```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

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]] = {}
```

```
 66
 67
 68  # ============================================================================
 69  # ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — ЗАГРУЗКА НАСТРОЕК
 70  # ============================================================================
 71
 72  def load_settings() -> Dict:
 73      """Загружает настройки из settings.json"""
 74      with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
 75          return json.load(f)
 76
 77
 78  def load_templates() -> Dict:
 79      """Загружает шаблоны формул"""
 80      if not os.path.exists(TEMPLATES_PATH):
 81          return {"templates": []}
 82      with open(TEMPLATES_PATH, "r", encoding="utf-8") as f:
 83          return json.load(f)
 84
 85
 86  # ============================================================================
 87  # ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ — СИГНАЛЫ
 88  # ============================================================================
 89
 90  def load_signals_from_folder(folder: str) -> List[Dict]:
 91      """Загружает описания сигналов из CSV файлов"""
 92      folder_abs = folder if os.path.isabs(folder) else
     os.path.normpath(os.path.join(BASE_DIR, folder))
 93      if not os.path.isdir(folder_abs):
 94          raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")
 95
 96      signals_map = {}
 97      for name in os.listdir(folder_abs):
 98          if not name.lower().endswith(".csv"):
 99              continue
100          path = os.path.join(folder_abs, name)
101          try:
102              try:
103                  df = pd.read_csv(path, sep=';')[['Tagname', 'Description',
     'Engineering Unit']]
104              except KeyError:
105                  df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
106              df = df.dropna(subset=['Tagname'])
107
108              for _, row in df.iterrows():
109                  tag = str(row['Tagname']).strip()
110                  desc = "" if pd.isna(row['Description']) else
     str(row['Description']).strip()
111                  try:
112                      unit = "" if pd.isna(row['Engineering Unit']) else
     str(row['Engineering Unit']).strip()
113                  except KeyError:
114                      unit = ""
115                  desc_full = ", ".join([x for x in [desc, unit] if x])
116
117                  if tag:
118                      signals_map[tag] = {
119                          "Tagname": tag,
120                          "Description": desc_full,
121                          "EngineeringUnit": unit
122                      }
123          except Exception as e:
124              print(f"[WARN] failed to read {path}: {e}")
125
126      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                 if cached_state == current_state:
254                     print(f"[OK] Signal index loaded from cache ({len(cached_index)}
    signals)")
255                     return cached_index
256                 else:
257                     print(f"[INFO] CSV files changed, rebuilding index...")
258             else:
259                 print(f"[INFO] Old cache format, rebuilding index...")
260         except Exception as e:
261             print(f"[WARN] Failed to load cached index: {e}")
262 
263     # Перестраиваем индекс
264     index = build_signal_index(folder)
265 
266     # Сохраняем с метаданными
267     try:
268         cache_data = {"index": index, "_folder_state": current_state}
269         with open(SIGNAL_INDEX_PATH, "wb") as f:
270             pickle.dump(cache_data, f)
271         print(f"[OK] Signal index cached with folder state")
272     except Exception as e:
273         print(f"[WARN] Failed to cache signal index: {e}")
274 
275     return index
276 
277 
278 def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
    pd.DataFrame]:
279     """Загружает только нужные сигналы из только нужных файлов"""
280     folder_abs = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
281 
282     signal_index = STATE.get("signal_index", {})
283     if not signal_index:
284         raise RuntimeError("Signal index not initialized")
285 
286     signal_names_set = set(signal_names)
287     found_signals = {}
288     files_to_load = set()
289 
290     for signal_name in signal_names_set:
291         if signal_name in signal_index:
292             files_to_load.update(signal_index[signal_name])
293 
294     print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)}
    files")
295 
296     for filepath in files_to_load:
297         try:
298             df = pd.read_csv(filepath, encoding="ISO-8859-2", sep=";")
299 
300             df["TIME"] = df["TIME"].str.replace(",", ".", regex=False)
301             df["TIME"] = df["TIME"].str.split(".").str[0]
302             combined = df["DATE"] + " " + df["TIME"]
303             df["datetime"] = pd.to_datetime(combined, format="%d.%m.%Y %H:%M:%S",
    errors="coerce")
304             df = df.dropna(subset=["datetime"])
305             df = df.drop(['DATE', 'TIME'], axis=1)
306             df = df.sort_values("datetime")
307 
308             available_columns = set(df.columns) & signal_names_set
309             for signal_name in available_columns:
310                 if signal_name not in found_signals:
311                     found_signals[signal_name] = df[["datetime", signal_name]].copy()
312                     found_signals[signal_name].columns = ["datetime", "value"]
```

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

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

    refresh_signals_cache()
    STATE["templates"] = load_templates()
    STATE["signal_index"] = load_signal_index(settings.get("signalArchiveFolder"))

    print(f"[OK] Loaded signals: {len(STATE['signals'])}")
    print(f"[OK] Signal index has {len(STATE['signal_index'])} unique signals")
    print(f"[OK] Loaded templates: {len(STATE['templates'].get('templates', []))}")


