531          if (Object.keys(AppState.elements).length > 0) {
6532              if (!confirm('Создать новый проект? Несохранённые изменения будут
      потеряны.')) {
6533                  return;
6534              }
6535          }
6536
6537          document.getElementById('workspace').innerHTML = '';
6538          document.getElementById('connections-svg').innerHTML = '';
6539
6540          resetState();
6541          Viewport.updateTransform();
6542      },
6543
6544          /**
6545       * Запрос имени файла и загрузка с сервера
6546       */
6547      async loadProjectPrompt() {
6548          const filename = window.prompt(
6549              "Введите имя файла проекта для загрузки (с сервера). Пример:
      scheme_logic.json",
6550              AppState.project.code ? `${AppState.project.code}_$
      {AppState.project.type}.json` : "scheme_type.json"
6551          );
6552
6553          if (!filename) return; // Отмена
6554
6555          try {
6556              // Используем обертку из Settings.js для запроса к /api/project/load
6557              const data = await Settings.loadProject(filename);
6558
6559              // Если загрузка успешна, вызываем основную функцию обработки данных
6560              this._processLoadedData(data);
6561              alert(`Проект "${filename}" успешно загружен с сервера.`);
6562
6563          } catch (error) {
6564              console.error('Ошибка загрузки проекта:', error);
6565              alert(`Ошибка загрузки проекта: ${error.message}`);
```

```
6566          }
6567      },
6568
6569      /**
6570       * Сохранение проекта
6571       */
6572          async saveProject() {
6573      // 1. Проверяем свойства проекта
6574      if (!AppState.project.code) {
6575          Modal.showProjectPropertiesModal();
6576          alert('Пожалуйста, укажите код проекта перед сохранением.');
6577          return;
6578      }
6579
6580      // Обновляем размеры рамок перед сохранением
6581      updateFrameChildren();
6582      // нормализуем id
6583      migrateIdsDashToUnderscore();
6584
6585      // подчистим связи прямо перед сохранением
6586      const exists = (id) => !!AppState.elements[id];
6587      AppState.connections = (AppState.connections || [])
6588        .map(c => ({
6589          ...c,
6590          fromElement: exists(c.fromElement) ? c.fromElement : c.fromElement.replace(/-/
      g, '_'),
6591          toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g, '_')
6592        }))
6593        .filter(c => exists(c.fromElement) && exists(c.toElement))
6594        .filter((c, idx, arr) => {
6595          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
6596          return arr.findIndex(x =>
6597            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
6598          ) === idx;
6599        });
6600
6601      // Генерируем код заранее
6602      let generatedCode = '';
6603      if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function') {
6604          try {
6605              generatedCode = CodeGen.generate() || '';
6606          } catch (err) {
6607              console.error('Code generation failed:', err);
6608          }
6609      }
6610
6611      // НОВОЕ: получаем состояние визуализатора перед сохранением
6612      let visualizerState = AppState.project?.visualizer_state || null;
6613
6614      if (AppState.currentVisualizerToken) {
6615          try {
6616              const freshState = await App.fetchVisualizerState();
6617              if (freshState) {
6618                  visualizerState = freshState;
6619                  console.log('Состояние визуализатора обновлено перед сохранением');
6620              }
6621          } catch (err) {
6622              console.warn('Не удалось получить состояние визуализатора:', err);
6623          }
6624      }
6625
6626      // 2. Сборка объекта проекта
6627      const project = {
6628          version: '1.0',
6629          project: AppState.project,
```

```
6630          elements: AppState.elements,
6631          connections: AppState.connections,
6632          counter: AppState.elementCounter,
6633          viewport: {
6634              zoom: AppState.viewport.zoom,
6635              panX: AppState.viewport.panX,
6636              panY: AppState.viewport.panY
6637          },
6638          code: generatedCode,
6639          visualizer_state: visualizerState  // НОВОЕ: сохраняем состояние визуализатора
6640      };
6641
6642      const filename = `${AppState.project.code || 'scheme'}_$
      {AppState.project.type}.json`;
6643
6644      // 3. Сохранение на сервер
6645      try {
6646          await Settings.saveProject(filename, project);
6647
6648          // НОВОЕ: обновляем состояние в AppState после успешного сохранения
6649          AppState.project.visualizer_state = visualizerState;
6650
6651          alert(`Проект успешно сохранен на сервере как: ${filename}`);
6652      } catch (error) {
6653          console.error('Ошибка сохранения проекта:', error);
6654          alert(`Ошибка сохранения проекта: ${error.message}`);
6655      }
6656  },
6657
6658      async showProjectList() {
6659          try {
6660              const result = await Settings.listProjects(); // нужно реализовать в
      settings.js
6661              const list = result.projects || [];
6662
6663              if (list.length === 0) {
6664                  alert('Проекты в папке не найдены.');
6665                  return;
6666              }
6667
6668              const choice = window.prompt(
6669                  'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code ||
      p.filename} — ${p.description}`).join('\n') +
6670                  '\n\nВведите номер проекта для загрузки:',
6671                  '1'
6672              );
6673              const index = parseInt(choice, 10) - 1;
6674              if (isNaN(index) || !list[index]) return;
6675
6676              await this.loadProjectByFilename(list[index].filename);
6677          } catch (error) {
6678              console.error(error);
6679              alert('Не удалось получить список проектов: ' + error.message);
6680          }
6681      },
6682
6683      async loadProjectByFilename(filename) {
6684          try {
6685              const data = await Settings.loadProject(filename);
6686              this._processLoadedData(data);
6687              alert(`Проект "${filename}" загружен.`);
6688          } catch (error) {
6689              console.error(error);
6690              alert('Ошибка загрузки проекта: ' + error.message);
6691          }
```

```
6692      },
6693
6694      openProjectListModal() {
6695    const modal = document.getElementById('modal-project-list');
6696    modal.classList.remove('hidden');
6697    document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
6698    this.refreshProjectList();
6699  },
6700
6701  closeProjectListModal() {
6702    const modal = document.getElementById('modal-project-list');
6703    modal.classList.add('hidden');
6704    document.body.classList.remove('modal-open');
6705  },
6706
6707  async refreshProjectList() {
6708    const tbody = document.getElementById('project-list-body');
6709    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
6710    try {
6711      const result = await Settings.listProjects();
6712      this.projectList = result.projects || [];
6713      this.filteredProjectList = [...this.projectList];
6714      this.renderProjectList();
6715    } catch (err) {
6716      console.error(err);
6717      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
6718    }
6719  },
6720
6721  renderProjectList() {
6722    const tbody = document.getElementById('project-list-body');
6723    const loadBtn = document.getElementById('project-load');
6724    loadBtn.disabled = true;
6725    this.selectedProjectFilename = null;
6726
6727    if (!this.filteredProjectList.length) {
6728      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
6729      return;
6730    }
6731
6732    tbody.innerHTML = '';
6733    this.filteredProjectList.forEach((item) => {
6734      const tr = document.createElement('tr');
6735      tr.innerHTML = `
6736        <td>${item.filename}</td>
6737        <td>${item.code || ''}</td>
6738        <td>${item.description || ''}</td>
6739        <td>${item.type || ''}</td>
6740      `;
6741      tr.addEventListener('click', () => {
6742        this.highlightRow(tr);
6743        this.selectedProjectFilename = item.filename;
6744        loadBtn.disabled = false;
6745      });
6746      tr.addEventListener('dblclick', () => {
6747        this.highlightRow(tr);
6748        this.selectedProjectFilename = item.filename;
6749        loadBtn.disabled = false;
6750        this.loadProjectFromList(item.filename);
6751      });
6752      tbody.appendChild(tr);
```

```
  });
},

highlightRow(row) {
  const tbody = row.parentElement;
  [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
  row.classList.add('selected');
},


// Фильтр по поисковой строке
filterProjectList(query) {
  const q = (query || '').trim().toLowerCase();
  if (!q) {
    this.filteredProjectList = [...this.projectList];
  } else {
    this.filteredProjectList = this.projectList.filter((item) => {
      return [
        item.filename,
        item.code,
        item.description,
        item.type
      ].some((field) => (field || '').toLowerCase().includes(q));
    });
  }
  this.renderProjectList();
},

async loadProjectFromList(filename) {
  try {
    const data = await Settings.loadProject(filename);
    this._processLoadedData(data);
    this.closeProjectListModal();
    alert(`Проект "${filename}" успешно загружен.`);
  } catch (error) {
    console.error(error);
    alert('Ошибка загрузки проекта: ' + error.message);
  }
},









    /**
     * Загрузка проекта
     */
_processLoadedData(data) {
  try {
    document.getElementById('workspace').innerHTML = '';
    document.getElementById('connections-svg').innerHTML = '';
    resetState();

    if (data.project) {
      AppState.project = { ...AppState.project, ...data.project };
    }

    // НОВОЕ: загружаем состояние визуализатора
    if (data.visualizer_state) {
      AppState.project.visualizer_state = data.visualizer_state;
      console.log('Загружено состояние визуализатора:', data.visualizer_state);
    } else {
      AppState.project.visualizer_state = null;
    }
```