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

```
# API — НАСТРОЙКИ И СИГНАЛЫ
# ============================================================================

@app.get("/api/settings")
def api_settings():
    """Возвращает настройки приложения"""
    return STATE["settings"]


@app.get("/api/signals")
def api_signals(q: str = "", limit: int = 50):
    """Поиск сигналов по маске (* — wildcard)"""
    signals = STATE["signals"] or []

    if not q:
        result = {"items": signals[:limit], "total": len(signals)}
    else:
        import re
        escaped = re.escape(q).replace(r"\*", ".*")
        rx = re.compile("^" + escaped + "$", re.IGNORECASE)
        items = [s for s in signals if rx.match(s["Tagname"])]
        result = {"items": items[:max(1, min(limit, 500))], "total": len(items)}

    return JSONResponse(
        content=result,
        headers={
            "Cache-Control": "no-cache, no-store, must-revalidate",
            "Pragma": "no-cache",
            "Expires": "0"
        }
    )


@app.get("/api/formula-templates")
def api_formula_templates():
    """Возвращает шаблоны формул"""
    return STATE.get("templates") or {"templates": []}


# ============================================================================
# API — ПРОЕКТЫ
# ============================================================================

@app.get("/api/project/list")
def list_projects():
    """Список всех проектов"""
    folder = STATE["settings"].get("projectDataFolder")
    if not folder:
        raise HTTPException(status_code=500, detail="Project folder not configured")

    project_dir = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
    os.makedirs(project_dir, exist_ok=True)

    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", {})
```

```
570         projects.append({
571             "filename": fname,
572             "code": project_meta.get("code") or project_meta.get("tagname") or "",
573             "description": project_meta.get("description") or "",
574             "type": project_meta.get("type") or ""
575         })
576     return {"projects": projects}
577 
578 
579 @app.post("/api/project/save")
580 async def save_project(request: Request):
581     """Сохраняет проект"""
582     try:
583         data = await request.json()
584         filename = data.get("filename")
585         content = data.get("content")
586 
587         if not filename or not content:
588             raise HTTPException(status_code=400, detail="Filename and content are
    required")
589 
590         path = get_project_path(filename)
591 
592         with open(path, "w", encoding="utf-8") as f:
593             json.dump(content, f, indent=2)
594 
595         # Обновляем кэш сигналов
596         refresh_signals_cache()
597 
598         print(f"[OK] Project saved: {filename}, signals cache refreshed:
    {len(STATE['signals'])} signals")
599         return {"status": "ok", "message": f"Project saved to {filename}"}
600 
601     except HTTPException as e:
602         raise e
603     except Exception as e:
604         print(f"Error saving project: {e}")
605         raise HTTPException(status_code=500, detail="Internal server error during
    save")
606 
607 
608 @app.get("/api/project/load/{filename}")
609 def load_project(filename: str):
610     """Загружает проект"""
611     try:
612         path = get_project_path(filename)
613 
614         if not os.path.exists(path):
615             raise HTTPException(status_code=404, detail="Project not found")
616 
617         with open(path, "r", encoding="utf-8") as f:
618             content = json.load(f)
619 
620         return content
621 
622     except HTTPException as e:
623         raise e
624     except Exception as e:
625         print(f"Error loading project: {e}")
626         raise HTTPException(status_code=500, detail="Internal server error during
    load")
627 
628 
629 # ============================================================================
630 # API — ДАННЫЕ СИГНАЛОВ
```

```
631 # ============================================================================
632
633 @app.post("/api/signal-data")
634 async def api_signal_data(request: Request):
635     """Загружает данные сигналов из архива"""
636     try:
637         data = await request.json()
638         signal_names = data.get("signal_names", [])
639         output_format = data.get("format", "parquet")
640
641         if not signal_names:
642             raise HTTPException(status_code=400, detail="signal_names is required")
643
644         folder = STATE["settings"].get("signalArchiveFolder")
645         if not folder:
646             raise HTTPException(status_code=500, detail="signalArchiveFolder not
    configured")
647
648         signals_data = load_signal_data_optimized(signal_names, folder)
649
650         response = {
651             "found": list(signals_data.keys()),
652             "not_found": [s for s in signal_names if s not in signals_data],
653             "format": output_format
654         }
655
656         if not signals_data:
657             raise HTTPException(status_code=404, detail="No signals found")
658
659         if output_format == "parquet":
660             return await _export_parquet(signals_data, response)
661         else:
662             return await _export_json(signals_data, response)
663
664     except HTTPException as e:
665         raise e
666     except Exception as e:
667         print(f"Error in api_signal_data: {e}")
668         raise HTTPException(status_code=500, detail=str(e))
669
670
671 async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
672     """Экспортирует данные в Parquet"""
673     try:
674         with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
675             tmp_path = tmp.name
676
677         rows = []
678         for signal_name, df in signals_data.items():
679             df_copy = df.copy()
680             df_copy["signal_name"] = signal_name
681             rows.append(df_copy)
682
683         combined = pd.concat(rows, ignore_index=True)
684         combined.to_parquet(tmp_path, compression='snappy', index=False)
685
686         file_size = os.path.getsize(tmp_path)
687         print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size /
    1024 / 1024:.2f} MB")
688
689         return FileResponse(
690             tmp_path,
691             media_type="application/octet-stream",
692             filename="signal_data.parquet",
693             headers={"X-Signal-Meta": json.dumps(meta)}
```

```
        )
    except Exception as e:
        print(f"[ERROR] Parquet export failed: {e}")
        raise


async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
    """Экспортирует данные в JSON"""
    try:
        data_dict = {}
        for signal_name, df in signals_data.items():
            df_copy = df.copy()
            df_copy["datetime"] = df_copy["datetime"].astype(str)
            data_dict[signal_name] = df_copy.to_dict(orient="records")

        response_data = {**meta, "data": data_dict}
        return JSONResponse(response_data)
    except Exception as e:
        print(f"[ERROR] JSON export failed: {e}")
        raise


@app.post("/api/resolve-signals")
async def api_resolve_signals(request: Request):
    """Разворачивает зависимости сигналов (матрёшку)"""
    try:
        data = await request.json()
        signal_names = data.get("signals", [])

        print(f"[INFO] Resolving dependencies for signals: {signal_names}")

        base_signals, synthetic_signals = resolve_signal_dependencies(signal_names)
        computation_order = topological_sort_signals(synthetic_signals)

        print(f"[INFO] Base signals: {base_signals}")
        print(f"[INFO] Synthetic signals: {list(synthetic_signals.keys())}")
        print(f"[INFO] Computation order: {computation_order}")

        return {
            "base_signals": list(base_signals),
            "synthetic_signals": synthetic_signals,
            "computation_order": computation_order
        }

    except ValueError as ve:
        raise HTTPException(status_code=400, detail=str(ve))
    except Exception as e:
        print(f"[ERROR] resolve-signals failed: {e}")
        raise HTTPException(status_code=500, detail=str(e))


# ============================================================================
# API — ВИЗУАЛИЗАТОР
# ============================================================================

@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}")

        return VisualizerStateResponse(
            success=True,
            state=request.state,
            message="Состояние сохранено"
        )
    except Exception as e:
        print(f"[ERROR] save_visualizer_state failed: {e}")
        return VisualizerStateResponse(
            success=False,
```

```
            message=f"Ошибка сохранения: {str(e)}"
        )


@app.get("/api/visualize/get-state/{session_token}")
async def get_visualizer_state(session_token: str) -> VisualizerStateResponse:
    """Возвращает состояние визуализатора (вызывается из редактора)"""
    session = visualize_sessions.get(session_token)

    if session is None:
        return VisualizerStateResponse(
            success=False,
            message="Сессия не найдена"
        )

    state = session.get("visualizer_state")

    if state is None:
        return VisualizerStateResponse(
            success=False,
            message="Состояние визуализатора не сохранено"
        )

    return VisualizerStateResponse(
        success=True,
        state=state
    )


# ============================================================================
# СТАТИЧЕСКИЕ ФАЙЛЫ (ФРОНТЕНД)
# ============================================================================

WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")

#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)
                i += 1

        return "".join(result)

    # ---------- вспомогательные функции ----------
    def _ensure_series(value) -> pd.Series:
        if isinstance(value, pd.Series):
            return value.reindex(index)
        if isinstance(value, pd.DataFrame):
            if value.shape[1] == 1:
                return value.iloc[:, 0].reindex(index)
            raise CodeEvaluationError("Невозможно привести DataFrame с несколькими
```

```
колонками к Series.")
        if isinstance(value, (list, tuple, np.ndarray)):
            arr = np.asarray(value, dtype=float)
            if arr.size == 1:
                arr = np.full(len(index), arr.item())
            elif arr.shape[0] != len(index):
                return pd.Series(np.nan, index=index)
            return pd.Series(arr, index=index)
        if value is None or np.isscalar(value):
            return pd.Series(value, index=index)
        try:
            return pd.Series(value, index=index)
        except Exception as exc:
            raise CodeEvaluationError(f"Невозможно преобразовать значение '{value}' к Series.") from exc

    def _aggregate_nanfunc(func, args, empty_value=np.nan):
        if not args:
            return pd.Series(empty_value, index=index)
        stacked = np.vstack([_ensure_series(arg).values for arg in args])
        return pd.Series(func(stacked, axis=0), index=index)

    def GETPOINT(*_):
        if "GETPOINT" not in warnings:
            warnings.append("GETPOINT пока не поддержан — возвращается NaN.")
        return pd.Series(np.nan, index=index)

    def PREV(param_name):
        if param_name not in series_map:
            return pd.Series(np.nan, index=index)
        return series_map[param_name].shift(1)

    def _history_series(param_name):
        if param_name not in series_map:
            return None
        return series_map[param_name]

    def _history_window(period):
        try:
            minutes = int(period)
        except (TypeError, ValueError):
            return None
        if minutes <= 0:
            return None
        return f"{minutes}min"

    def _history_apply(param_name, period, fn):
        s = _history_series(param_name)
        window = _history_window(period)
        if s is None or window is None:
            return pd.Series(np.nan, index=index)
        return fn(s.rolling(window))

    HISTORYAVG = lambda n, p: _history_apply(n, p, lambda r: r.mean())
    HISTORYCOUNT = lambda n, p: _history_apply(n, p, lambda r: r.count())
    HISTORYSUM = lambda n, p: _history_apply(n, p, lambda r: r.sum())
    HISTORYMAX = lambda n, p: _history_apply(n, p, lambda r: r.max())
    HISTORYMIN = lambda n, p: _history_apply(n, p, lambda r: r.min())
    HISTORYDIFF = lambda n, p: _history_apply(n, p, lambda r: r.max() - r.min())

    def HISTORYGRADIENT(param_name, period):
        s = _history_series(param_name)
        window = _history_window(period)
        if s is None or window is None:
            return pd.Series(np.nan, index=index)
```

```
1013
1014          def slope(window_series: pd.Series):
1015              valid = window_series.dropna()
1016              if len(valid) < 2:
1017                  return np.nan
1018              x = valid.index.view(np.int64).astype(float) / 1e9
1019              y = valid.values.astype(float)
1020              x_mean = x.mean()
1021              y_mean = y.mean()
1022              denom = np.sum((x - x_mean) ** 2)
1023              if denom == 0:
1024                  return np.nan
1025              return np.sum((x - x_mean) * (y - y_mean)) / denom
1026
1027          return s.rolling(window).apply(slope, raw=False)
1028
1029      def ROUND(a, b=0):
1030          a_values = _ensure_series(a).values
1031          b_values = _ensure_series(b).values
1032          decimals = [
1033              0 if np.isnan(dec) else int(round(dec))
1034              for dec in b_values
1035          ]
1036          rounded = np.array([
1037              np.round(val, dec) if not np.isnan(val) else np.nan
1038              for val, dec in zip(a_values, decimals)
1039          ])
1040          return pd.Series(rounded, index=index)
1041
1042      # ---------- окружение eval ----------
1043      env = {
1044          "np": np,
1045          "ABS": lambda a: pd.Series(np.abs(_ensure_series(a).values), index=index),
1046          "EXP": lambda a: pd.Series(np.exp(_ensure_series(a).values), index=index),
1047          "POW": lambda a, b: pd.Series(np.power(_ensure_series(a).values,
      _ensure_series(b).values), index=index),
1048          "MIN": lambda *args: _aggregate_nanfunc(np.nanmin, args),
1049          "MAX": lambda *args: _aggregate_nanfunc(np.nanmax, args),
1050          "AVG": lambda *args: _aggregate_nanfunc(np.nanmean, args, empty_value=0.0),
1051          "MED": lambda *args: _aggregate_nanfunc(np.nanmedian, args),
1052          "ROUND": ROUND,
1053          "WHEN": lambda cond, t_val, f_val: pd.Series(
1054              np.where(_ensure_series(cond).astype(bool).values,
1055                       _ensure_series(t_val).values,
1056                       _ensure_series(f_val).values),
1057              index=index,
1058          ),
1059          "LOG": lambda x: pd.Series(np.log(_ensure_series(x).values), index=index),
1060          # Логарифм по основанию 10 (если нужен)
1061          "LOG10": lambda x: pd.Series(np.log10(_ensure_series(x).values), index=index),
1062          "PREV": PREV,
1063          "HISTORYAVG": HISTORYAVG,
1064          "HISTORYCOUNT": HISTORYCOUNT,
1065          "HISTORYSUM": HISTORYSUM,
1066          "HISTORYMAX": HISTORYMAX,
1067          "HISTORYMIN": HISTORYMIN,
1068          "HISTORYDIFF": HISTORYDIFF,
1069          "HISTORYGRADIENT": HISTORYGRADIENT,
1070          "GETPOINT": GETPOINT,
1071      }
1072
1073      for original_name, safe_name in safe_name_map.items():
1074          env[safe_name] = series_map[original_name]
1075
1076      def _normalize_expression(expr: str) -> str:
```

```
        expr = re.sub(r"\bAND\b", "&", expr, flags=re.IGNORECASE)
        expr = re.sub(r"\bOR\b", "|", expr, flags=re.IGNORECASE)
        expr = re.sub(r"\bNOT\b", "~", expr, flags=re.IGNORECASE)
        expr = expr.replace("<>", "!=")
        expr = re.sub(r"(?<![<>=!])=(?![<>=])", "==", expr)
        return expr

    normalized_code = _normalize_expression(code_str)
    normalized_code = _replace_signal_names(normalized_code)

    try:
        raw_result = eval(normalized_code, {"__builtins__": {}}, env)
    except Exception as exc:
        raise CodeEvaluationError(str(exc)) from exc

    result_series = _ensure_series(raw_result)
    result_series.name = result_series.name or "CODE_RESULT"
    return result_series, warnings

def compute_code_signal(
    code_str: str,
    df_all: pd.DataFrame,
    warn_callback=lambda msg: None,
) -> pd.Series:
    """
    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

    # 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()

    except Exception as exc:
        st.error(f"❌ Ошибка загрузки базовых сигналов: {exc}")
        return None


def resolve_and_load_all_signals(input_signals: List[str]) -> tuple[pd.DataFrame | None, List[str], List[str]]:
    if not input_signals:
        return None, [], []

    try:
        with st.spinner("🔍 Разворачиваем зависимости сигналов..."):
            resolve_resp = requests.post(
                f"{api_url}/api/resolve-signals",
                json={"signals": input_signals}
            )
            resolve_resp.raise_for_status()
            resolve_data = resolve_resp.json()

        base_signals = resolve_data.get("base_signals", [])
        synthetic_signals = resolve_data.get("synthetic_signals", {})
        computation_order = resolve_data.get("computation_order", [])

        project_signals = set(input_signals)
        dependency_signals = set()
        for syn_name, syn_data in synthetic_signals.items():
            if syn_name not in project_signals:
                dependency_signals.add(syn_name)
            for dep in syn_data.get("dependencies", []):
                if dep not in project_signals:
                    dependency_signals.add(dep)

        for bs in base_signals:
            if bs not in project_signals:
                dependency_signals.add(bs)

        st.session_state.signal_groups = {
            "project": project_signals,
            "dependencies": dependency_signals
        }

        st.info(f"📊 Сигналов проекта: {len(project_signals)} | Из зависимостей: {len(dependency_signals)}")

        if synthetic_signals:
            with st.expander("🔗 Граф зависимостей синтетических сигналов"):
                for syn_name in computation_order:
```

```
                    deps = synthetic_signals[syn_name].get("dependencies", [])
                    marker = "📌" if syn_name in project_signals else "🔗"
                    st.text(f"  {marker} {syn_name} ← {deps}")

        df_all = None
        found_signals = []
        not_found_signals = []

        if base_signals:
            with st.spinner(f"📥 Загружаем {len(base_signals)} базовых сигналов..."):
                df_all = load_base_signals_data(base_signals)
                if df_all is not None:
                    found_signals = list(df_all.columns)
                    not_found_signals = [s for s in base_signals if s not in df_all.columns]

        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
```

```
1395
1396
1397  def make_unique_name(base_name: str) -> str:
1398      existing = set()
1399      if st.session_state.signals_data is not None:
1400          existing |= set(st.session_state.signals_data.columns)
1401      existing |= set(st.session_state.derived_signals.keys())
1402      if base_name not in existing:
1403          return base_name
1404      idx = 2
1405      while f"{base_name}_{idx}" in existing:
1406          idx += 1
1407      return f"{base_name}_{idx}"
1408
1409
1410  # --- синтетический сигнал из CODE ---
1411  code_signal_name = st.session_state.code_signal_name
1412  df_for_code = get_all_signals_df(exclude={code_signal_name} if code_signal_name else
      None)
1413  code_key = (session_token, CODE)
1414
1415  already_have_series = (
1416      st.session_state.code_signal_name is not None
1417      and st.session_state.code_signal_name in st.session_state.derived_signals
1418  )
1419
1420  if CODE and df_for_code is not None:
1421      need_recalc = (st.session_state.get("code_key") != code_key) or (not
      already_have_series)
1422
1423      if need_recalc:
1424          try:
1425              synthetic_series = compute_code_signal(
1426                  CODE,
1427                  df_for_code,
1428                  warn_callback=lambda msg: st.warning(msg, icon="⚠️"),
1429              )
1430              target_name = code_signal_name or make_unique_name("CODE_RESULT")
1431              synthetic_series.name = target_name
1432
1433              st.session_state.derived_signals[target_name] = pd.DataFrame({target_name:
      synthetic_series})
1434              st.session_state.code_signal_name = target_name
1435              st.session_state.selected_signals.add(target_name)
1436
1437              st.session_state.code_key = code_key
1438              st.success(f"Синтетический сигнал обновлён: {target_name}")
1439          except Exception as exc:
1440              st.warning(f"Не удалось вычислить CODE: {exc}")
1441
1442  elif not CODE:
1443      if code_signal_name:
1444          st.session_state.derived_signals.pop(code_signal_name, None)
1445          st.session_state.selected_signals.discard(code_signal_name)
1446          st.session_state.code_signal_name = None
1447      st.session_state.code_key = None
1448
1449
1450  # === ЗАГРУЗКА СОХРАНЁННОГО СОСТОЯНИЯ (один раз) ===
1451  df_all_signals = get_all_signals_df()
1452
1453  if not st.session_state.state_loaded and INITIAL_VISUALIZER_STATE and df_all_signals
      is not None:
1454      available_signals = set(df_all_signals.columns.tolist())
1455
```

```
    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")

    st.session_state.state_loaded = True
    st.session_state.has_unsaved_changes = False


# === ФУНКЦИЯ СОХРАНЕНИЯ СОСТОЯНИЯ ===
def save_current_state():
    """Сохраняет текущее состояние на сервер"""
    if not session_token:
        st.error("Нет токена сессии для сохранения")
        return False

    state = create_visualizer_state(
        st.session_state.selected_signals,
        st.session_state.plot_areas
    )

    try:
        resp = requests.post(
            f"{api_url}/api/visualize/save-state",
            json={
                "session_token": session_token,
                "state": state
            }
        )
        resp.raise_for_status()
        result = resp.json()

        if result.get("success"):
            st.session_state.has_unsaved_changes = False
            return True
        else:
            st.error(f"Ошибка сохранения: {result.get('message')}")
            return False
    except Exception as e:
        st.error(f"Ошибка сохранения состояния: {e}")
        return False


# === SIDEBAR ===
with st.sidebar:
    st.header("Выбор сигналов")

    # НОВОЕ: Кнопка сохранения состояния
    if session_token:
        save_col1, save_col2 = st.columns([2, 1])
        with save_col1:
            if st.button("💾 Сохранить состояние", use_container_width=True):
```

```
                if save_current_state():
                    st.success("✅ Состояние сохранено!")
                    st.info("💡 Теперь сохраните проект в редакторе")
        with save_col2:
            if st.session_state.has_unsaved_changes:
                st.markdown("🔴 *Изменения*")
            else:
                st.markdown("🟢 *Сохранено*")
        st.divider()

    if df_all_signals is not None:
        available_signals = df_all_signals.columns.tolist()

        signal_groups = st.session_state.get("signal_groups", {
            "project": set(available_signals),
            "dependencies": set()
        })

        project_signals = [s for s in available_signals if s in
signal_groups.get("project", set())]
        dependency_signals = [s for s in available_signals if s in
signal_groups.get("dependencies", set())]

        if project_signals:
            st.subheader("📌 Сигналы проекта")
            for signal in project_signals:
                is_synthetic = signal in st.session_state.get("synthetic_computed",
{})
                label = f"⚙️ {signal}" if is_synthetic else signal

                checked = st.checkbox(
                    label,
                    value=(signal in st.session_state.selected_signals),
                    key=f"proj_{signal}"
                )
                if checked and signal not in st.session_state.selected_signals:
                    st.session_state.selected_signals.add(signal)
                    mark_unsaved()
                elif not checked and signal in st.session_state.selected_signals:
                    st.session_state.selected_signals.discard(signal)
                    mark_unsaved()

        if dependency_signals:
            st.divider()
            with st.expander(f"🔗 Из зависимостей ({len(dependency_signals)})",
expanded=False):
                for signal in dependency_signals:
                    is_synthetic = signal in
st.session_state.get("synthetic_computed", {})
                    label = f"⚙️ {signal}" if is_synthetic else signal

                    checked = st.checkbox(
                        label,
                        value=(signal in st.session_state.selected_signals),
                        key=f"dep_{signal}"
                    )
                    if checked and signal not in st.session_state.selected_signals:
                        st.session_state.selected_signals.add(signal)
                        mark_unsaved()
                    elif not checked and signal in st.session_state.selected_signals:
                        st.session_state.selected_signals.discard(signal)
                        mark_unsaved()

        st.divider()
        col1, col2 = st.columns(2)
```

```
1581          with col1:
1582              if st.button("✅ Все проекта"):
1583                  st.session_state.selected_signals.update(project_signals)
1584                  mark_unsaved()
1585                  st.rerun()
1586          with col2:
1587              if st.button("❌ Снять все"):
1588                  st.session_state.selected_signals.clear()
1589                  mark_unsaved()
1590                  st.rerun()
1591
1592          st.divider()
1593          st.subheader("Создать обрезанный сигнал")
1594
1595          base_df = st.session_state.signals_data
1596          if base_df is not None and not base_df.empty:
1597              base_choice = st.selectbox("Исходный сигнал", base_df.columns)
1598              series = base_df[base_choice].dropna()
1599              if not series.empty:
1600                  col1, col2 = st.columns(2)
1601                  with col1:
1602                      start_date = st.date_input(
1603                          "Начало",
1604                          value=series.index.min().date(),
1605                      )
1606                  with col2:
1607                      end_date = st.date_input(
1608                          "Конец",
1609                          value=series.index.max().date(),
1610                      )
1611
1612                  start_ts = pd.Timestamp(start_date)
1613                  end_ts = pd.Timestamp(end_date) + pd.Timedelta(days=1) - pd.Timedelta(
1614                      microseconds=1
1615                  )
1616
1617                  default_name = f"{base_choice}__{start_ts.date()}_{end_ts.date()}"
1618                  new_name = st.text_input("Имя нового сигнала", value=default_name)
1619
1620                  col3, col4 = st.columns(2)
1621                  if col3.button("Создать"):
1622                      name_unique = make_unique_name(new_name.strip())
1623                      cut_series = series[(series.index >= start_ts) & (series.index <=
      end_ts)]
1624                      if cut_series.empty:
1625                          st.warning("В выбранном диапазоне нет точек.")
1626                      else:
1627                          st.session_state.derived_signals[name_unique] = pd.DataFrame(
1628                              {name_unique: cut_series}
1629                          )
1630                          st.success(f"Создан обрезанный сигнал: {name_unique}")
1631                          st.rerun()
1632                  if col4.button("Очистить все обрезанные"):
1633                      st.session_state.derived_signals = {
1634                          k: v
1635                          for k, v in st.session_state.derived_signals.items()
1636                          if k == st.session_state.code_signal_name
1637                      }
1638                      st.session_state.selected_signals = {
1639                          sig
1640                          for sig in st.session_state.selected_signals
1641                          if (st.session_state.signals_data is not None and sig in
      st.session_state.signals_data.columns)
1642                          or sig == st.session_state.code_signal_name
1643                      }
```

```
                    st.rerun()

        if st.session_state.derived_signals:
            st.subheader("Удалить обрезанный/синтетический сигнал")
            derived_names = [name for name in st.session_state.derived_signals.keys()]
            delete_candidate = st.selectbox("Выберите", ["—"] + derived_names)
            if st.button("Удалить выбранный") and delete_candidate != "—":
                st.session_state.derived_signals.pop(delete_candidate, None)
                st.session_state.selected_signals.discard(delete_candidate)
                if delete_candidate == st.session_state.code_signal_name:
                    st.session_state.code_signal_name = None
                st.rerun()

        st.divider()
        st.subheader("Области построения")
        col_a, col_b = st.columns(2)
        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", []),
                key=f"signals_sel_{i}",
            )

            # Проверяем изменились ли сигналы
            if set(selected) != set(plot_area.get("signals", [])):
                mark_unsaved()
            st.session_state.plot_areas[i]["signals"] = selected

            if selected:
                df_plot = df_all_signals[selected].copy()
                df_plot_num = df_plot.apply(sanitize_numeric_column)

                valid_index = df_plot_num.dropna(how="all").index
                if len(valid_index) == 0:
                    st.warning("Нет числовых данных для выбранных сигналов.")
                else:
                    full_x_min = valid_index.min()
                    full_x_max = valid_index.max()

                    y_data = df_plot_num.values.flatten()
                    y_data = y_data[~np.isnan(y_data)]
                    full_y_min = float(y_data.min()) if len(y_data) > 0 else 0.0
                    full_y_max = float(y_data.max()) if len(y_data) > 0 else 1.0

                    y_padding = (full_y_max - full_y_min) * 0.05
                    full_y_min -= y_padding
                    full_y_max += y_padding

                    if plot_area.get('x_range') is None:
                        plot_area['x_range'] = [full_x_min, full_x_max]

                    if plot_area.get('y_range') is None:
                        plot_area['y_range'] = [full_y_min, full_y_max]

                    x_start_ts, x_end_ts = plot_area['x_range']
                    mask_visible = (valid_index >= x_start_ts) & (valid_index <=
    x_end_ts)
                    visible_index = valid_index[mask_visible]

                    if len(visible_index) == 0:
                        st.warning("В выбранном диапазоне X нет данных.")
                    else:
                        if plot_area.get('cursor_time') is None:
                            plot_area['cursor_time'] =
    visible_index[len(visible_index) // 2]

                        cursor_time = plot_area['cursor_time']
                        if cursor_time < x_start_ts or cursor_time > x_end_ts:
```

```
                            cursor_time = visible_index[len(visible_index) // 2]
                            plot_area['cursor_time'] = cursor_time

                        cursor_pos, _ = find_nearest_index_in_range(
                            visible_index, cursor_time, x_start_ts, x_end_ts
                        )

                        if st.session_state.global_cursor_time is not None:
                            global_cursor = st.session_state.global_cursor_time
                            if x_start_ts <= global_cursor <= x_end_ts:
                                cursor_pos, cursor_time = find_nearest_index_in_range(
                                    visible_index, global_cursor, x_start_ts, x_end_ts
                                )
                                plot_area['cursor_time'] = cursor_time

                        ts_idx = st.slider(
                            "📍 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)

                        with st.expander(f"📍 Добавить маркеры для графика #{plot_area['id']}"):
                            col_x, col_y = st.columns(2)
                            with col_x:
                                st.markdown("**Вертикальная линия (X)**")
                                x_date = st.date_input("Дата", value=ts.date(), key=f"x_date_{i}")
                                x_time = st.time_input("Время", value=ts.time(), key=f"x_time_{i}")
                                x_full = pd.Timestamp.combine(x_date, x_time)
                                if st.button("Добавить V-line", key=f"add_vline_{i}"):
                                    shapes.append({
                                        'type': 'vline',
                                        'x': x_full,
                                        'dash': 'dot',
                                        'color': 'blue'
                                    })
                                    plot_area['shapes'] = shapes
                                    mark_unsaved()
                                    st.success(f"Добавлена линия на {x_full}")
                                    st.rerun()

                            with col_y:
                                st.markdown("**Горизонтальная линия (Y)**")
                                y_value = st.number_input("Значение Y", value=0.0, key=f"y_val_{i}")
                                if st.button("Добавить H-line", key=f"add_hline_{i}"):
                                    shapes.append({
                                        'type': 'hline',
                                        'y': y_value,
                                        'dash': 'dash',
                                        'color': 'green'
                                    })
                                    plot_area['shapes'] = shapes
                                    mark_unsaved()
                                    st.success(f"Добавлена линия на Y={y_value}")
                                    st.rerun()

                            if shapes:
                                st.markdown("**Текущие маркеры:**")
                                for j, s in enumerate(shapes):
                                    if s['type'] == 'vline':
                                        st.text(f"  V-line: {s['x']} ({s['color']})")
                                    else:
                                        st.text(f"  H-line: Y={s['y']} ({s['color']})")
                                if st.button(f"🗑️ Очистить маркеры",
```

```
    key=f"clear_shapes_{i}"):
                                    plot_area['shapes'] = []
                                    mark_unsaved()
                                    st.rerun()

                        nearest =
    df_plot_num.reindex(df_plot_num.index.union([ts])).sort_index()
                        nearest = nearest.ffill().loc[ts]

                        st.markdown("**📊 Статистика:**")
                        stats_df = compute_stats_numeric(df_plot)
                        if stats_df.empty:
                            st.info("Нет данных для статистики.")
                        else:
                            stats_view = stats_df.copy()
                            stats_view["value"] = nearest.reindex(stats_view.index)
                            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
    }


def deserialize_plot_area(
    area_data: Dict[str, Any],
    available_signals: Set[str]
) -> Optional[Dict[str, Any]]:
    """
    Десериализует область графика.
    Фильтрует сигналы, которых нет в available_signals.
    """
    area_id = area_data.get('id', 1)

    # Фильтруем сигналы — оставляем только существующие
    raw_signals = area_data.get('signals', [])
    valid_signals = [s for s in raw_signals if s in available_signals]

    # Если после фильтрации не осталось сигналов — пропускаем область
    # (но можно оставить пустую, если хотите)

    # Десериализуем маркеры
    shapes = []
    for shape_data in area_data.get('shapes', []):
        shape = deserialize_shape(shape_data)
        if shape is not None:
            shapes.append(shape)

    return {
        'id': area_id,
        'signals': valid_signals,
        'shapes': shapes,
        'cursor_time': None,  # Пересчитывается при загрузке
        'x_range': None,      # Пересчитывается при загрузке
        'y_range': None       # Пересчитывается при загрузке
    }


def create_visualizer_state(
    selected_signals: Set[str],
    plot_areas: List[Dict[str, Any]]
) -> Dict[str, Any]:
    """
    Создаёт объект состояния визуализатора для сохранения.

    Args:
        selected_signals: Набор выбранных сигналов
        plot_areas: Список областей графиков

    Returns:
```

```
2081          Словарь, готовый для JSON сериализации
2082      """
2083      return {
2084          'version': STATE_VERSION,
2085          'selected_signals': sorted(list(selected_signals)),
2086          'plot_areas': [serialize_plot_area(pa) for pa in plot_areas]
2087      }
2088
2089
2090  def load_visualizer_state(
2091      state_data: Optional[Dict[str, Any]],
2092      available_signals: Set[str]
2093  ) -> tuple[Set[str], List[Dict[str, Any]], List[str]]:
2094      """
2095      Загружает и валидирует состояние визуализатора.
2096      Мягкая загрузка: если сигнала из состояния нет в проекте, он просто игнорируется
      без предупреждений.
2097      """
2098      warnings = []
2099
2100      # Если состояния нет — возвращаем пустые значения
2101      if state_data is None:
2102          return set(), [], []
2103
2104      # Проверяем версию
2105      version = state_data.get('version', 1)
2106      if version > STATE_VERSION:
2107          warnings.append(f"Версия состояния ({version}) новее текущей")
2108
2109      # --- Мягкая загрузка выбранных сигналов ---
2110      raw_selected = state_data.get('selected_signals', [])
2111      selected_signals = set()
2112
2113      for sig in raw_selected:
2114          # Добавляем сигнал только если он реально существует в проекте сейчас
2115          if sig in available_signals:
2116              selected_signals.add(sig)
2117          # Если сигнала нет — просто молчим (никаких missing_signals и warnings)
2118
2119      # --- Мягкая загрузка областей графиков ---
2120      plot_areas = []
2121      for area_data in state_data.get('plot_areas', []):
2122          area = deserialize_plot_area(area_data, available_signals)
2123          # Если область валидна (в ней есть сигналы или маркеры), добавляем её
2124          if area is not None:
2125              # Если в области были сигналы, которые удалили из проекта,
2126              # deserialize_plot_area их уже отфильтровал внутри.
2127              # Если в области вообще не осталось сигналов — мы всё равно её создадим,
2128              # но она будет пустой (пользователь сам решит, что с ней делать).
2129              plot_areas.append(area)
2130
2131      return selected_signals, plot_areas, warnings
2132
2133
2134  def state_to_json(state: Dict[str, Any]) -> str:
2135      """Конвертирует состояние в JSON строку"""
2136      return json.dumps(state, ensure_ascii=False, indent=2)
2137
2138
2139  def state_from_json(json_str: str) -> Optional[Dict[str, Any]]:
2140      """Парсит JSON строку в состояние"""
2141      try:
2142          return json.loads(json_str)
2143      except (json.JSONDecodeError, TypeError):
2144          return None
```