```
6818
6819      // НОВОЕ: сбрасываем токен предыдущей сессии визуализатора
6820      AppState.currentVisualizerToken = null;
6821
6822      AppState.elementCounter = data.counter || 0;
6823
6824      if (data.viewport) {
6825        AppState.viewport.zoom = data.viewport.zoom || 1;
6826        AppState.viewport.panX = data.viewport.panX || 0;
6827        AppState.viewport.panY = data.viewport.panY || 0;
6828      }
6829
6830      const elements = data.elements || {};
6831      Object.values(elements)
6832        .filter(e => e.type === 'output-frame')
6833        .forEach(elemData => {
6834          Elements.addElement(
6835            elemData.type,
6836            elemData.x,
6837            elemData.y,
6838            elemData.props,
6839            elemData.id,
6840            elemData.width,
6841            elemData.height
6842          );
6843        });
6844
6845      Object.values(elements)
6846        .filter(e => e.type !== 'output-frame')
6847        .forEach(elemData => {
6848          Elements.addElement(
6849            elemData.type,
6850            elemData.x,
6851            elemData.y,
6852            elemData.props,
6853            elemData.id,
6854            elemData.width,
6855            elemData.height
6856          );
6857        });
6858
6859      AppState.connections = data.connections || [];
6860
6861      // Миграция id: '-' -> '_'
6862      migrateIdsDashToUnderscore();
6863
6864      // очистка соединений: удалить битые и дубликаты
6865      const exists = (id) => !!AppState.elements[id];
6866
6867      AppState.connections = (AppState.connections || [])
6868        .filter(c => exists(c.fromElement) && exists(c.toElement))
6869        .filter((c, idx, arr) => {
6870          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
6871          return arr.findIndex(x =>
6872            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
6873          ) === idx;
6874        });
6875
6876      // корректно восстанавливаем счётчик
6877      const counterFromFile = Number(data.counter);
6878      AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
6879
6880      const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
6881          if (!el?.id) return max;
6882          const match = String(el.id).match(/_(\d+)$/);
```

```
6883          const num = match ? parseInt(match[1], 10) : NaN;
6884          return Number.isFinite(num) ? Math.max(max, num) : max;
6885      }, 0);
6886
6887      AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
6888
6889      Viewport.updateTransform();
6890      Connections.drawConnections();
6891      updateFrameChildren();
6892
6893    } catch (e) {
6894      alert('Ошибка обработки данных проекта: ' + e.message);
6895      console.error(e);
6896    }
6897  }
6898  };
6899
6900  // web/js/regenerate_code.js
6901  // Usage: node web/js/regenerate_code.js <project_json_path> <templates_json_path>
6902
6903  const fs = require('fs');
6904  const path = require('path');
6905  const vm = require('vm');
6906
6907  const projectPath = process.argv[2];
6908  const templatesPath = process.argv[3];
6909
6910  if (!projectPath || !templatesPath) {
6911    console.error('Usage: node web/js/regenerate_code.js <project_json_path>
     <templates_json_path>');
6912    process.exit(2);
6913  }
6914
6915  const filesToLoad = [
6916    path.join(__dirname, 'config.js'),
6917    path.join(__dirname, 'codegen_optimizer.js'),
6918    path.join(__dirname, 'codegen_graph.js'),
6919    path.join(__dirname, 'codegen.js'),
6920  ];
6921
6922  const utilsCode = `
6923  (function() {
6924    function splitArgsTopLevel(argStr) {
6925      const out = [];
6926      let cur = '';
6927      let depth = 0;
6928      for (let i = 0; i < argStr.length; i++) {
6929        const ch = argStr[i];
6930        if (ch === '(') depth++;
6931        if (ch === ')') depth--;
6932        if (ch === ',' && depth === 0) {
6933          out.push(cur.trim());
6934          cur = '';
6935        } else {
6936          cur += ch;
6937        }
6938      }
6939      if (cur.trim()) out.push(cur.trim());
6940      return out;
6941    }
6942
6943    function expandFormulaTemplates(expr, templatesMap) {
6944      if (!expr) return expr;
6945      if (!templatesMap) return expr;
6946
```

```
6947      for (let pass = 0; pass < 10; pass++) {
6948        let changed = false;
6949
6950        expr = expr.replace(/([A-Za-z_]\\w*)\\s*\\(([^()]|\\([^()]*\\))*\\)/g, (match,
      name) => {
6951          const tpl = templatesMap[name];
6952          if (!tpl) return match;
6953
6954          const open = match.indexOf('(');
6955          const close = match.lastIndexOf(')');
6956          const inside = match.slice(open + 1, close);
6957          const callArgs = splitArgsTopLevel(inside);
6958
6959          let formalArgs = [];
6960          let argsConfig = {};
6961
6962          if (Array.isArray(tpl.args)) {
6963            formalArgs = tpl.args;
6964          } else if (typeof tpl.args === 'object' && tpl.args !== null) {
6965            formalArgs = Object.keys(tpl.args);
6966            argsConfig = tpl.args;
6967          }
6968
6969          if (callArgs.length !== formalArgs.length) return match;
6970
6971          let body = String(tpl.body || '0');
6972          let conditions = [];
6973
6974          formalArgs.forEach((fName, i) => {
6975            const actualVal = callArgs[i];
6976            const re = new RegExp('\\\\b' + fName + '\\\\b', 'g');
6977            body = body.replace(re, '(' + actualVal + ')');
6978
6979            if (argsConfig[fName]) {
6980              const conf = argsConfig[fName];
6981              if (conf.min !== undefined && conf.min !== null) {
6982                conditions.push('(' + actualVal + ' >= ' + conf.min + ')');
6983              }
6984              if (conf.max !== undefined && conf.max !== null) {
6985                conditions.push('(' + actualVal + ' <= ' + conf.max + ')');
6986              }
6987            }
6988          });
6989
6990          let resultExpr = '(' + body + ')';
6991          if (conditions.length > 0) {
6992            const conditionString = conditions.join(' AND ');
6993            const fallbackValue = tpl.return_value !== undefined ? tpl.return_value : 0;
6994            resultExpr = 'WHEN(' + conditionString + ', ' + body + ', ' + fallbackValue
      + ')';
6995          }
6996
6997          changed = true;
6998          return resultExpr;
6999        });
7000
7001        if (!changed) break;
7002      }
7003
7004      return expr;
7005    }
7006
7007    // Прокидываем и в Utils, и как глобальные
7008    window.Utils = window.Utils || {};
7009    window.Utils.splitArgsTopLevel = splitArgsTopLevel;
```

```
    window.Utils.expandFormulaTemplates = expandFormulaTemplates;

    window.splitArgsTopLevel = splitArgsTopLevel;
    window.expandFormulaTemplates = expandFormulaTemplates;
  })();
  `;

  try {
    const project = JSON.parse(fs.readFileSync(projectPath, 'utf-8'));
    const templates = JSON.parse(fs.readFileSync(templatesPath, 'utf-8'));

    // Логгер в stderr
    const toStderr = (...args) => {
      try { process.stderr.write(args.join(' ') + '\n'); } catch (e) {}
    };

    const sandbox = {
      console: {
        log: toStderr,
        info: toStderr,
        warn: toStderr,
        error: toStderr,
        debug: toStderr,
      },
      AppState: {
        elements: project.elements || {},
        connections: project.connections || [],
        elementCounter: project.counter || 0,
        project: project.project || {}
      },
      Settings: {
        templates: templates.templates || [],
        getTemplatesMap() {
          const map = {};
          (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
          return map;
        }
      },
      document: {},
      performance: { now: () => Date.now() },
      window: {},
    };
    sandbox.window = sandbox;
    sandbox.self = sandbox;
    sandbox.global = sandbox;

    const context = vm.createContext(sandbox);

    vm.runInContext(utilsCode, context, { filename: 'utilsCode' });

    for (const f of filesToLoad) {
      const code = fs.readFileSync(f, 'utf-8');
      vm.runInContext(code, context, { filename: f });
    }

    const CodeGen = context.window.CodeGen || context.CodeGen;
    if (!CodeGen) {
      throw new Error("CodeGen is undefined after loading scripts");
    }

    const code = CodeGen.generate();
    project.code = code;

    // Только JSON в stdout
    process.stdout.write(JSON.stringify(project, null, 2));
```

```
} catch (e) {
  // Ошибки — в stderr, код возврата 1
  console.error(e && e.stack ? e.stack : String(e));
  process.exit(1);
}

// settings.js — ПОЛНАЯ ИСПРАВЛЕННАЯ ВЕРСИЯ

const Settings = {
  config: null,
  templates: null,
  apiUrl: '',  // ← Добавь это! Пустая строка = относительные пути

  async init() {
    try {
      const r = await fetch('/api/settings');
      if (r.ok) this.config = await r.json();
    } catch (e) {
      console.warn('Settings load failed:', e);
    }
    try {
      const t = await this.fetchFormulaTemplates();
      this.templates = t.templates || [];
    } catch (e) {
      this.templates = [];
    }
  },

  getTemplatesMap() {
    const map = {};
    (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
    return map;
  },