index.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Редактор логических схем</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div id="app">
        <div id="menu">
            <button class="menu-btn" id="btn-new">📄 Новый</button>
            <button class="menu-btn" id="btn-save">💾 Сохранить</button>
            <button class="menu-btn" id="btn-load">📂 Загрузить</button>
            <button class="menu-btn" id="btn-generate-code">🧩 Код</button>
            <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</button>
            <button class="menu-btn" id="btn-visualize">📈 Визуализировать</button>
            <div class="menu-separator"></div>
            <div class="zoom-controls">
                <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
                <span id="zoom-level">100%</span>
                <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
                <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
                <button class="menu-btn" id="btn-zoom-reset">1:1</button>
            </div>
            <input type="file" id="file-input" accept=".json">
        </div>

        <div id="main">
            <div id="palette">
                <h3>📦 Элементы</h3>
                <div class="palette-section">
                    <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>

                    <div class="palette-item" data-type="group">
                        <svg viewBox="0 0 60 40">
                        <rect x="6" y="8" width="48" height="24" rx="4"
                                fill="none" stroke="#6b7280" stroke-width="2" stroke-dasharray="4,2"/>
                        <text x="14" y="25" fill="#6b7280" font-size="10" font-weight="bold">GROUP</text>
                        </svg>
                        <div class="palette-item-name">Группа</div>
                    </div>
                    </div>

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

                    <div class="palette-item" data-type="input-signal">
                        <svg viewBox="0 0 60 40">
                            <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460" stroke="#4a90d9" stroke-width="2"/>
                            <text x="12" y="24" fill="#eee" font-size="10">IN</text>
                        </svg>
                        <div class="palette-item-name">Входной сигнал</div>
                    </div>
                </div>
                <div class="palette-section">
                    <div class="palette-section-title">ВЫХОДЫ</div>

```

```
                    <div class="palette-item" data-type="output">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="6"
fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
                            <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</
text>
                        </svg>
                        <div class="palette-item-name">Выход</div>
                    </div>
                </div>

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

                    <div class="palette-item" data-type="and">
                        <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="22" y="25" fill="#eee" font-size="12" font-
weight="bold">И</text>
                        </svg>
                        <div class="palette-item-name">И (AND)</div>
                    </div>

                    <div class="palette-item" data-type="or">
                        <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">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"/>
2261                             <text x="8" y="25" fill="#f59e0b" font-size="10" font-
      weight="bold">✓/✗</text>
2262                         </svg>
2263                         <div class="palette-item-name">Сепаратор</div>
2264                     </div>
2265                 </div>
2266
2267                 <div class="palette-section">
2268                     <div class="palette-section-title">ЗНАЧЕНИЯ</div>
2269
2270                     <div class="palette-item" data-type="const">
2271                         <svg viewBox="0 0 60 40">
2272                             <rect x="10" y="8" width="40" height="24" rx="3"
      fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
2273                             <text x="24" y="25" fill="#3b82f6" font-size="14" font-
      weight="bold">C</text>
2274                         </svg>
2275                         <div class="palette-item-name">Константа</div>
2276                     </div>
2277
2278                     <div class="palette-item" data-type="formula">
2279                         <svg viewBox="0 0 60 40">
2280                             <rect x="5" y="5" width="50" height="30" rx="5"
      fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
2281                             <text x="12" y="25" fill="#f59e0b" font-size="11" font-
      weight="bold">f(x)</text>
2282                         </svg>
2283                         <div class="palette-item-name">Формула</div>
2284                     </div>
2285                 </div>
2286
2287                 <div class="type-legend">
2288                     <div class="type-legend-item">
2289                         <div class="type-legend-dot logic"></div>
2290                         <span>Логический</span>
2291                     </div>
2292                     <div class="type-legend-item">
2293                         <div class="type-legend-dot number"></div>
2294                         <span>Числовой</span>
2295                     </div>
2296                 </div>
2297             </div>
2298
2299             <div id="workspace-container">
2300                 <svg id="connections-svg"></svg>
2301                 <div id="workspace"></div>
2302                   <!-- Прямоугольник для выделения элементов -->
2303                 <div id="selection-rect"></div>
2304
2305                 <!-- Мини-карта -->
2306                 <div id="minimap">
2307                     <div id="minimap-viewport"></div>
2308                     <canvas id="minimap-canvas"></canvas>
2309                 </div>
2310
2311                 <!-- Координаты и информация -->
2312                 <div id="viewport-info">
2313                     <span id="cursor-pos">X: 0, Y: 0</span>
2314                     <span id="selection-info"></span>
2315                 </div>
2316             </div>
2317         </div>
2318     </div>
2319
```

```
    <!-- 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>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="code-modal-close">Закрыть</
    button>
            </div>
        </div>
    </div>

    <div id="context-menu">
        <div class="context-item" id="ctx-properties">⚙️ Свойства</div>
        <div class="context-item" id="ctx-copy">📋 Копировать</div>
        <div class="context-item" id="ctx-delete">🗑️ Удалить</div>
    </div>

   <!-- Модули JavaScript -->
    <!-- Модули JavaScript -->
    <script src="js/config.js"></script>
    <script src="js/state.js"></script>
    <script src="js/utils.js"></script>
    <script src="js/viewport.js"></script>
    <script src="js/elements.js"></script>
    <script src="js/connections.js"></script>
    <script src="js/outputs.js"></script>  <!-- ← Этот файл опционален теперь -->
    <script src="js/modal.js"></script>
    <script src="js/project.js"></script>
    <script src="js/codegen_graph.js"></script>
    <script src="js/codegen_optimizer.js"></script>
    <script src="js/codegen.js"></script>
    <script src="js/settings.js"></script>

    <script src="js/app.js"></script>

    <div id="modal-project-list" class="modal hidden">
        <div class="modal__content modal__content--wide">
            <h2 class="modal__title">Выбор проекта</h2>

```

```
2382             <div class="project-list__toolbar">
2383             <input id="project-search" type="text" placeholder="Фильтр по имени или
      описанию…" />
2384             <button id="project-refresh" class="btn btn-secondary">Обновить</button>
2385             </div>
2386
2387             <div class="project-list__table-container">
2388             <table class="project-list__table">
2389                 <thead>
2390                 <tr>
2391                     <th>Файл</th>
2392                     <th>Tagname</th>
2393                     <th>Description</th>
2394                     <th>Тип</th>
2395                 </tr>
2396                 </thead>
2397                 <tbody id="project-list-body">
2398                 <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
2399                 </tbody>
2400             </table>
2401             </div>
2402
2403             <div class="modal__actions">
2404                 <button id="project-cancel" class="btn btn-secondary">Отмена</button>
2405                 <button id="project-load" class="btn btn-primary" disabled>Загрузить</
      button>
2406             </div>
2407         </div>
2408     </div>
2409
2410 </body>
2411 </html>
2412
2413 /**
2414  * Главный модуль приложения
2415  * app.js
2416  */
2417
2418 const App = {
2419     /**
2420      * Инициализация приложения
2421      */
2422     init() {
2423         Settings.init().catch(console.error);
2424         //Settings.init().then(() => {
2425         //    // если хочешь — можно обновить UI (например, статус “Сигналы
      загружены”)
2426         //    console.log('Settings loaded, signals:', Settings.signals.length);
2427         //    }).catch(err => console.error(err));
2428         //console.log('signals loaded:', Settings.signals.slice(0, 5));
2429         this.setupPaletteDragDrop();
2430         this.setupGlobalMouseHandlers();
2431         this.setupContextMenu();
2432         this.setupWorkspaceClick();
2433         this.setupOutputCounter();
2434         this.setupMultiSelection();
2435
2436         // Инициализация модулей
2437         Viewport.init();
2438         Modal.init();
2439         Project.init();
2440
2441         // Первоначальное определение выходов (только если модуль загружен)
2442         if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
2443             Outputs.updateOutputStatus();
```

```
2444          }
2445
2446          console.log('Logic Scheme Editor initialized');
2447          document.getElementById('btn-generate-code').addEventListener('click', () => {
2448              const code = CodeGen.generate();
2449              document.getElementById('code-output').value = code;
2450              document.getElementById('code-modal-overlay').style.display = 'flex';
2451          });
2452
2453          document.getElementById('code-modal-close').addEventListener('click', () => {
2454              document.getElementById('code-modal-overlay').style.display = 'none';
2455          });
2456          document.getElementById('btn-visualize').addEventListener('click', () => {
2457              App.openSignalVisualizer();
2458  });
2459      },
2460
2461  openSignalVisualizer() {
2462      try {
2463          // 1) Собираем входные сигналы
2464          const signals = Object.values(AppState.elements)
2465              .filter(e => e && e.type === 'input-signal')
2466              .map(e => e.props?.name || e.id);
2467          const uniqSignals = [...new Set(signals)];
2468
2469          if (uniqSignals.length === 0) {
2470              alert('Нет входных сигналов в схеме.');
2471              return;
2472          }
2473
2474          // 2) Генерируем код
2475          let codeStr = '';
2476          if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
      {
2477              codeStr = CodeGen.generate() || '';
2478          }
2479
2480          // 3) Определяем URL-ы динамически
2481          const currentHost = window.location.hostname;
2482          const apiPort = window.location.port || 8000;
2483          const visualizerPort = Settings.config?.visualizerPort || 8501;
2484
2485          const apiUrl = `http://${currentHost}:${apiPort}`;
2486          const visualizerBase = `http://${currentHost}:${visualizerPort}`;
2487
2488          console.log('API URL:', apiUrl);
2489          console.log('Visualizer URL:', visualizerBase);
2490
2491          // 4) Получаем сохранённое состояние визуализатора из проекта
2492          const visualizerState = AppState.project?.visualizer_state || null;
2493
2494          if (visualizerState) {
2495              console.log('Передаём сохранённое состояние визуализатора:',
      visualizerState);
2496          }
2497
2498          // 5) Создаём сессию на backend (с передачей состояния)
2499          fetch('/api/visualize/session', {
2500              method: 'POST',
2501              headers: { 'Content-Type': 'application/json' },
2502              body: JSON.stringify({
2503                  signals: uniqSignals,
2504                  code: codeStr,
2505                  visualizer_state: visualizerState  // НОВОЕ: передаём состояние
2506              })
```