  // ← ОДНА функция fetchSignals с cache-busting
  async fetchSignals(mask, limit = 50) {
    const timestamp = Date.now();
    const url = `${this.apiUrl}/api/signals?q=${encodeURIComponent(mask || '')}
&limit=${limit}&_t=${timestamp}`;
    const r = await fetch(url);
    if (!r.ok) throw new Error('Failed to fetch signals');
    return await r.json();
  },

  async saveProject(filename, projectData) {
    if (!filename.endsWith('.json')) {
      filename += '.json';
    }
    const r = await fetch(`${this.apiUrl}/api/project/save`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        filename: filename,
        content: projectData
      })
    });
    if (!r.ok) throw new Error('Failed to save project');
    return r.json();
  },

  async listProjects() {
    const r = await fetch(`${this.apiUrl}/api/project/list`);
    if (!r.ok) throw new Error('Failed to list projects');
    return r.json();
  },
```

```
7139
7140    async fetchFormulaTemplates() {
7141      const r = await fetch(`${this.apiUrl}/api/formula-templates`);
7142      if (!r.ok) throw new Error('Failed to fetch formula templates');
7143      return await r.json();
7144    },
7145
7146    async loadProject(filename) {
7147      if (!filename.endsWith('.json')) {
7148        filename += '.json';
7149      }
7150      const r = await fetch(`${this.apiUrl}/api/project/load/$
      {encodeURIComponent(filename)}`);
7151      if (!r.ok) {
7152        if (r.status === 404) {
7153          throw new Error(`Project "${filename}" not found (404)`);
7154        }
7155        throw new Error('Failed to load project');
7156      }
7157      return r.json();
7158    }
7159  };
7160
7161  /**
7162   * Глобальное состояние приложения
7163   * state.js
7164   */
7165
7166  const AppState = {
7167      // Элементы схемы
7168      elements: {},
7169      connections: [],
7170      elementCounter: 0,
7171
7172      // Выделение
7173      selectedElement: null,
7174      selectedElements: [],
7175
7176      // Перетаскивание
7177      draggingElement: null,
7178      dragOffset: { x: 0, y: 0 },
7179      isDraggingFromPalette: false,
7180      dragPreview: null,
7181      dragType: null,
7182
7183      // Соединения
7184      connectingFrom: null,
7185      connectingFromType: null,
7186      tempLine: null,
7187
7188      // Resize
7189      resizing: null,
7190
7191      // Viewport (масштабирование и перемещение)
7192      viewport: {
7193          zoom: 1,
7194          panX: 0,
7195          panY: 0,
7196          isPanning: false,
7197          lastMouseX: 0,
7198          lastMouseY: 0
7199      },
7200
7201      // Свойства проекта
7202      project: {
```

```
7203          code: '',
7204          type: PROJECT_TYPE.PARAMETER,
7205          description: '',
7206          dimension: '',
7207          possibleCause: '',
7208          guidelines: '',
7209          visualizer_state: null  // НОВОЕ: состояние визуализатора
7210      },
7211
7212      // Выходные сигналы (автоматически определяются)
7213      outputs: {
7214          logical: [],
7215          numeric: []
7216      },
7217
7218      // НОВОЕ: токен текущей сессии визуализатора
7219      currentVisualizerToken: null
7220  };
7221
7222  /**
7223   * Сброс состояния
7224   */
7225  function resetState() {
7226      AppState.elements = {};
7227      AppState.connections = [];
7228      AppState.elementCounter = 0;
7229      AppState.selectedElement = null;
7230      AppState.selectedElements = [];
7231      AppState.draggingElement = null;
7232      AppState.connectingFrom = null;
7233      AppState.tempLine = null;
7234      AppState.resizing = null;
7235
7236      AppState.viewport = {
7237          zoom: 1,
7238          panX: 0,
7239          panY: 0,
7240          isPanning: false,
7241          lastMouseX: 0,
7242          lastMouseY: 0
7243      };
7244
7245      AppState.project = {
7246          code: '',
7247          type: PROJECT_TYPE.PARAMETER,
7248          description: '',
7249          dimension: '',
7250          possibleCause: '',
7251          guidelines: '',
7252          visualizer_state: null  // НОВОЕ: сбрасываем состояние визуализатора
7253      };
7254
7255      AppState.outputs = {
7256          logical: [],
7257          numeric: []
7258      };
7259
7260      // НОВОЕ: сбрасываем токен визуализатора
7261      AppState.currentVisualizerToken = null;
7262  }
7263
7264  /**
7265   * Вспомогательные функции
7266   * utils.js
7267   */
```

```
7268
7269  /**
7270   * Генерация уникального ID
7271   */
7272  function generateId() {
7273      AppState.elementCounter++;
7274      return `elem_${AppState.elementCounter}`;
7275  }
7276
7277  function getInputPortType(elementId, portIdentifier) {
7278      const element = AppState.elements[elementId];
7279      if (!element) return SIGNAL_TYPE.ANY;
7280
7281      const config = ELEMENT_TYPES[element.type];
7282      if (!config) return SIGNAL_TYPE.ANY;
7283
7284      let portIndex = portIdentifier;
7285
7286      // Обработка технического порта условия
7287      if (typeof portIdentifier === 'string') {
7288          if (portIdentifier === 'cond-0' && config.hasConditionPort) {
7289              return config.conditionPortType || SIGNAL_TYPE.LOGIC;
7290          }
7291
7292          if (portIdentifier.startsWith('in-')) {
7293              portIndex = parseInt(portIdentifier.split('-')[1], 10);
7294          }
7295      }
7296
7297      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
7298          portIndex = 0;
7299      }
7300
7301      // Динамические входы для AND/OR берут тип из конфига
7302      if ((element.type === 'and' || element.type === 'or')) {
7303          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
7304      }
7305
7306      if (element.type === 'formula') {
7307          return SIGNAL_TYPE.ANY;
7308      }
7309
7310      const types = config.inputTypes || [];
7311      if (types.length === 0) return SIGNAL_TYPE.ANY;
7312
7313      if (portIndex < types.length) {
7314          return types[portIndex] || SIGNAL_TYPE.ANY;
7315      }
7316
7317      return types[types.length - 1] || SIGNAL_TYPE.ANY;
7318  }
7319
7320  function getOutputPortType(elementId, portIdentifier) {
7321      const element = AppState.elements[elementId];
7322      if (!element) return SIGNAL_TYPE.ANY;
7323
7324      const config = ELEMENT_TYPES[element.type];
7325      if (!config) return SIGNAL_TYPE.ANY;
7326
7327      let portIndex = portIdentifier;
7328
7329      if (typeof portIdentifier === 'string') {
7330          if (portIdentifier.startsWith('out-')) {
7331              portIndex = parseInt(portIdentifier.split('-')[1], 10);
```

```
7332          }
7333      }
7334
7335      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
7336          portIndex = 0;
7337      }
7338
7339      const types = config.outputTypes || [];
7340      if (types.length === 0) return SIGNAL_TYPE.ANY;
7341
7342      if (portIndex < types.length) {
7343          return types[portIndex] || SIGNAL_TYPE.ANY;
7344      }
7345
7346      return types[types.length - 1] || SIGNAL_TYPE.ANY;
7347  }
7348  /**
7349   * Проверка совместимости типов сигналов
7350   *
7351   * Новая логика:
7352   * - ANY совместим со всем
7353   * - TRUE совместим с LOGIC, TRUE, ANY
7354   * - FALSE совместим с LOGIC, FALSE, ANY
7355   * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
7356   * - NUMERIC совместим с NUMERIC, ANY
7357   */
7358  function areTypesCompatible(outputType, inputType) {
7359      // Если один из типов ANY - совместимы
7360      if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
7361          return true;
7362      }
7363
7364      // Если типы одинаковые - совместимы
7365      if (outputType === inputType) {
7366          return true;
7367      }
7368
7369      // TRUE/FALSE совместимы с LOGIC
7370      if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
7371          inputType === SIGNAL_TYPE.LOGIC) {
7372          return true;
7373      }
7374
7375      // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
7376      if (outputType === SIGNAL_TYPE.LOGIC &&
7377          (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
7378          return true;
7379      }
7380
7381      return false;
7382  }
7383
7384  /**
7385   * Проверка, находится ли элемент внутри рамки
7386   */
7387  function isInsideFrame(elemId, frameId) {
7388      const elem = AppState.elements[elemId];
7389      const frame = AppState.elements[frameId];
7390
7391      if (!elem || !frame || frame.type !== 'output-frame') return false;
7392
7393      const elemCenterX = elem.x + elem.width / 2;
7394      const elemCenterY = elem.y + elem.height / 2;
7395
7396      return elemCenterX > frame.x &&
```

```
            elemCenterX < frame.x + frame.width &&
            elemCenterY > frame.y &&
            elemCenterY < frame.y + frame.height;
}

/**
 * Обновить принадлежность элементов к рамкам
 */
function updateFrameChildren() {
    // Сначала очистим children у рамок и parentFrame у всех элементов
    Object.values(AppState.elements).forEach(elem => {
        if (elem.type === 'output-frame') {
            elem.children = [];
        } else {
            // удаляем parentFrame по умолчанию (пересчитаем ниже)
            if (elem.parentFrame) delete elem.parentFrame;
        }
    });

    // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
попадает
    Object.values(AppState.elements).forEach(elem => {
        if (!elem || elem.type === 'output-frame') return;

        Object.values(AppState.elements).forEach(frame => {
            if (!frame || frame.type !== 'output-frame') return;

            if (isInsideFrame(elem.id, frame.id)) {
                // добавляем в массив детей рамки
                frame.children.push(elem.id);
                // отмечаем у элемента родительскую рамку
                if (AppState.elements[elem.id]) {
                    AppState.elements[elem.id].parentFrame = frame.id;
                }
            }
        });
    });
}

/**
 * Преобразование координат экрана в координаты холста
 */
function screenToCanvas(screenX, screenY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
    const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;

    return { x, y };
}

/**
 * Преобразование координат холста в координаты экрана
 */
function canvasToScreen(canvasX, canvasY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
    const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;

    return { x, y };
}

```

```
7461  /**
7462   * Проверка, является ли порт выходным (не подключен к другим элементам)
7463   */
7464  function isOutputPort(elemId, portIndex) {
7465      const portKey = `out-${portIndex}`;
7466
7467      // Проверяем, есть ли соединения от этого порта
7468      const hasConnection = AppState.connections.some(conn =>
7469          conn.fromElement === elemId && conn.fromPort === portKey
7470      );
7471
7472      return !hasConnection;
7473  }
7474
7475  /**
7476   * Получить информацию о выходном порте
7477   */
7478  function getOutputPortInfo(elemId, portIndex) {
7479      const elem = AppState.elements[elemId];
7480      if (!elem) return null;
7481
7482      const config = ELEMENT_TYPES[elem.type];
7483      if (!config) return null;
7484
7485      return {
7486          elementId: elemId,
7487          elementType: elem.type,
7488          elementName: config.name,
7489          portIndex: portIndex,
7490          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
7491          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
7492          // Дополнительная информация для идентификации
7493          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
7494      };
7495  }
7496
7497  function splitArgsTopLevel(argStr) {
7498    const out = [];
7499    let cur = '';
7500    let depth = 0;
7501    for (let i = 0; i < argStr.length; i++) {
7502      const ch = argStr[i];
7503      if (ch === '(') depth++;
7504      if (ch === ')') depth--;
7505      if (ch === ',' && depth === 0) {
7506        out.push(cur.trim());
7507        cur = '';
7508      } else {
7509        cur += ch;
7510      }
7511    }
7512    if (cur.trim()) out.push(cur.trim());
7513    return out;
7514  }
7515
7516  // js/codegen.js
7517
7518  function expandFormulaTemplates(expr, templatesMap) {
7519      if (!expr) return expr;
7520      if (!templatesMap) return expr;
7521
7522      // несколько проходов на случай вложенных шаблонов
7523      for (let pass = 0; pass < 10; pass++) {
7524          let changed = false;
```

```
7525
7526          // Регулярка ищет вызовы функций: funcName(arg1, arg2, ...)
7527          expr = expr.replace(/([A-Za-z_]\w*)\s*\(([^()]|\([^()]*\))*\)/g, (match, name)
      => {
7528              const tpl = templatesMap[name];
7529              if (!tpl) return match;
7530
7531              // 1. Извлекаем аргументы из вызова: h(10, 20, 30) -> ["10", "20", "30"]
7532              const open = match.indexOf('(');
7533              const close = match.lastIndexOf(')');
7534              const inside = match.slice(open + 1, close);
7535              const callArgs = splitArgsTopLevel(inside);
7536
7537              // 2. Определяем формальные параметры (ключи) и конфиг
7538              let formalArgs = [];
7539              let argsConfig = {};
7540
7541              if (Array.isArray(tpl.args)) {
7542                  // Старый формат: "args": ["p", "t"]
7543                  formalArgs = tpl.args;
7544              } else if (typeof tpl.args === 'object' && tpl.args !== null) {
7545                  // Новый формат: "args": { "p": {"min":...}, ... }
7546                  formalArgs = Object.keys(tpl.args);
7547                  argsConfig = tpl.args;
7548              }
7549
7550              // Если количество аргументов не совпало — не трогаем (чтобы не сломать)
7551              if (callArgs.length !== formalArgs.length) return match;
7552
7553              // 3. Подготовка тела функции
7554              let body = String(tpl.body || '0');
7555
7556              // 4. Сбор условий валидации (min/max)
7557              let conditions = [];
7558
7559              formalArgs.forEach((fName, i) => {
7560                  const actualVal = callArgs[i]; // То, что передали: "10" или
      "sensor_1"
7561
7562                  // Подстановка значения в тело: заменяем параметр p на 10
7563                  const re = new RegExp(`\\b${fName}\\b`, 'g');
7564                  body = body.replace(re, `(${actualVal})`);
7565
7566                  // Проверка ограничений (только для нового формата)
7567                  if (argsConfig[fName]) {
7568                      const conf = argsConfig[fName];
7569
7570                      // Проверка min
7571                      if (conf.min !== undefined && conf.min !== null) {
7572                          conditions.push(`(${actualVal} >= ${conf.min})`);
7573                      }
7574                      // Проверка max
7575                      if (conf.max !== undefined && conf.max !== null) {
7576                          conditions.push(`(${actualVal} <= ${conf.max})`);
7577                      }
7578                  }
7579              });
7580
7581              // 5. Формирование результата
7582              let resultExpr = `(${body})`;
7583
7584              // Если есть условия, заворачиваем в WHEN
7585              if (conditions.length > 0) {
7586                  const conditionString = conditions.join(' AND ');
7587                  const fallbackValue = tpl.return_value !== undefined ?
```

```
      tpl.return_value : 0;
7588
7589                  // WHEN(условия, формула, значение_по_умолчанию)
7590                  resultExpr = `WHEN(${conditionString}, ${body}, ${fallbackValue})`;
7591              }
7592
7593              changed = true;
7594              return resultExpr;
7595          });
7596
7597          if (!changed) break;
7598      }
7599
7600      return expr;
7601  }
7602
7603  /**
7604   * Модуль управления viewport (масштабирование и перемещение)
7605   * viewport.js
7606   */
7607
7608  const Viewport = {
7609      /**
7610       * Инициализация viewport
7611       */
7612      init() {
7613          this.setupZoomControls();
7614          this.setupPanning();
7615          this.setupMouseWheel();
7616          this.setupMinimap();
7617          this.setupCursorPosition();
7618          this.updateTransform();
7619          const container = document.getElementById('workspace-container');
7620          const rect = container.getBoundingClientRect();
7621          AppState.viewport.panX = 100; // немного отступить от левого края
7622          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
      AppState.viewport.zoom;
7623          this.updateTransform();
7624      },
7625
7626      /**
7627       * Настройка кнопок масштабирования
7628       */
7629      setupZoomControls() {
7630          document.getElementById('btn-zoom-in').addEventListener('click', () => {
7631              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
7632          });
7633
7634          document.getElementById('btn-zoom-out').addEventListener('click', () => {
7635              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
7636          });
7637
7638          document.getElementById('btn-zoom-reset').addEventListener('click', () => {
7639              this.setZoom(1);
7640              this.setPan(0, 0);
7641          });
7642
7643          document.getElementById('btn-zoom-fit').addEventListener('click', () => {
7644              this.fitToContent();
7645          });
7646      },
7647
7648      /**
7649       * Настройка перемещения (pan)
7650       */
```

```
    setupPanning() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('mousedown', (e) => {
            // Средняя кнопка мыши или пробел + левая кнопка
            if (e.button === 1 || (e.button === 0 && e.target === container)) {
                e.preventDefault();
                AppState.viewport.isPanning = true;
                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
                container.style.cursor = 'grabbing';
            }
        });

        document.addEventListener('mousemove', (e) => {
            if (AppState.viewport.isPanning) {
                const dx = e.clientX - AppState.viewport.lastMouseX;
                const dy = e.clientY - AppState.viewport.lastMouseY;

                this.setPan(
                    AppState.viewport.panX + dx,
                    AppState.viewport.panY + dy
                );

                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
            }
        });

        document.addEventListener('mouseup', (e) => {
            if (AppState.viewport.isPanning) {
                AppState.viewport.isPanning = false;
                document.getElementById('workspace-container').style.cursor = '';
            }
        });

        // Клавиша пробел для режима перемещения
        document.addEventListener('keydown', (e) => {
            if (e.code === 'Space' && !e.repeat) {
                document.getElementById('workspace-container').style.cursor = 'grab';
            }
        });

        document.addEventListener('keyup', (e) => {
            if (e.code === 'Space') {
                document.getElementById('workspace-container').style.cursor = '';
            }
        });
    },

    /**
     * Настройка масштабирования колесом мыши
     */
    setupMouseWheel() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('wheel', (e) => {
            e.preventDefault();

            const rect = container.getBoundingClientRect();
            const mouseX = e.clientX - rect.left;
            const mouseY = e.clientY - rect.top;

            // Позиция мыши на холсте до масштабирования
            const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
```

```
      AppState.viewport.zoom;
7716              const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
      AppState.viewport.zoom;
7717
7718              // Новый масштаб
7719              const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
      VIEWPORT_CONFIG.zoomStep;
7720              const newZoom = Math.max(
7721                  VIEWPORT_CONFIG.minZoom,
7722                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
7723              );
7724
7725              // Корректируем pan, чтобы точка под курсором осталась на месте
7726              const newPanX = mouseX - canvasPosBeforeX * newZoom;
7727              const newPanY = mouseY - canvasPosBeforeY * newZoom;
7728
7729              AppState.viewport.zoom = newZoom;
7730              AppState.viewport.panX = newPanX;
7731              AppState.viewport.panY = newPanY;
7732
7733              this.updateTransform();
7734          }, { passive: false });
7735      },
7736
7737      /**
7738       * Установить масштаб
7739       */
7740      setZoom(zoom) {
7741          const container = document.getElementById('workspace-container');
7742          const rect = container.getBoundingClientRect();
7743
7744          // Центр экрана
7745          const centerX = rect.width / 2;
7746          const centerY = rect.height / 2;
7747
7748          // Позиция центра на холсте
7749          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
7750          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
7751
7752          // Новый масштаб
7753          const newZoom = Math.max(
7754              VIEWPORT_CONFIG.minZoom,
7755              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
7756          );
7757
7758          // Корректируем pan
7759          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
7760          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
7761          AppState.viewport.zoom = newZoom;
7762
7763          this.updateTransform();
7764      },
7765
7766      /**
7767       * Установить смещение
7768       */
7769      setPan(x, y) {
7770          AppState.viewport.panX = x;
7771          AppState.viewport.panY = y;
7772          this.updateTransform();
7773      },
7774
7775      /**
```

```
7776        * Вписать содержимое в экран
7777        */
7778       fitToContent() {
7779           const elements = Object.values(AppState.elements);
7780           if (elements.length === 0) {
7781               this.setZoom(1);
7782               this.setPan(0, 0);
7783               return;
7784           }
7785
7786           // Находим границы содержимого
7787           let minX = Infinity, minY = Infinity;
7788           let maxX = -Infinity, maxY = -Infinity;
7789
7790           elements.forEach(elem => {
7791               minX = Math.min(minX, elem.x);
7792               minY = Math.min(minY, elem.y);
7793               maxX = Math.max(maxX, elem.x + elem.width);
7794               maxY = Math.max(maxY, elem.y + elem.height);
7795           });
7796
7797           const contentWidth = maxX - minX;
7798           const contentHeight = maxY - minY;
7799
7800           const container = document.getElementById('workspace-container');
7801           const rect = container.getBoundingClientRect();
7802
7803           const padding = 50;
7804           const availableWidth = rect.width - padding * 2;
7805           const availableHeight = rect.height - padding * 2;
7806
7807           const zoomX = availableWidth / contentWidth;
7808           const zoomY = availableHeight / contentHeight;
7809           const newZoom = Math.min(zoomX, zoomY, 1);
7810
7811           AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
7812           AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
      (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
7813           AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
      (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
7814
7815           this.updateTransform();
7816       },
7817
7818       /**
7819        * Обновить трансформацию
7820        */
7821       updateTransform() {
7822           const workspace = document.getElementById('workspace');
7823           const svg = document.getElementById('connections-svg');
7824
7825           const transform = `translate(${AppState.viewport.panX}px, $
      {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
7826
7827           workspace.style.transform = transform;
7828           svg.style.transform = transform;
7829
7830           // Обновляем отображение масштаба
7831           document.getElementById('zoom-level').textContent = `$
      {Math.round(AppState.viewport.zoom * 100)}%`;
7832
7833           // Обновляем мини-карту
7834           this.updateMinimap();
7835       },
7836
```

```
    /**
     * Настройка мини-карты
     */
    setupMinimap() {
        const minimap = document.getElementById('minimap');
        const canvas = document.getElementById('minimap-canvas');

        canvas.width = MINIMAP_CONFIG.width;
        canvas.height = MINIMAP_CONFIG.height;

        // Клик по мини-карте для перемещения
        minimap.addEventListener('click', (e) => {
            const rect = minimap.getBoundingClientRect();
            const x = e.clientX - rect.left;
            const y = e.clientY - rect.top;

            this.navigateToMinimapPosition(x, y);
        });
    },

    /**
     * Обновить мини-карту
     */
    updateMinimap() {
        const canvas = document.getElementById('minimap-canvas');
        const ctx = canvas.getContext('2d');
        const viewportEl = document.getElementById('minimap-viewport');

        // Очищаем
        ctx.fillStyle = '#0a0a1a';
        ctx.fillRect(0, 0, canvas.width, canvas.height);

        // Масштаб мини-карты
        const scale = Math.min(
            canvas.width / VIEWPORT_CONFIG.canvasWidth,
            canvas.height / VIEWPORT_CONFIG.canvasHeight
        );

        // Рисуем элементы
        Object.values(AppState.elements).forEach(elem => {
            const x = elem.x * scale;
            const y = elem.y * scale;
            const w = Math.max(elem.width * scale, 2);
            const h = Math.max(elem.height * scale, 2);

            ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
            ctx.fillRect(x, y, w, h);
        });

        // Рисуем viewport
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
        const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
        const vpW = (rect.width / AppState.viewport.zoom) * scale;
        const vpH = (rect.height / AppState.viewport.zoom) * scale;

        viewportEl.style.left = `${vpX}px`;
        viewportEl.style.top = `${vpY}px`;
        viewportEl.style.width = `${vpW}px`;
        viewportEl.style.height = `${vpH}px`;
    },

    /**
```

```
7902        * Перейти к позиции на мини-карте
7903        */
7904       navigateToMinimapPosition(minimapX, minimapY) {
7905           const canvas = document.getElementById('minimap-canvas');
7906           const container = document.getElementById('workspace-container');
7907           const rect = container.getBoundingClientRect();
7908
7909           const scale = Math.min(
7910               canvas.width / VIEWPORT_CONFIG.canvasWidth,
7911               canvas.height / VIEWPORT_CONFIG.canvasHeight
7912           );
7913
7914           const canvasX = minimapX / scale;
7915           const canvasY = minimapY / scale;
7916
7917           // Центрируем viewport на этой точке
7918           AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
7919           AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
7920
7921           this.updateTransform();
7922       },
7923
7924       /**
7925        * Отслеживание позиции курсора
7926        */
7927       setupCursorPosition() {
7928           const container = document.getElementById('workspace-container');
7929
7930           container.addEventListener('mousemove', (e) => {
7931               const pos = screenToCanvas(e.clientX, e.clientY);
7932               document.getElementById('cursor-pos').textContent =
7933                   `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
7934           });
7935       }
7936   };
7937
7938   index.html
7939
7940   <!DOCTYPE html>
7941   <html lang="ru">
7942   <head>
7943       <meta charset="UTF-8">
7944       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7945       <title>Редактор логических схем</title>
7946       <link rel="stylesheet" href="css/styles.css">
7947   </head>
7948   <body>
7949       <div id="app">
7950           <div id="menu">
7951               <button class="menu-btn" id="btn-new">📄 Новый</button>
7952               <button class="menu-btn" id="btn-save">💾 Сохранить</button>
7953               <button class="menu-btn" id="btn-load">📂 Загрузить</button>
7954               <button class="menu-btn" id="btn-generate-code">🧩 Код</button>
7955               <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</
      button>
7956               <button class="menu-btn" id="btn-visualize">📈 Визуализировать</button>
7957               <div class="menu-separator"></div>
7958               <div class="zoom-controls">
7959                   <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
7960                   <span id="zoom-level">100%</span>
7961                   <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
7962                   <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
7963                   <button class="menu-btn" id="btn-zoom-reset">1:1</button>
7964               </div>
7965               <input type="file" id="file-input" accept=".json">
```