```
2507          })
2508          .then(r => {
2509              if (!r.ok) throw new Error('Failed to create visualize session');
2510              return r.json();
2511          })
2512          .then(data => {
2513              const token = data.token;
2514
2515              // НОВОЕ: сохраняем токен для последующего получения состояния
2516              AppState.currentVisualizerToken = token;
2517
2518              const params = new URLSearchParams();
2519              params.set('session', token);
2520              params.set('api_url', apiUrl);
2521
2522              const visualizerUrl = `${visualizerBase}/?${params.toString()}`;
2523              console.log('Opening visualizer:', visualizerUrl);
2524              window.open(visualizerUrl, '_blank');
2525          })
2526          .catch(err => {
2527              console.error(err);
2528              alert('Не удалось открыть визуализатор: ' + err.message);
2529          });
2530
2531      } catch (e) {
2532          console.error(e);
2533          alert('Ошибка при подготовке визуализации: ' + e.message);
2534      }
2535  },
2536
2537  /**
2538   * Получает состояние визуализатора с сервера
2539   * Вызывается перед сохранением проекта
2540   */
2541  async fetchVisualizerState() {
2542      if (!AppState.currentVisualizerToken) {
2543          console.log('Нет активной сессии визуализатора');
2544          return null;
2545      }
2546
2547      try {
2548          const response = await fetch(`/api/visualize/get-state/$
      {AppState.currentVisualizerToken}`);
2549
2550          if (!response.ok) {
2551              console.warn('Не удалось получить состояние визуализатора:',
      response.status);
2552              return null;
2553          }
2554
2555          const result = await response.json();
2556
2557          if (result.success && result.state) {
2558              console.log('Получено состояние визуализатора:', result.state);
2559              return result.state;
2560          }
2561
2562          return null;
2563      } catch (error) {
2564          console.error('Ошибка получения состояния визуализатора:', error);
2565          return null;
2566      }
2567  },
2568
2569      /**
```

```
2570        * Отмена состояния drag из палитры (helper)
2571        */
2572       cancelPaletteDrag() {
2573           if (AppState.dragPreview) {
2574               try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
2575               AppState.dragPreview = null;
2576           }
2577           AppState.isDraggingFromPalette = false;
2578           AppState.dragType = null;
2579       },
2580
2581       /**
2582        * Настройка счётчика выходов в меню
2583        */
2584       setupOutputCounter() {
2585           // Не создавать повторно, если уже есть
2586           if (document.getElementById('btn-outputs')) return;
2587
2588           const menu = document.getElementById('menu');
2589
2590           // Создаём кнопку с счётчиком выходов
2591           const outputBtn = document.createElement('button');
2592           outputBtn.className = 'menu-btn output-btn';
2593           outputBtn.id = 'btn-outputs';
2594           outputBtn.innerHTML = `
2595               📤 Выходы
2596               <span id="output-counter" class="output-counter">0</span>
2597           `;
2598
2599           // Вставляем после кнопки свойств проекта
2600           const projectBtn = document.getElementById('btn-project-settings');
2601           if (projectBtn) {
2602               projectBtn.after(outputBtn);
2603           } else {
2604               menu.appendChild(outputBtn);
2605           }
2606
2607           outputBtn.addEventListener('click', () => {
2608               Modal.showProjectPropertiesModal();
2609           });
2610       },
2611
2612       /**
2613        * Настройка drag & drop из палитры
2614        */
2615       setupPaletteDragDrop() {
2616           document.querySelectorAll('.palette-item').forEach(item => {
2617               item.addEventListener('mousedown', (e) => {
2618                   // Только левая кнопка мыши должна запускать drag из палитры
2619                   if (e.button !== 0) return;
2620                   e.preventDefault();
2621
2622                   AppState.isDraggingFromPalette = true;
2623                   AppState.dragType = item.dataset.type;
2624
2625                   AppState.dragPreview = document.createElement('div');
2626                   AppState.dragPreview.className = 'drag-preview';
2627                   AppState.dragPreview.textContent =
     ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
2628                   AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2629                   AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2630                   document.body.appendChild(AppState.dragPreview);
2631               });
2632           });
2633       },
```

```
2634
2635      /**
2636       * Глобальные обработчики мыши
2637       */
2638  /**
2639   * Глобальные обработчики мыши
2640   */
2641  setupGlobalMouseHandlers() {
2642      document.addEventListener('mousemove', (e) => {
2643          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
2644              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2645              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2646          }
2647          if (AppState.resizing) {
2648              Elements.handleResize(e);
2649              return;
2650          }
2651          if (AppState.draggingElement) {
2652              Elements.handleDrag(e);
2653          }
2654          if (AppState.tempLine && AppState.connectingFrom) {
2655              Connections.drawTempConnection(e);
2656          }
2657      });
2658
2659      document.addEventListener('mouseup', (e) => {
2660          if (AppState.resizing) {
2661              AppState.resizing = null;
2662              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
2663          }
2664
2665          if (AppState.isDraggingFromPalette) {
2666              try {
2667                  if (AppState.dragPreview) {
2668                      AppState.dragPreview.remove();
2669                      AppState.dragPreview = null;
2670                  }
2671
2672                  const container = document.getElementById('workspace-container');
2673                  const rect = container.getBoundingClientRect();
2674
2675                  if (e.clientX >= rect.left && e.clientX <= rect.right &&
2676                      e.clientY >= rect.top && e.clientY <= rect.bottom) {
2677
2678                      const canvasPos = screenToCanvas(e.clientX, e.clientY);
2679                      const config = ELEMENT_TYPES[AppState.dragType];
2680                      if (config) {
2681                          const defaultWidth = config.minWidth || 120;
2682                          const defaultHeight = config.minHeight || 60;
2683
2684                          // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
      обработать
2685                          const newElement = Elements.addElement(
2686                              AppState.dragType,
2687                              canvasPos.x - defaultWidth / 2,
2688                              canvasPos.y - defaultHeight / 2
2689                          );
2690
2691                          if (newElement && typeof Outputs !== 'undefined') {
2692                              Outputs.updateOutputStatus();
2693                          }
2694                      } else {
2695                          console.error('Неизвестный тип элемента при drop:',
      AppState.dragType);
2696                      }
```

```
                }
            } finally {
                App.cancelPaletteDrag();
            }
        }

        if (AppState.draggingElement) {
            AppState.draggingElement = null;
        }

        Connections.clearConnectionState();
    });

    document.addEventListener('keydown', (e) => {
    // 1. Проверяем, не печатает ли пользователь текст
        const target = e.target;
        const isInput = target.tagName === 'INPUT' ||
                        target.tagName === 'TEXTAREA' ||
                        target.isContentEditable;

        if (isInput) return; // Если печатаем - игнорируем глобальные хоткеи

        // 2. Проверяем, не открыто ли модальное окно
        const modal = document.getElementById('modal-overlay');
        const projectModal = document.getElementById('project-modal-overlay');
        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);
            }
        });

```

```
2762          document.getElementById('ctx-delete').addEventListener('click', () => {
2763              document.getElementById('context-menu').style.display = 'none';
2764
2765              // Используем новую функцию для удаления всех выделенных
2766              Elements.deleteSelectedElements();
2767
2768              if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
2769                  Outputs.updateOutputStatus();
2770              }
2771          });
2772          document.getElementById('ctx-copy').addEventListener('click', () => {
2773              document.getElementById('context-menu').style.display = 'none';
2774              Elements.copySelectedElements();
2775          });
2776      },
2777
2778      /**
2779       * Клик по рабочей области
2780       */
2781      // app.js
2782      // app.js
2783      setupWorkspaceClick() {
2784          const container = document.getElementById('workspace-container');
2785
2786          container.addEventListener('click', (e) => {
2787              // Если мы только что закончили тянуть РАМКУ (реальное выделение), не
      сбрасываем
2788              if (AppState.marqueeJustEnded) return;
2789
2790              // Если кликнули ЛЕВОЙ кнопкой мыши НЕ по элементу и НЕ по порту
2791              if (e.button === 0 && !e.target.closest('.element') && !
      e.target.closest('.port')) {
2792                  Elements.deselectAll();
2793              }
2794          });
2795      },
2796      /**
2797   * --- Выделение рамкой и множественное перемещение ---
2798   */
2799      // app.js
2800      setupMultiSelection() {
2801          const container = document.getElementById('workspace-container');
2802          const rectEl = document.getElementById('selection-rect');
2803
2804          container.addEventListener('mousedown', (e) => {
2805              // РАМКА: только ЛЕВАЯ кнопка (0) и клик НЕ по элементу
2806              if (e.button !== 0 || e.target.closest('.element') ||
      e.target.closest('#minimap')) return;
2807
2808              const pos = screenToCanvas(e.clientX, e.clientY);
2809              AppState.multiSelecting = true;
2810              AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
      pos.y, w: 0, h: 0 };
2811
2812              rectEl.style.left = e.clientX + 'px';
2813              rectEl.style.top = e.clientY + 'px';
2814              rectEl.style.width = '0px';
2815              rectEl.style.height = '0px';
2816              rectEl.style.display = 'block';
2817          });
2818
2819          document.addEventListener('mousemove', (e) => {
2820              if (!AppState.multiSelecting) return;
2821
2822              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.
```

```
2885        */
2886   /**
2887    * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
2888    */
2889       collectAllCond(graph) {
2890           if (!graph) return null;
2891
2892           let c = null;
2893           const elem = graph.elem;
2894
2895           // 1. Собираем условия через cond-порт (как было)
2896           if (graph.condInput) {
2897               const condConn = graph.condInput.conn;
2898               const fromGraph = graph.condInput.fromGraph;
2899               const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
2900               c = oneCond;
2901
2902               // Рекурсивно идём вверх по cond-цепочке
2903               const upCond = this.collectAllCond(fromGraph);
2904               if (upCond) {
2905                   c = c ? Optimizer.And(c, upCond) : upCond;
2906               }
2907           }
2908
2909           // 2. НОВОЕ: если это separator — учитываем контекст его входа
2910           if (elem.type === 'separator' && graph.inputs.length > 0) {
2911               const inputGraph = graph.inputs[0].fromGraph;
2912               const inputContext = this.collectAllCond(inputGraph);
2913               if (inputContext) {
2914                   c = c ? Optimizer.And(c, inputContext) : inputContext;
2915               }
2916           }
2917
2918           return c;
2919       },
2920       buildDependencyGraph(elementId) {
2921           const graph = {
2922               nodeId: elementId,
2923               elem: AppState.elements[elementId],
2924               inputs: [],
2925               condInput: null,
2926           };
2927
2928           if (!graph.elem) return null;
2929
2930           const inConns = AppState.connections
2931           .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
2932           .sort((a, b) => {
2933               const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
2934               const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
2935               return ai - bi;
2936           });
2937
2938           inConns.forEach(conn => {
2939           graph.inputs.push({
2940               conn,
2941               fromGraph: this.buildDependencyGraph(conn.fromElement)
2942           });
2943           });
2944
2945           const condConn = AppState.connections.find(c =>
2946               c.toElement === elementId && c.toPort === 'cond-0'
2947           );
2948           if (condConn) {
2949               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);
            }

            case 'and': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.And(result, inLogic) : inLogic;
                }
                return result || Optimizer.TrueCond;
            }

            case 'or': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.Or(result, inLogic) : inLogic;
                }
                return result || Optimizer.FalseCond;
            }

            case 'not': {
                const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
                return Optimizer.Not(inLogic);
            }

            case 'separator': {
                return this.evalLogic(graph.inputs[0]?.fromGraph);
            }

            default:
                return Optimizer.TrueCond;
        }
    },

    /**
     * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
     */
    evalValue(graph) {
        if (!graph) return Optimizer.Const(0);
        const elem = graph.elem;
```

```
3015
3016          switch (elem.type) {
3017              case 'input-signal':
3018                  return Optimizer.Var(elem.props.name || graph.nodeId);
3019
3020              case 'const':
3021                  return Optimizer.Const(Number(elem.props.value) || 0);
3022
3023              case 'formula': {
3024                  const expr = this.buildFormulaExpr(elem);
3025                  return Optimizer.Var(expr);
3026              }
3027
3028              case 'separator':
3029                  return this.evalValue(graph.inputs[0]?.fromGraph);
3030
3031              default:
3032                  return Optimizer.Const(0);
3033          }
3034      },
3035
3036      // js/codegen_graph.js
3037
3038      /**
3039       * Рекурсивно собрать полный контекст условий для элемента
3040       * через всю цепочку cond-портов вверх
3041       */
3042    // В codegen_graph.js, в evalFullContext добавь:
3043
3044      evalFullContext(graph) {
3045          if (!graph) return null;
3046
3047          let context = null;
3048          const elem = graph.elem;
3049
3050          console.log(`evalFullContext для ${elem.id} (${elem.type})`);
3051
3052          // 1. Если сам элемент имеет cond-порт — собираем его условие
3053          if (graph.condInput) {
3054              const condConn = graph.condInput.conn;
3055              console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
      {condConn.fromPort}`);
3056
3057              const condLogic = this.evalConditionFromPort(
3058                  graph.condInput.fromGraph,
3059                  condConn.fromPort
3060              );
3061              console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
3062              context = condLogic;
3063
3064              // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
3065              const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
3066              if (upstreamContext) {
3067                  console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
3068                  context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
3069              }
3070          } else {
3071              console.log(`  → нет cond-0`);
3072          }
3073
3074          console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
3075          return context;
```

```
3076      },
3077
3078      /**
3079       * Получить УСЛОВИЕ для cond-порта элемента
3080       * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
3081       */
3082      evalConditionFromPort(graph, fromPort) {
3083          if (!graph) return null;
3084          const elem = graph.elem;
3085
3086          // Если это сепаратор — вычисляем его вход и применяем ветвление
3087          if (elem.type === 'separator') {
3088              const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
3089
3090              if (fromPort === 'out-0') {
3091                  return inputLogic;
3092              } else if (fromPort === 'out-1') {
3093                  return Optimizer.Not(inputLogic);
3094              }
3095          }
3096
3097          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
3098          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
3099              return this.evalLogic(graph);
3100          }
3101
3102          return null;
3103      },
3104
3105      /**
3106       * Главная функция: получить {cond, expr} для элемента
3107       */
3108      evalGraphValue(graph) {
3109
3110          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
3111
3112          const elem = graph.elem;
3113          //let cond = null;
3114
3115          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
3116          let cond = this.collectAllCond(graph);
3117
3118          let expr = null;
3119
3120          switch (elem.type) {
3121              case 'input-signal':
3122                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
3123                  break;
3124
3125              case 'const':
3126                  expr = Optimizer.Const(Number(elem.props.value) || 0);
3127                  break;
3128
3129              case 'formula': {
3130                  // Для формулы также собираем условия от всех входных элементов
3131                  const inputConds = graph.inputs.map(inp => {
3132                      const inResult = this.evalGraphValue(inp.fromGraph);
3133                      return inResult.cond;
3134                  }).filter(c => c);
3135
3136                  // Объединяем cond-порт с условиями от входов
3137                  for (const inCond of inputConds) {
3138                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
3139                  }
```

```
3140
3141                 expr = Optimizer.Var(this.buildFormulaExpr(elem));
3142                 break;
3143             }
3144
3145             case 'separator':
3146                 // Сепаратор — просто пробрасываем значение дальше
3147                 return this.evalGraphValue(graph.inputs[0]?.fromGraph);
3148
3149             // Логические элементы не должны здесь быть
3150             case 'and':
3151             case 'or':
3152             case 'not':
3153             case 'if':
3154             default:
3155                 expr = Optimizer.Const(0);
3156         }
3157
3158         return { cond, expr };
3159     },
3160
3161     buildIfLogic(leftVal, op, rightVal) {
3162         const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
3163         const rightName = rightVal.type === 'var' ? rightVal.name :
     String(rightVal.n);
3164
3165         const leftZero = leftVal.type === 'const' && leftVal.n === 0;
3166         const rightZero = rightVal.type === 'const' && rightVal.n === 0;
3167
3168         switch (op) {
3169             case '=':
3170                 if (rightZero) return Optimizer.Eq0(leftName);
3171                 if (leftZero) return Optimizer.Eq0(rightName);
3172                 return Optimizer.Cmp(leftName, '=', rightName);
3173             case '!=':
3174                 if (rightZero) return Optimizer.Ne0(leftName);
3175                 if (leftZero) return Optimizer.Ne0(rightName);
3176                 return Optimizer.Cmp(leftName, '!=', rightName);
3177             case '>':
3178             case '<':
3179             case '>=':
3180             case '<=':
3181                 return Optimizer.Cmp(leftName, op, rightName);
3182             default:
3183                 return Optimizer.TrueCond;
3184         }
3185     },
3186
3187
3188     buildFormulaExpr(elem) {
3189         let result = elem.props.expression || '0';
3190
3191         // 1) Сначала раскрываем шаблоны (h и др.)
3192         const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
3193             ? Settings.getTemplatesMap()
3194             : null;
3195         result = expandFormulaTemplates(result, map);
3196
3197         // 2) Потом раскрываем ссылки на формулы
3198         const formulaRefs = result.match(/formula[_-]\d+/g) || [];
3199         for (const ref of formulaRefs) {
3200             const refElem = AppState.elements[ref];
3201             if (refElem && refElem.type === 'formula') {
3202                 const refExpr = this.buildFormulaExpr(refElem);
3203                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
```

```
            }
        }

        return result;
    }
};

window.CodeGenGraph = CodeGenGraph;

// js/codegen_optimizer.js

let _depth = 0;
const MAX_DEPTH = 200;

// === Конструкторы ===
function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
function And(a, b) {
    if (!a) return b;
    if (!b) return a;
    return { kind: 'cond', type: 'and', a, b };
}
function Or(a, b) {
    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;
        }
    }

    // Преобразует cmp-условие в интервал по одной переменной
    // Возвращает { varName, min, minInc, max, maxInc } или null
    function cmpToInterval(c) {
        if (!c || c.type !== 'cmp') return null;

        const lNum = parseNumberLiteral(c.l);
        const rNum = parseNumberLiteral(c.r);

        let varName, op, val;

        if (lNum == null && rNum != null) {
            // var OP const
            varName = c.l;
            op = c.op;
            val = rNum;
        } else if (lNum != null && rNum == null) {
            // const OP var  ->  var (OP') const
            varName = c.r;
            op = reverseOp(c.op);
            if (!op) return null;
            val = lNum;
        } else {
            // Либо обе стороны числа, либо обе не числа — не трогаем
            return null;
        }

        // Интересуют только упорядочивающие операторы
        switch (op) {
            case '<':
            case '<=':
            case '>':
            case '>=':
            case '=':
                break;
            default:
                return null;
        }

        let min = Number.NEGATIVE_INFINITY;
        let max = Number.POSITIVE_INFINITY;
```

```
3334      let minInc = false;
3335      let maxInc = false;
3336
3337      switch (op) {
3338          case '<':
3339              max = val; maxInc = false; break;
3340          case '<=':
3341              max = val; maxInc = true; break;
3342          case '>':
3343              min = val; minInc = false; break;
3344          case '>=':
3345              min = val; minInc = true; break;
3346          case '=':
3347              min = val; minInc = true;
3348              max = val; maxInc = true;
3349              break;
3350      }
3351
3352      return { varName, min, minInc, max, maxInc };
3353  }
3354
3355  function intervalSubset(a, b) {
3356      if (!a || !b) return false;
3357
3358      // Нижняя граница: a.min >= b.min
3359      const amin = a.min, bmin = b.min;
3360      if (amin === Number.NEGATIVE_INFINITY) {
3361          if (bmin !== Number.NEGATIVE_INFINITY) return false;
3362          // оба -∞ — ок
3363      } else if (bmin === Number.NEGATIVE_INFINITY) {
3364          // b начинается “раньше” — ок
3365      } else if (amin > bmin) {
3366          // a стартует правее b — ок
3367      } else if (amin < bmin) {
3368          // a захватывает меньшее значение — не подмножество
3369          return false;
3370      } else {
3371          // amin === bmin
3372          if (a.minInc && !b.minInc) {
3373              // a включает границу, а b — нет → в a есть точка, не входящая в b
3374              return false;
3375          }
3376      }
3377
3378      // Верхняя граница: a.max <= b.max
3379      const amax = a.max, bmax = b.max;
3380      if (amax === Number.POSITIVE_INFINITY) {
3381          if (bmax !== Number.POSITIVE_INFINITY) return false;
3382      } else if (bmax === Number.POSITIVE_INFINITY) {
3383          // b идёт дальше — ок
3384      } else if (amax < bmax) {
3385          // a заканчивается раньше — ок
3386      } else if (amax > bmax) {
3387          return false;
3388      } else {
3389          // amax === bmax
3390          if (a.maxInc && !b.maxInc) {
3391              return false;
3392          }
3393      }
3394
3395      return true;
3396  }
3397
3398  // Удаляет избыточные cmp-условия в массиве атомов
```

```
3399  // mode: 'and' | 'or'
3400  function removeRedundantCmpAtoms(atoms, mode) {
3401      if (!atoms || atoms.length < 2) return atoms;
3402
3403      const keep = new Array(atoms.length).fill(true);
3404
3405      for (let i = 0; i < atoms.length; i++) {
3406          if (!keep[i]) continue;
3407          const a = atoms[i];
3408          if (!a || a.type !== 'cmp') continue;
3409
3410          for (let j = 0; j < atoms.length; j++) {
3411              if (i === j || !keep[j]) continue;
3412              const b = atoms[j];
3413              if (!b || b.type !== 'cmp') continue;
3414
3415              const rel = cmpImplicationRelation(a, b);
3416              if (!rel) continue;
3417
3418              if (rel === 'a_in_b') {
3419                  if (mode === 'or') {
3420                      // A ⊆ B → A OR B = B  → A лишнее
3421                      keep[i] = false;
3422                      break;
3423                  } else if (mode === 'and') {
3424                      // A ⊆ B → A AND B = A → B лишнее
3425                      keep[j] = false;
3426                  }
3427              } else if (rel === 'b_in_a') {
3428                  if (mode === 'or') {
3429                      // B ⊆ A → A OR B = A → B лишнее
3430                      keep[j] = false;
3431                  } else if (mode === 'and') {
3432                      // B ⊆ A → A AND B = B → A лишнее
3433                      keep[i] = false;
3434                      break;
3435                  }
3436              }
3437          }
3438      }
3439
3440      return atoms.filter((_, idx) => keep[idx]);
3441  }
3442
3443  // Отношение между двумя cmp-условиями через интервалы
3444  // 'a_in_b'  — A ⊆ B
3445  // 'b_in_a'  — B ⊆ A
3446  // 'equal'   — одинаковые интервалы (редко используем)
3447  // null      — не можем определить
3448  function cmpImplicationRelation(c1, c2) {
3449      const i1 = cmpToInterval(c1);
3450      const i2 = cmpToInterval(c2);
3451      if (!i1 || !i2) return null;
3452      if (i1.varName !== i2.varName) return null;
3453
3454      const aInB = intervalSubset(i1, i2);
3455      const bInA = intervalSubset(i2, i1);
3456
3457      if (aInB && bInA) return 'equal';
3458      if (aInB) return 'a_in_b';
3459      if (bInA) return 'b_in_a';
3460      return null;
3461  }
3462
3463  // Разворот оператора при перестановке аргументов (левый/правый)
```

```
3464  function reverseOp(op) {
3465      switch (op) {
3466          case '<':  return '>';
3467          case '>':  return '<';
3468          case '<=': return '>=';
3469          case '>=': return '<=';
3470          case '=':
3471          case '!=':
3472              return op;
3473          default:
3474              return null;
3475      }
3476  }
3477
3478  // Аккуратный парсер числового литерала.
3479  // Возвращает число или null, если строка не чисто числовая.
3480  function parseNumberLiteral(s) {
3481      if (typeof s !== 'string') return null;
3482      const trimmed = s.trim().replace(',', '.');
3483
3484      // Только простые вещи: -123, 45, 3.14
3485      if (!/^-?\d+(\.\d+)?$/.test(trimmed)) return null;
3486
3487      const n = Number(trimmed);
3488      return Number.isFinite(n) ? n : null;
3489  }
3490
3491
3492  function negateAtomKey(key) {
3493      if (!key) return null;
3494      if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
3495      if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
3496      if (key.startsWith('cmp:')) {
3497          const parts = key.slice(4).split(':');
3498          if (parts.length === 3) {
3499              const negOp = negateOp(parts[1]);
3500              if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
3501          }
3502      }
3503      return null;
3504  }
3505
3506  function isNegation(a, b) {
3507      if (!a || !b) return false;
3508      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
3509      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
3510      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
3511          return a.op === negateOp(b.op);
3512      }
3513      if (a.type === 'not' && condEq(a.x, b)) return true;
3514      if (b.type === 'not' && condEq(b.x, a)) return true;
3515      return false;
3516  }
3517
3518  function isAtomCond(t) {
3519      return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
3520  }
3521
3522  function pruneOrByContext(orTerm, contextAtoms) {
3523      const branches = flattenOr(orTerm);
3524      const kept = [];
3525
3526      for (const br of branches) {
3527          let contradicts = false;
3528
```