```
7966          </div>
7967
7968          <div id="main">
7969              <div id="palette">
7970                  <h3>📦 Элементы</h3>
7971                  <div class="palette-section">
7972                      <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>
7973
7974                      <div class="palette-item" data-type="group">
7975                          <svg viewBox="0 0 60 40">
7976                          <rect x="6" y="8" width="48" height="24" rx="4"
7977                                  fill="none" stroke="#6b7280" stroke-width="2" stroke-
      dasharray="4,2"/>
7978                          <text x="14" y="25" fill="#6b7280" font-size="10" font-
      weight="bold">GROUP</text>
7979                          </svg>
7980                          <div class="palette-item-name">Группа</div>
7981                      </div>
7982                      </div>
7983
7984                  <div class="palette-section">
7985                      <div class="palette-section-title">ВХОДЫ</div>
7986
7987                      <div class="palette-item" data-type="input-signal">
7988                          <svg viewBox="0 0 60 40">
7989                              <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460"
      stroke="#4a90d9" stroke-width="2"/>
7990                              <text x="12" y="24" fill="#eee" font-size="10">IN</text>
7991                          </svg>
7992                          <div class="palette-item-name">Входной сигнал</div>
7993                      </div>
7994                  </div>
7995                  <div class="palette-section">
7996                      <div class="palette-section-title">ВЫХОДЫ</div>
7997
7998                      <div class="palette-item" data-type="output">
7999                          <svg viewBox="0 0 60 40">
8000                              <rect x="5" y="5" width="50" height="30" rx="6"
      fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
8001                              <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</
      text>
8002                          </svg>
8003                          <div class="palette-item-name">Выход</div>
8004                      </div>
8005                  </div>
8006
8007                  <div class="palette-section">
8008                      <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>
8009
8010                      <div class="palette-item" data-type="and">
8011                          <svg viewBox="0 0 60 40">
8012                              <rect x="5" y="5" width="50" height="30" rx="5"
      fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
8013                              <text x="22" y="25" fill="#eee" font-size="12" font-
      weight="bold">И</text>
8014                          </svg>
8015                          <div class="palette-item-name">И (AND)</div>
8016                      </div>
8017
8018                      <div class="palette-item" data-type="or">
8019                          <svg viewBox="0 0 60 40">
8020                              <rect x="5" y="5" width="50" height="30" rx="5"
      fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
8021                              <text x="12" y="25" fill="#eee" font-size="11" font-
      weight="bold">ИЛИ</text>
```

```
                        </svg>
                        <div class="palette-item-name">ИЛИ (OR)</div>
                    </div>

                    <div class="palette-item" data-type="not">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
                            <text x="12" y="25" fill="#eee" font-size="11" font-
    weight="bold">НЕТ</text>
                        </svg>
                        <div class="palette-item-name">НЕТ (NOT)</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">СРАВНЕНИЕ</div>

                    <div class="palette-item" data-type="if">
                        <svg viewBox="0 0 60 40">
                            <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460"
    stroke="#e94560" stroke-width="2"/>
                            <text x="14" y="24" fill="#eee" font-size="9" font-
    weight="bold">ЕСЛИ</text>
                        </svg>
                        <div class="palette-item-name">ЕСЛИ (IF)</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>

                    <div class="palette-item" data-type="separator">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="8" width="50" height="24" rx="3"
    fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
                            <text x="8" y="25" fill="#f59e0b" font-size="10" font-
    weight="bold">✓/✗</text>
                        </svg>
                        <div class="palette-item-name">Сепаратор</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">ЗНАЧЕНИЯ</div>

                    <div class="palette-item" data-type="const">
                        <svg viewBox="0 0 60 40">
                            <rect x="10" y="8" width="40" height="24" rx="3"
    fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
                            <text x="24" y="25" fill="#3b82f6" font-size="14" font-
    weight="bold">C</text>
                        </svg>
                        <div class="palette-item-name">Константа</div>
                    </div>

                    <div class="palette-item" data-type="formula">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
                            <text x="12" y="25" fill="#f59e0b" font-size="11" font-
    weight="bold">f(x)</text>
                        </svg>
                        <div class="palette-item-name">Формула</div>
                    </div>
```

```
                </div>

                <div class="type-legend">
                    <div class="type-legend-item">
                        <div class="type-legend-dot logic"></div>
                        <span>Логический</span>
                    </div>
                    <div class="type-legend-item">
                        <div class="type-legend-dot number"></div>
                        <span>Числовой</span>
                    </div>
                </div>
            </div>

            <div id="workspace-container">
                <svg id="connections-svg"></svg>
                <div id="workspace"></div>
                  <!-- Прямоугольник для выделения элементов -->
                <div id="selection-rect"></div>

                <!-- Мини-карта -->
                <div id="minimap">
                    <div id="minimap-viewport"></div>
                    <canvas id="minimap-canvas"></canvas>
                </div>

                <!-- Координаты и информация -->
                <div id="viewport-info">
                    <span id="cursor-pos">X: 0, Y: 0</span>
                    <span id="selection-info"></span>
                </div>
            </div>
        </div>
    </div>

    <!-- Модальные окна -->
    <div id="modal-overlay">
        <div id="modal">
            <h3 id="modal-title">Свойства элемента</h3>
            <div id="modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="modal-cancel">Отмена</button>
                <button class="modal-btn save" id="modal-save">Сохранить</button>
            </div>
        </div>
    </div>

    <!-- Модальное окно свойств проекта -->
    <div id="project-modal-overlay" class="modal-overlay-class">
        <div id="project-modal" class="modal-class">
            <h3>Свойства проекта</h3>
            <div id="project-modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="project-modal-cancel">Отмена</
button>
                <button class="modal-btn save" id="project-modal-save">Сохранить</
button>
            </div>
        </div>
    </div>

    <div id="code-modal-overlay" class="modal-overlay-class">
        <div id="code-modal" class="modal-class">
            <h3>Сгенерированный код</h3>
            <textarea id="code-output" style="width:100%; height:300px;"></textarea>
```

```
8140             <div class="modal-buttons">
8141                 <button class="modal-btn cancel" id="code-modal-close">Закрыть</
      button>
8142             </div>
8143         </div>
8144     </div>
8145 
8146     <div id="context-menu">
8147         <div class="context-item" id="ctx-properties">⚙️ Свойства</div>
8148         <div class="context-item" id="ctx-copy">📋 Копировать</div>
8149         <div class="context-item" id="ctx-delete">🗑️ Удалить</div>
8150     </div>
8151 
8152    <!-- Модули JavaScript -->
8153     <!-- Модули JavaScript -->
8154     <script src="js/config.js"></script>
8155     <script src="js/state.js"></script>
8156     <script src="js/utils.js"></script>
8157     <script src="js/viewport.js"></script>
8158     <script src="js/elements.js"></script>
8159     <script src="js/connections.js"></script>
8160     <script src="js/outputs.js"></script>  <!-- ← Этот файл опционален теперь -->
8161     <script src="js/modal.js"></script>
8162     <script src="js/project.js"></script>
8163     <script src="js/codegen_graph.js"></script>
8164     <script src="js/codegen_optimizer.js"></script>
8165     <script src="js/codegen.js"></script>
8166     <script src="js/settings.js"></script>
8167 
8168     <script src="js/app.js"></script>
8169 
8170     <div id="modal-project-list" class="modal hidden">
8171         <div class="modal__content modal__content--wide">
8172             <h2 class="modal__title">Выбор проекта</h2>
8173 
8174             <div class="project-list__toolbar">
8175             <input id="project-search" type="text" placeholder="Фильтр по имени или
      описанию…" />
8176             <button id="project-refresh" class="btn btn-secondary">Обновить</button>
8177             </div>
8178 
8179             <div class="project-list__table-container">
8180             <table class="project-list__table">
8181                 <thead>
8182                 <tr>
8183                     <th>Файл</th>
8184                     <th>Tagname</th>
8185                     <th>Description</th>
8186                     <th>Тип</th>
8187                 </tr>
8188                 </thead>
8189                 <tbody id="project-list-body">
8190                 <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
8191                 </tbody>
8192             </table>
8193             </div>
8194 
8195             <div class="modal__actions">
8196                 <button id="project-cancel" class="btn btn-secondary">Отмена</button>
8197                 <button id="project-load" class="btn btn-primary" disabled>Загрузить</
      button>
8198             </div>
8199         </div>
8200     </div>
8201 
```

```
</body>
</html>

styles.css

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: #1a1a2e;
    color: #eee;
    overflow: hidden;
}

#app {
    display: flex;
    flex-direction: column;
    height: 100vh;
}

/* ====== МЕНЮ ====== */
#menu {
    background: #16213e;
    padding: 10px 20px;
    display: flex;
    gap: 10px;
    align-items: center;
    border-bottom: 2px solid #0f3460;
    z-index: 100;
    flex-wrap: wrap;
}

.menu-btn {
    background: #0f3460;
    color: #eee;
    border: none;
    padding: 8px 16px;
    border-radius: 5px;
    cursor: pointer;
    transition: background 0.3s;
    font-size: 13px;
}

.menu-btn:hover {
    background: #e94560;
}

.menu-separator {
    width: 1px;
    height: 30px;
    background: #0f3460;
    margin: 0 10px;
}

.zoom-controls {
    display: flex;
    align-items: center;
    gap: 8px;
    background: #0a0a1a;
    padding: 5px 10px;
    border-radius: 5px;
```

```
}

.zoom-btn {
    width: 30px;
    height: 30px;
    padding: 0;
    font-size: 18px;
    font-weight: bold;
}

#zoom-level {
    min-width: 50px;
    text-align: center;
    font-size: 12px;
    color: #aaa;
}

/* ====== ОСНОВНАЯ ОБЛАСТЬ ====== */
#main {
    display: flex;
    flex: 1;
    overflow: hidden;
}

/* ====== ПАЛИТРА ====== */
#palette {
    width: 200px;
    background: #16213e;
    padding: 15px;
    border-right: 2px solid #0f3460;
    overflow-y: auto;
    z-index: 10;
    flex-shrink: 0;
}

#palette h3 {
    margin-bottom: 15px;
    color: #e94560;
    text-align: center;
    font-size: 14px;
}

.palette-section {
    margin-bottom: 15px;
}

.palette-section-title {
    font-size: 11px;
    color: #888;
    margin-bottom: 8px;
    padding-bottom: 3px;
    border-bottom: 1px solid #333;
}

.palette-item {
    background: #0f3460;
    padding: 8px;
    margin-bottom: 6px;
    border-radius: 8px;
    cursor: grab;
    text-align: center;
    transition: all 0.3s;
    border: 2px solid transparent;
    user-select: none;
}
```

```
8332
8333  .palette-item:hover {
8334      border-color: #e94560;
8335      transform: scale(1.02);
8336  }
8337
8338  .palette-item:active {
8339      cursor: grabbing;
8340  }
8341
8342  .palette-item svg {
8343      width: 50px;
8344      height: 32px;
8345      margin-bottom: 2px;
8346      pointer-events: none;
8347  }
8348
8349  .palette-item-name {
8350      font-size: 10px;
8351      color: #aaa;
8352      pointer-events: none;
8353  }
8354
8355  .type-legend {
8356      margin-top: 15px;
8357      padding-top: 10px;
8358      border-top: 1px solid #333;
8359      font-size: 10px;
8360  }
8361
8362  .type-legend-item {
8363      display: flex;
8364      align-items: center;
8365      gap: 8px;
8366      margin-bottom: 5px;
8367  }
8368
8369  .type-legend-dot {
8370      width: 12px;
8371      height: 12px;
8372      border-radius: 50%;
8373      border: 2px solid #fff;
8374  }
8375  .type-legend-dot.logic { background: #a855f7; }
8376  .type-legend-dot.number { background: #3b82f6; }
8377
8378  /* ====== РАБОЧАЯ ОБЛАСТЬ ====== */
8379  #workspace-container {
8380      flex: 1;
8381      position: relative;
8382      overflow: hidden;
8383      background-color: #0a0a1a;
8384      background-image:
8385          linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
8386          linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);
8387      background-size: 25px 25px;
8388  }
8389
8390  #workspace {
8391      position: absolute;
8392      transform-origin: 0 0;
8393      width: 5000px;
8394      height: 5000px;
8395  }
8396
```

```
#connections-svg {
    position: absolute;
    transform-origin: 0 0;
    pointer-events: none;
    z-index: 5;
    width: 5000px;
    height: 5000px;
}

#connections-svg path {
    pointer-events: stroke;
}

/* ====== ЭЛЕМЕНТЫ ====== */
.element {
    position: absolute;
    background: #0f3460;
    border: 2px solid #4a90d9;
    border-radius: 8px;
    cursor: move;
    user-select: none;
    z-index: 10;
    display: flex;
    flex-direction: column;
}

.element.selected {
    border-color: #e94560;
    box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
}

.element-header {
    background: #4a90d9;
    padding: 5px 10px;
    border-radius: 5px 5px 0 0;
    font-size: 11px;
    font-weight: bold;
    text-align: center;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
}

.element-body {
    padding: 10px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex: 1;
    gap: 8px;
}

.element-symbol {
    font-size: 16px;
    font-weight: bold;
    flex: 1;
    text-align: center;
    padding: 0 5px;
    word-break: break-all;
    color: #eee;
}

/* ====== ПОРТЫ ====== */
.ports-left, .ports-right {
    display: flex;
```

```
8462      flex-direction: column;
8463      justify-content: space-around;
8464      gap: 10px;
8465      height: 100%;
8466  }
8467
8468  .port {
8469      width: 14px;
8470      height: 14px;
8471      border-radius: 50%;
8472      border: 2px solid #fff;
8473      cursor: crosshair;
8474      transition: all 0.2s;
8475      position: relative;
8476      flex-shrink: 0;
8477  }
8478
8479  .port:hover { transform: scale(1.3); }
8480  .port.input { margin-left: -8px; }
8481  .port.output { margin-right: -8px; }
8482  .port.connected { background: #4ade80; }
8483
8484  /* Типы портов */
8485  .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
8486  .port.logic-port:hover { background: #c084fc; }
8487  .port.logic-port.connected { background: #7c3aed; }
8488
8489  .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
8490  .port.number-port:hover { background: #60a5fa; }
8491  .port.number-port.connected { background: #2563eb; }
8492
8493  .port.any-port { background: #6b7280; border-color: #d1d5db; }
8494  .port.any-port:hover { background: #9ca3af; }
8495  .port.any-port.connected { background: #4b5563; }
8496
8497  .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !
      important; }
8498  .port.output.no-port { background: #f87171 !important; border-color: #fecaca !
      important; }
8499
8500  .port.incompatible { opacity: 0.3; cursor: not-allowed; }
8501  .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
8502
8503  /* ====== RESIZE HANDLES ====== */
8504  .resize-handle {
8505      position: absolute;
8506      width: 12px;
8507      height: 12px;
8508      background: #e94560;
8509      border: 1px solid #fff;
8510      border-radius: 3px;
8511      z-index: 20;
8512      opacity: 0;
8513      transition: opacity 0.2s;
8514  }
8515  .element.selected .resize-handle { opacity: 0.8; }
8516  .resize-handle:hover { opacity: 1; }
8517  .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
8518  .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor:
      ew-resize; }
8519  .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%);
      cursor: ns-resize; }
8520
8521
8522  /* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
```

```
.element.input-signal {
    background: transparent;
    border: none;
}

.element.input-signal .element-header {
    display: none; /* У трапеции нет заголовка */
}

.element.input-signal .element-body {
    padding: 0;
    background: #0f3460;
    border: 2px solid #4a90d9;
    clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding-left: 15px;
    padding-right: 25px;
}

.element.input-signal .element-symbol {
    text-align: left;
    color: #eee;
}

.element.input-signal.selected .element-body {
    border-color: #e94560;
}

/* ====== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ====== */
.element.output {
    background: rgba(16, 185, 129, 0.1);
    border: 2px dashed #10b981;
}

.element.output .element-header {
    display: none; /* У выхода нет заголовка */
}

.element.output .element-body {
    padding-left: 20px;
}

.element.output .element-symbol {
    color: #10b981;
    font-size: 14px;
}

.element.output.selected {
    border-color: #e94560;
    border-style: dashed;
}


/* Formula condition port */
/* Универсальный стиль для технического порта (сверху) */
.element.has-condition-port {
    margin-top: 30px; /* Даем место порту над элементом */
}

.condition-port-wrapper {
    position: absolute;
    top: -28px;
    left: 50%;
```

```
8588      transform: translateX(-50%);
8589      display: flex;
8590      flex-direction: column;
8591      align-items: center;
8592      gap: 4px;
8593      pointer-events: none;
8594      z-index: 21;
8595  }
8596
8597  .condition-port-label {
8598      font-size: 10px;
8599      color: #f59e0b;
8600      font-weight: 600;
8601      white-space: nowrap;
8602  }
8603
8604  .port.condition-port {
8605      pointer-events: auto;
8606      width: 16px;
8607      height: 16px;
8608      border-radius: 50%;
8609      border: 2px solid #f59e0b;
8610      background: #fff7ed;
8611      margin: 0; /* Сбрасываем лишние отступы */
8612  }
8613  .element.formula .condition-port:hover { background: #fde68a; }
8614
8615
8616  /* ====== СОЕДИНЕНИЯ ====== */
8617  .connection {
8618      fill: none !important;  /* ← добавляем !important */
8619      stroke: #4a90d9;
8620      stroke-width: 2.5;
8621  }
8622  .connection:hover {
8623      stroke: #e94560;
8624      stroke-width: 4;
8625  }
8626
8627  .connection.logic-conn { stroke: #a855f7; }
8628  .connection.numeric-conn { stroke: #3b82f6; }
8629  .connection.any-conn { stroke: #6b7280; }
8630  .connection.true-conn { stroke: #4ade80; }
8631  .connection.false-conn { stroke: #f87171; }
8632
8633  .connection.yes-conn { stroke: #4ade80; }
8634  .connection.no-conn { stroke: #f87171; }
8635
8636  .temp-connection {
8637      fill: none !important;  /* ← добавляем !important */
8638      stroke: #e94560;
8639      stroke-width: 2;
8640      stroke-dasharray: 5, 5;
8641  }
8642  .temp-connection.invalid { stroke: #ef4444; }
8643
8644  /* ====== ПРОЧЕЕ ====== */
8645  .drag-preview {
8646      position: fixed;
8647      pointer-events: none;
8648      opacity: 0.8;
8649      z-index: 1000;
8650      background: #0f3460;
8651      border: 2px solid #e94560;
8652      border-radius: 8px;
```

```
    padding: 10px 15px;
    color: #fff;
    font-size: 12px;
}

#minimap {
    position: absolute;
    bottom: 20px;
    right: 20px;
    width: 200px;
    height: 150px;
    background: #16213e;
    border: 2px solid #0f3460;
    border-radius: 8px;
    overflow: hidden;
    z-index: 50;
}

#minimap-canvas { width: 100%; height: 100%; }
#minimap-viewport {
    position: absolute;
    border: 2px solid #e94560;
    background: rgba(233, 69, 96, 0.2);
    pointer-events: none;
}