```
3529          for (const ctx of contextAtoms) {
3530              if (isNegation(br, ctx)) {
3531                  contradicts = true;
3532                  break;
3533              }
3534          }
3535
3536          if (!contradicts) kept.push(br);
3537      }
3538
3539      if (kept.length === 0) return FalseCond;
3540      if (kept.length === 1) return kept[0];
3541      return buildOr(kept);
3542  }
3543
3544  function condNegationEq(a, b) {
3545      if (!a || !b) return false;
3546
3547      // Простая проверка: a == NOT(b)
3548      if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;
3549
3550      // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
3551      // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
3552      if (a.type === 'or' && b.type === 'and') {
3553          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
3554                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3555      }
3556      // Симметрично
3557      if (a.type === 'and' && b.type === 'or') {
3558          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
3559                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3560      }
3561
3562      // Проверка атомов: (X = 0) vs (X != 0)
3563      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
3564      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
3565
3566      // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
3567      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
3568          return a.op === negateOp(b.op);
3569      }
3570
3571      return false;
3572  }
3573
3574
3575
3576  function condEq(a, b) {
3577      if (a === b) return true;
3578      if (!a || !b) return false;
3579      if (a.type !== b.type) return false;
3580
3581      switch (a.type) {
3582          case 'eq0':
3583          case 'ne0':
3584              return a.v === b.v;
3585          case 'cmp':
3586              return a.l === b.l && a.op === b.op && a.r === b.r;
3587          case 'true':
3588          case 'false':
3589              return true;
3590          case 'not':
3591              return condEq(a.x, b.x);
3592          case 'and':
3593          case 'or':
```

```
            return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
                   (condEq(a.a, b.b) && condEq(a.b, b.a));
        default:
            return false;
    }
}

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

function flattenOr(c) {
    if (!c) return [];
    if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
    return [c];
}

function buildAnd(terms) {
    if (terms.length === 0) return TrueCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = And(result, terms[i]);
    }
    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': {
```

```
3724              const x = simplifyCondCore(c.x);
3725              if (!x) return TrueCond;
3726              if (x.type === 'true') return FalseCond;
3727              if (x.type === 'false') return TrueCond;
3728              if (x.type === 'not') return simplifyCondCore(x.x);
3729              if (x.type === 'eq0') return Ne0(x.v);
3730              if (x.type === 'ne0') return Eq0(x.v);
3731              if (x.type === 'cmp') {
3732                  const negOp = negateOp(x.op);
3733                  if (negOp) return Cmp(x.l, negOp, x.r);
3734              }
3735              if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
3736              if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
3737              return Not(x);
3738          }
3739
3740  case 'and': {
3741      const a = simplifyCondCore(c.a);
3742      const b = simplifyCondCore(c.b);
3743
3744      if (!a) return b;
3745      if (!b) return a;
3746      if (a.type === 'false' || b.type === 'false') return FalseCond;
3747      if (a.type === 'true') return b;
3748      if (b.type === 'true') return a;
3749
3750      const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
3751
3752      // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
3753      const eq0Vars = new Map(); // var -> term
3754      const ne0Vars = new Map(); // var -> term
3755      const cmpTerms = [];
3756      const otherTerms = [];
3757
3758      for (const t of allTerms) {
3759          if (t.type === 'true') continue;
3760          if (t.type === 'false') return FalseCond;
3761
3762          if (t.type === 'eq0') {
3763              // Проверка на противоречие сразу
3764              if (ne0Vars.has(t.v)) {
3765                  console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
3766                  return FalseCond;
3767              }
3768              eq0Vars.set(t.v, t);
3769          } else if (t.type === 'ne0') {
3770              // Проверка на противоречие сразу
3771              if (eq0Vars.has(t.v)) {
3772                  console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
3773                  return FalseCond;
3774              }
3775              ne0Vars.set(t.v, t);
3776          } else if (t.type === 'cmp') {
3777              cmpTerms.push(t);
3778          } else if (t.type === 'or') {
3779              // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
3780              const orTerms = flattenOr(t);
3781              const validBranches = [];
3782
3783              for (const branch of orTerms) {
3784                  let branchValid = true;
3785
3786                  if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
3787                      console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
      {branch.v} = 0`);
```

```
3788                     branchValid = false;
3789                 } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
3790                     console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
     {branch.v} != 0`);
3791                     branchValid = false;
3792                 }
3793 
3794                 if (branchValid) {
3795                     validBranches.push(branch);
3796                 }
3797             }
3798 
3799             if (validBranches.length === 0) {
3800                 console.log(`Все ветки OR противоречат контексту → FALSE`);
3801                 return FalseCond;
3802             } else if (validBranches.length === 1) {
3803                 // Если осталась только одна ветка OR, добавляем её напрямую
3804                 const singleBranch = validBranches[0];
3805                 if (singleBranch.type === 'eq0') {
3806                     if (ne0Vars.has(singleBranch.v)) return FalseCond;
3807                     eq0Vars.set(singleBranch.v, singleBranch);
3808                 } else if (singleBranch.type === 'ne0') {
3809                     if (eq0Vars.has(singleBranch.v)) return FalseCond;
3810                     ne0Vars.set(singleBranch.v, singleBranch);
3811                 } else {
3812                     otherTerms.push(singleBranch);
3813                 }
3814             } else {
3815                 // Перестраиваем OR только с валидными ветками
3816                 otherTerms.push(buildOr(validBranches));
3817             }
3818         } else {
3819             otherTerms.push(t);
3820         }
3821     }
3822 
3823     // Собираем уникальные атомы
3824     const atomMap = new Map();
3825 
3826     for (const [v, term] of eq0Vars) {
3827         const key = atomKey(term);
3828         if (key) atomMap.set(key, term);
3829     }
3830 
3831     for (const [v, term] of ne0Vars) {
3832         const key = atomKey(term);
3833         if (key) atomMap.set(key, term);
3834     }
3835 
3836     for (const term of cmpTerms) {
3837         const key = atomKey(term);
3838         if (key) {
3839             const negKey = negateAtomKey(key);
3840             if (negKey && atomMap.has(negKey)) {
3841                 return FalseCond;
3842             }
3843             if (!atomMap.has(key)) {
3844                 atomMap.set(key, term);
3845             }
3846         }
3847     }
3848 
3849     let uniqueAtoms = Array.from(atomMap.values());
3850     uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
3851 
```

```
3852      let result = [...uniqueAtoms, ...otherTerms];
3853
3854      // Поглощение: X AND (X OR Y) = X
3855      // === НОВОЕ: выбрасываем из OR ветки, противоречащие контексту AND ===
3856      const contextAtoms = result.filter(t => isAtomCond(t));
3857      result = result.map(t => {
3858          if (t.type !== 'or') return t;
3859          return pruneOrByContext(t, contextAtoms);
3860      }).filter(t => t.type !== 'true'); // на всякий случай
3861
3862      result = applyAndAbsorption(result);
3863
3864      if (result.length === 0) return TrueCond;
3865      if (result.length === 1) return result[0];
3866
3867      return buildAnd(result);
3868  }
3869
3870          case 'or': {
3871              const a = simplifyCondCore(c.a);
3872              const b = simplifyCondCore(c.b);
3873
3874              if (!a) return b;
3875              if (!b) return a;
3876              if (a.type === 'true' || b.type === 'true') return TrueCond;
3877              if (a.type === 'false') return b;
3878              if (b.type === 'false') return a;
3879
3880              const allTerms = [...flattenOr(a), ...flattenOr(b)];
3881              const atomMap = new Map();
3882              const otherTerms = [];
3883
3884              for (const t of allTerms) {
3885                  if (t.type === 'true') return TrueCond;
3886                  if (t.type === 'false') continue;
3887
3888                  const key = atomKey(t);
3889                  if (key) {
3890                      const negKey = negateAtomKey(key);
3891                      if (negKey && atomMap.has(negKey)) {
3892                          return TrueCond;
3893                      }
3894                      if (!atomMap.has(key)) {
3895                          atomMap.set(key, t);
3896                      }
3897                  } else {
3898                      otherTerms.push(t);
3899                  }
3900              }
3901
3902              let uniqueAtoms = Array.from(atomMap.values());
3903              uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
3904
3905              let result = [...uniqueAtoms, ...otherTerms];
3906
3907              // Поглощение: X OR (X AND Y) = X
3908              result = applyOrAbsorption(result);
3909
3910              if (result.length === 0) return FalseCond;
3911              if (result.length === 1) return result[0];
3912
3913              return buildOr(result);
3914          }
3915
3916          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--;
            return expr;
        }

        try {
            return simplifyExprCore(expr);
        } finally {
            _depth--;
        }
    }

    function simplifyExprCore(expr) {
        if (!expr || expr.kind !== 'expr') return expr;

        switch (expr.type) {
            case 'const':
            case 'var':
                return expr;

            case 'op': {
                const l = simplifyExprCore(expr.l);
                const r = simplifyExprCore(expr.r);

                if (expr.op === '+') {
                    if (r?.type === 'const' && r.n === 0) return l;
                    if (l?.type === 'const' && l.n === 0) return r;
                }
                if (expr.op === '*') {
                    if (l?.type === 'const' && l.n === 0) return Const(0);
                    if (r?.type === 'const' && r.n === 0) return Const(0);
                    if (l?.type === 'const' && l.n === 1) return r;
                    if (r?.type === 'const' && r.n === 1) return l;
                }
                return Op(expr.op, l, r);
            }

            case 'when': {
                const c = simplifyCond(expr.c);
                const t = simplifyExprCore(expr.t);
                const e = simplifyExprCore(expr.e);

```

```
            if (c?.type === 'true') return t;
            if (c?.type === 'false') return e;
            if (exprEq(t, e)) return t;
            // ✅ НОВОЕ: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
            if (e && e.type === 'when') {
                const c2 = simplifyCond(e.c);
                const t2 = simplifyExprCore(e.t);
                const e2 = simplifyExprCore(e.e);

                if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
                    return When(c, t, t2);
                }
            }
            // Узкое правило: WHEN(A∧B, t1, WHEN(A∧¬B, t2, WHEN(¬A, t3, e3))) -> ... t3
            if (e && e.type === 'when') {
                const c2 = e.c, t2 = e.t, e2 = e.e;

                if (e2 && e2.type === 'when') {
                    const c3 = e2.c, t3 = e2.t;

                    const shared = findSharedAndComplement(c, c2);
                    if (shared && condNegationEq(c3, shared.shared)) {
                        return When(c, t, When(c2, t2, t3));
                    }
                }
            }

            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 '?';
    }
}
```

```
4045
4046  window.Optimizer = {
4047      Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
4048      Const, Var, Op, When,
4049      simplifyCond, simplifyExpr,
4050      printCond, printExpr,
4051      condEq, exprEq
4052  };
4053
4054  // js/codegen.js
4055
4056  const CodeGen = {
4057      _cache: {},
4058      _branchCache: {},
4059      _resolveCache: {},
4060      _visiting: new Set(),
4061
4062      reset() {
4063          this._cache = {};
4064          this._branchCache = {};
4065          this._resolveCache = {};
4066          this._visiting = new Set();
4067      },
4068
4069      toExpr(valueStr) {
4070          const s = String(valueStr).trim();
4071          if (s === '0') return Optimizer.Const(0);
4072          const num = parseFloat(s);
4073          if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
4074          return Optimizer.Var(s);
4075      },
4076
4077      exprToName(exprAst) {
4078          if (!exprAst) return '0';
4079          if (exprAst.type === 'var') return exprAst.name;
4080          if (exprAst.type === 'const') return String(exprAst.n);
4081          return Optimizer.printExpr(exprAst);
4082      },
4083
4084      mergeCond(a, b) {
4085          if (!a && !b) return null;
4086          if (!a) return b;
4087          if (!b) return a;
4088          if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
4089          return Optimizer.And(a, b);
4090      },
4091
4092      getConn(toId, toPort) {
4093          return AppState.connections.find(c => c.toElement === toId && c.toPort ===
      toPort);
4094      },
4095
4096      getConns(toId, prefix) {
4097          return AppState.connections.filter(c => c.toElement === toId &&
      c.toPort.startsWith(prefix));
4098      },
4099
4100      buildFormulaExpr(elem) {
4101          let result = elem.props.expression || '0';
4102
4103          // 1) Сначала раскрываем шаблоны (h и др.)
4104          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
4105              ? Settings.getTemplatesMap()
4106              : null;
4107          result = expandFormulaTemplates(result, map);
```