#viewport-info {
    position: absolute;
    bottom: 20px;
    left: 20px;
    background: rgba(22, 33, 62, 0.9);
    padding: 8px 12px;
    border-radius: 5px;
    font-size: 11px;
    color: #888;
    z-index: 50;
    display: flex;
    gap: 15px;
}
#selection-info { color: #e94560; }

#modal-overlay, .modal-overlay-class {
    display: none;
    position: fixed;
    top: 0; left: 0;
    width: 100%; height: 100%;
    background: rgba(0, 0, 0, 0.7);
    z-index: 1000;
    justify-content: center;
    align-items: center;
}

#modal, .modal-class {
    background: #16213e;
    border-radius: 10px;
    padding: 20px;
    min-width: 400px;
    max-width: 600px;
    max-height: 80vh;
    overflow-y: auto;
    border: 2px solid #0f3460;
}

#modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
.modal-row { margin-bottom: 15px; }
```

```
.modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
.modal-row input, .modal-row select, .modal-row textarea {
    width: 100%;
    padding: 10px;
    background: #0f3460;
    border: 1px solid #4a90d9;
    border-radius: 5px;
    color: #eee;
    font-size: 14px;
}
.modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline: none; border-color: #e94560; }
.modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
.signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-radius: 5px; padding: 5px; margin-top: 5px; }
.signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size: 12px; }
.signal-item:hover { background: #4a90d9; }
.modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
.modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer; font-size: 14px; transition: background 0.3s; }
.modal-btn.save { background: #4ade80; color: #000; }
.modal-btn.save:hover { background: #22c55e; }
.modal-btn.cancel { background: #6b7280; color: #fff; }
.modal-btn.cancel:hover { background: #4b5563; }

#context-menu {
    display: none;
    position: fixed;
    background: #16213e;
    border: 1px solid #0f3460;
    border-radius: 5px;
    padding: 5px 0;
    z-index: 1001;
    min-width: 150px;
    box-shadow: 0 5px 20px rgba(0,0,0,0.3);
}
.context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition: background 0.2s; }
.context-item:hover { background: #0f3460; }

#file-input { display: none; }

.project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
.project-type-btn { flex: 1; padding: 15px; background: #0f3460; border: 2px solid #4a90d9; border-radius: 8px; color: #eee; cursor: pointer; text-align: center; transition: all 0.3s; }
.project-type-btn:hover { border-color: #e94560; }
.project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
.project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
.project-type-btn .type-name { font-weight: bold; }
.project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }

.conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-radius: 8px; margin-top: 10px; }
.conditional-fields.visible { display: block; }

::-webkit-scrollbar { width: 8px; height: 8px; }
::-webkit-scrollbar-track { background: #0a0a1a; }
::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
::-webkit-scrollbar-thumb:hover { background: #e94560; }

/* Стили для выходов */
.output-btn { position: relative; }
```

```
8774  .output-counter { display: inline-block; background: #e94560; color: white; font-size:
      11px; font-weight: bold; padding: 2px 6px; border-radius: 10px; margin-left: 5px; min-
      width: 18px; text-align: center; }
8775  .output-counter:empty, .output-counter[style*="display: none"] { display: none; }
8776  .element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
8777  .element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important;
      border-color: #fbbf24 !important; }
8778  .port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation:
      pulse-output 1.5s infinite; }
8779  @keyframes pulse-output {
8780      0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
8781      50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
8782  }
8783
8784  .outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-
      height: 250px; overflow-y: auto; }
8785  .outputs-section { margin-bottom: 15px; }
8786  .outputs-section:last-child { margin-bottom: 0; }
8787  .outputs-section-title { color: #10b981; font-weight: bold; font-size: 13px; margin-
      bottom: 10px; padding-bottom: 5px; border-bottom: 1px solid #333; display: flex;
      align-items: center; gap: 8px; }
8788  .outputs-section-title .section-icon { font-size: 16px; }
8789  .outputs-list { display: flex; flex-direction: column; gap: 5px; }
8790  .output-item { display: flex; align-items: center; gap: 10px; padding: 8px 12px;
      background: rgba(16, 185, 129, 0.1); border: 1px solid rgba(16, 185, 129, 0.3);
      border-radius: 5px; cursor: pointer; transition: all 0.2s; }
8791  .output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981;
      transform: translateX(5px); }
8792  .output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59,
      130, 246, 0.3); }
8793  .output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color:
      #3b82f6; }
8794  .output-icon { font-size: 14px; }
8795  .output-name { font-weight: bold; color: #eee; }
8796  .output-port { color: #888; font-size: 12px; margin-left: auto; }
8797  .no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
8798  .outputs-hint { margin-top: 10px; padding: 10px; background: rgba(59, 130, 246, 0.1);
      border-radius: 5px; font-size: 12px; color: #888; line-height: 1.4; }
8799  .element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-
      color: rgba(240,80,80,0.8) !important; }
8800  .element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-
      color: rgba(250,200,30,0.8) !important; }
8801  /* TRUE/FALSE порты (для сепаратора) */
8802  .port.true-port {
8803      background: #4ade80 !important;
8804      border-color: #bbf7d0 !important;
8805  }
8806  .port.true-port:hover {
8807      background: #22c55e !important;
8808  }
8809  .port.true-port.connected {
8810      background: #16a34a !important;
8811  }
8812
8813  .port.false-port {
8814      background: #f87171 !important;
8815      border-color: #fecaca !important;
8816  }
8817  .port.false-port:hover {
8818      background: #ef4444 !important;
8819  }
8820  .port.false-port.connected {
8821      background: #dc2626 !important;
8822  }
8823
```

```
/* Сепаратор стиль */
.element.separator {
    background: #0f3460;
    border: 2px solid #f59e0b;
}

.element.separator.selected {
    border-color: #e94560;
    box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
}

/* === Выделение рамкой === */
#selection-rect {
  position: absolute;
  border: 1px dashed #e94560;
  background: rgba(233, 69, 96, 0.1);
  pointer-events: none;
  display: none;
  z-index: 200;
}

/* === Кастомный элемент “Группа” === */
.element.group {
  background: rgba(107, 114, 128, 0.12);
  border: 2px dashed #6b7280;
  border-radius: 8px;
  position: absolute;
  z-index: 1;              /* ниже обычных элементов (у них z-index: 10) */
}

.element.group .group-title {
  pointer-events: auto;
}

.group-title {
  position: absolute;
  top: -20px;
  left: 5px;
  font-size: 11px;
  color: #ccc;
  background: #16213e;
  padding: 2px 6px;
  border-radius: 4px;
  pointer-events: auto; /* можно кликнуть для выбора */
}

.modal.hidden { display: none; }
.modal { position: fixed; inset: 0; display: flex; align-items: center; justify-content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
.modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px; max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow: hidden; }
.modal__content--wide { width: 800px; }
.modal__title { margin: 0; }

.project-list__toolbar { display: flex; gap: 12px; }
.project-list__toolbar input { flex: 1; padding: 6px 10px; }
.project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd; border-radius: 6px; }
.project-list__table { width: 100%; border-collapse: collapse; }
.project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom: 1px solid #eee; }
.project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
.project-list__table tbody tr:hover { background: #f0f6ff; }
.project-list__empty { text-align: center; color: #888; padding: 16px; }
```

```
.modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
.project-list__table th,
.project-list__table td {
  color: #111;          /* насыщённый чёрный текст */
  padding: 8px 12px;
  border-bottom: 1px solid #eee;
}
.modal__content--wide {
  width: 860px;
  max-height: 90vh;    /* занимает 90% экрана */
}

.project-list__table-container {
  flex: 1;
  overflow: auto;
  border: 1px solid #ddd;
  border-radius: 6px;
  max-height: 60vh;     /* много строк */
}

.element-comment {
  padding: 6px 10px 10px;
  font-size: 11px;
  color: #cbd5e1;
  opacity: 0.9;
  border-top: 1px solid rgba(255,255,255,0.08);
  white-space: pre-wrap;
  word-break: break-word;
}

.element-comment:empty { display: none; }

/* Tooltip для шаблонов формул */
.template-item {
    position: relative;
}

.template-tooltip {
    position: fixed;
    background: #1a1a2e;
    border: 1px solid #4a90d9;
    border-radius: 6px;
    padding: 8px 12px;
    color: #e0e0e0;
    font-size: 12px;
    max-width: 280px;
    line-height: 1.4;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
    z-index: 10000;
    pointer-events: none;
    opacity: 0;
    transition: opacity 0.15s ease;
}

.template-tooltip.visible {
    opacity: 1;
}

.template-tooltip::before {
    content: '';
    position: absolute;
    top: -6px;
    left: 20px;
    border-left: 6px solid transparent;
    border-right: 6px solid transparent;
```

```
8949      border-bottom: 6px solid #4a90d9;
8950  }
8951
8952  .template-tooltip-title {
8953      font-weight: 600;
8954      color: #4a90d9;
8955      margin-bottom: 4px;
8956  }
```