```
4108
4109          // 2) Потом раскрываем ссылки на формулы
4110          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
4111          for (const ref of formulaRefs) {
4112              const refElem = AppState.elements[ref];
4113              if (refElem && refElem.type === 'formula') {
4114                  const refExpr = this.buildFormulaExpr(refElem);
4115                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
4116              }
4117          }
4118
4119          return result;
4120      },
4121
4122      // === Получить ЧИСТУЮ логику элемента ===
4123      getPureLogic(id) {
4124          const cacheKey = `logic:${id}`;
4125          if (cacheKey in this._cache) {
4126              return this._cache[cacheKey];
4127          }
4128
4129          const elem = AppState.elements[id];
4130          if (!elem) return null;
4131
4132          let logic = null;
4133
4134          switch (elem.type) {
4135              case 'if': {
4136                  const leftConn = this.getConn(id, 'in-0');
4137                  const rightConn = this.getConn(id, 'in-1');
4138
4139                  const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
      Optimizer.Const(0);
4140                  const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
      Optimizer.Const(0);
4141
4142                  const op = (elem.props.operator || '=').trim();
4143                  const leftName = this.exprToName(leftVal);
4144                  const rightName = this.exprToName(rightVal);
4145
4146                  const leftZero = leftVal.type === 'const' && leftVal.n === 0;
4147                  const rightZero = rightVal.type === 'const' && rightVal.n === 0;
4148
4149                  switch (op) {
4150                      case '=':
4151                          if (rightZero) {
4152                              logic = Optimizer.Eq0(leftName);
4153                          } else if (leftZero) {
4154                              logic = Optimizer.Eq0(rightName);
4155                          } else {
4156                              logic = Optimizer.Cmp(leftName, '=', rightName);
4157                          }
4158                          break;
4159                      case '!=':
4160                          if (rightZero) {
4161                              logic = Optimizer.Ne0(leftName);
4162                          } else if (leftZero) {
4163                              logic = Optimizer.Ne0(rightName);
4164                          } else {
4165                              logic = Optimizer.Cmp(leftName, '!=', rightName);
4166                          }
4167                          break;
4168                      case '>':
4169                      case '<':
4170                      case '>=':
```

```
                    case '<=':
                        logic = Optimizer.Cmp(leftName, op, rightName);
                        break;
                    default:
                        logic = Optimizer.TrueCond;
                }
                break;
            }

            case 'and':
            case 'or': {
                const isAnd = elem.type === 'and';
                const count = elem.props.inputCount || 2;
                let result = null;

                for (let i = 0; i < count; i++) {
                    const conn = this.getConn(id, `in-${i}`);
                    if (!conn) continue;

                    const val = this.getPureLogic(conn.fromElement);
                    if (!val) continue;

                    if (result === null) {
                        result = val;
                    } else {
                        result = isAnd ? Optimizer.And(result, val) :
Optimizer.Or(result, val);
                    }
                }
                logic = result || Optimizer.FalseCond;
                break;
            }

            case 'not': {
                const conn = this.getConn(id, 'in-0');
                const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
                logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
                break;
            }

            case 'separator': {
                const conn = this.getConn(id, 'in-0');
                logic = conn ? this.getPureLogic(conn.fromElement) :
Optimizer.FalseCond;
                break;
            }

            default:
                logic = null;
        }

        // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
        if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
elem.type === 'not') {
            const ctx = this.getConditionFromPort(id);
            if (ctx) {
                if (logic) {
                    logic = Optimizer.And(ctx, logic);
                } else {
                    logic = ctx;
                }
            }
        }

        this._cache[cacheKey] = logic;
```

```
4233          return logic;
4234      },
4235
4236      // === Получить значение ===
4237      getValue(id) {
4238          const elem = AppState.elements[id];
4239          if (!elem) return Optimizer.Const(0);
4240
4241          switch (elem.type) {
4242              case 'input-signal':
4243                  // Имя сигнала или id как Var(...)
4244                  return this.toExpr(elem.props.name || id);
4245
4246              case 'const':
4247                  return Optimizer.Const(Number(elem.props.value) || 0);
4248
4249              case 'formula': {
4250                  // Используем текст формулы как выражение
4251                  const exprStr = this.buildFormulaExpr(elem) || '0';
4252                  return this.toExpr(exprStr);
4253              }
4254
4255              default:
4256                  // На всякий случай — даём символическое имя, а не 0
4257                  if (elem.props && typeof elem.props.name === 'string') {
4258                      return this.toExpr(elem.props.name);
4259                  }
4260                  return this.toExpr(id);
4261          }
4262      },
4263
4264      // === Получить ПОЛНОЕ условие для ветки сепаратора ===
4265      getBranchCondition(sepId, fromPort) {
4266          const cacheKey = `${sepId}:${fromPort}`;
4267          if (cacheKey in this._branchCache) {
4268              return this._branchCache[cacheKey];
4269          }
4270
4271          const sep = AppState.elements[sepId];
4272          if (!sep || sep.type !== 'separator') return null;
4273
4274          const inputLogic = this.getPureLogic(sepId);
4275          const sepContext = this.getConditionFromPort(sepId);
4276
4277          let branchLogic;
4278          if (fromPort === 'out-1') {
4279              branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
4280          } else {
4281              branchLogic = inputLogic || Optimizer.TrueCond;
4282          }
4283
4284          let result;
4285          if (sepContext) {
4286              result = Optimizer.And(sepContext, branchLogic);
4287          } else {
4288              result = branchLogic;
4289          }
4290
4291          this._branchCache[cacheKey] = result;
4292          return result;
4293      },
4294
4295      // === Получить условие от cond-порта ===
4296      getConditionFromPort(id) {
4297          const conn = this.getConn(id, 'cond-0');
```

```
4298          if (!conn) return null;
4299
4300          const sourceElem = AppState.elements[conn.fromElement];
4301          if (!sourceElem) return null;
4302
4303          if (sourceElem.type === 'separator') {
4304              return this.getBranchCondition(conn.fromElement, conn.fromPort);
4305          }
4306
4307          return this.getPureLogic(conn.fromElement);
4308      },
4309
4310      // === Основная функция разрешения ===
4311      resolve(id) {
4312          if (id in this._resolveCache) {
4313              return this._resolveCache[id];
4314          }
4315
4316          if (this._visiting.has(id)) {
4317              return null;
4318          }
4319          this._visiting.add(id);
4320
4321          const elem = AppState.elements[id];
4322          if (!elem) {
4323              this._visiting.delete(id);
4324              return null;
4325          }
4326
4327          let result = null;
4328
4329          try {
4330              switch (elem.type) {
4331                  case 'input-signal':
4332                      result = {
4333                          isValue: true,
4334                          cond: null,
4335                          expr: this.toExpr(elem.props.name || id)
4336                      };
4337                      break;
4338
4339                  case 'const': {
4340                      const cond = this.getConditionFromPort(id);
4341                      result = {
4342                          isValue: true,
4343                          cond: cond,
4344                          expr: Optimizer.Const(Number(elem.props.value) || 0)
4345                      };
4346                      break;
4347                  }
4348
4349                  case 'formula': {
4350                      let cond = this.getConditionFromPort(id);
4351
4352                      const inConns = this.getConns(id, 'in-');
4353                      for (const conn of inConns) {
4354                          const inputNode = this.resolve(conn.fromElement);
4355                          if (inputNode && inputNode.cond) {
4356                              cond = this.mergeCond(cond, inputNode.cond);
4357                          }
4358                      }
4359
4360                      const fullExpr = this.buildFormulaExpr(elem);
4361                      result = {
4362                          isValue: true,
```

```
                        cond: cond,
                        expr: Optimizer.Var(fullExpr)
                    };
                    break;
                }

                default:
                    result = null;
            }
        } finally {
            this._visiting.delete(id);
        }

        this._resolveCache[id] = result;
        return result;
    },

    generate() {
        console.log('=== Генерация кода (граф) ===');
        this.reset();

        try {
            const outputs = Object.values(AppState.elements).filter(e => e.type ===
'output');

            if (outputs.length === 0) {
                return '/* Нет выходов */';
            }

            const allVariants = [];

            for (const out of outputs) {
                const conns = this.getConns(out.id, 'in-');

                for (const conn of conns) {
                    console.log(`\n=== Обработка выхода ${out.id}, вход от $
{conn.fromElement} ===`);
                    const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
                    const result = CodeGenGraph.evalGraphValue(graph);
                    console.log(`Результат: cond=${Optimizer.printCond(result.cond)},
expr=${Optimizer.printExpr(result.expr)}`);

                    if (!result || !result.expr) continue;

                    const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
null;
                    const isZero = result.expr.type === 'const' && result.expr.n ===
0;

                    if (isZero && !cond) continue;

                    allVariants.push({
                        cond,
                        expr: result.expr,
                        isZero
                    });
                }
            }

            console.log('Варианты:', allVariants.map(v => ({
                cond: Optimizer.printCond(v.cond),
                expr: Optimizer.printExpr(v.expr)
            })));

            if (allVariants.length === 0) return '0';
```

```

            const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
            if (valueVariants.length === 0) return '0';

            let result = Optimizer.Const(0);

            for (let i = valueVariants.length - 1; i >= 0; i--) {
                const v = valueVariants[i];
                if (v.cond) {
                    result = Optimizer.When(v.cond, v.expr, result);
                } else {
                    result = v.expr;
                }
            }

            const simplified = Optimizer.simplifyExpr(result);
            return Optimizer.printExpr(simplified);

        } catch (err) {
            console.error('Ошибка:', err);
            return `/* Ошибка: ${err.message} */`;
        }
    }
};

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
```

```
4488          outputs: 1,
4489          inputLabels: ['A', 'B'],
4490          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
4491          outputLabels: ['результат'],
4492          outputTypes: [SIGNAL_TYPE.LOGIC],
4493          color: '#a855f7',
4494          hasProperties: true,  // ← Теперь есть свойства (для изменения количества
      входов)
4495          resizable: true,
4496          minWidth: 120,
4497          minHeight: 80,
4498          hasConditionPort: true,
4499          conditionPortType: SIGNAL_TYPE.LOGIC,
4500          defaultProps: {
4501              inputCount: 2  // ← Новое свойство
4502          }
4503      },
4504      'or': {
4505          name: 'ИЛИ',
4506          inputs: 2,  // По умолчанию 2
4507          outputs: 1,
4508          inputLabels: ['A', 'B'],
4509          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
4510          outputLabels: ['результат'],
4511          outputTypes: [SIGNAL_TYPE.LOGIC],
4512          color: '#a855f7',
4513          hasProperties: true,  // ← Теперь есть свойства
4514          resizable: true,
4515          minWidth: 120,
4516          minHeight: 80,
4517          hasConditionPort: true,
4518          conditionPortType: SIGNAL_TYPE.LOGIC,
4519          defaultProps: {
4520              inputCount: 2  // ← Новое свойство
4521          }
4522      },
4523      'not': {
4524          name: 'НЕ',
4525          inputs: 1,
4526          outputs: 1,
4527          inputLabels: ['A'],
4528          inputTypes: [SIGNAL_TYPE.LOGIC],
4529          outputLabels: ['¬A'],
4530          outputTypes: [SIGNAL_TYPE.LOGIC],
4531          color: '#a855f7',
4532          hasProperties: true,
4533          resizable: true,
4534          minWidth: 100,
4535          minHeight: 60,
4536          hasConditionPort: true,
4537          conditionPortType: SIGNAL_TYPE.LOGIC
4538      },
4539      'if': {
4540          name: 'ЕСЛИ',
4541          inputs: 2,
4542          outputs: 1,  // ← Только один выход!
4543          inputLabels: ['A', 'B'],
4544          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
4545          outputLabels: ['результат'],  // ← Просто результат
4546          outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
4547          color: '#e94560',
4548          hasProperties: true,
4549          defaultProps: { operator: '=' },
4550          resizable: true,
4551          minWidth: 120,
```

```
4552          minHeight: 80,
4553          hasConditionPort: true,
4554          conditionPortType: SIGNAL_TYPE.LOGIC
4555      },
4556      'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
4557          name: 'Сепаратор',
4558          inputs: 1,
4559          outputs: 2,
4560          inputLabels: ['сигнал'],
4561          inputTypes: [SIGNAL_TYPE.LOGIC],
4562          outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
4563          outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
4564          color: '#f59e0b',
4565          hasProperties: true,
4566          resizable: true,
4567          minWidth: 120,
4568          minHeight: 80,
4569          hasConditionPort: true,
4570          conditionPortType: SIGNAL_TYPE.LOGIC
4571      },
4572      'const': {
4573          name: 'Константа',
4574          inputs: 0,
4575          outputs: 1,
4576          outputLabels: ['out'],
4577          outputTypes: [SIGNAL_TYPE.NUMERIC],
4578          color: '#3b82f6',
4579          hasProperties: true,
4580          defaultProps: { value: 0 },
4581          resizable: true,
4582          minWidth: 120,
4583          minHeight: 60,
4584          hasConditionPort: true,
4585          conditionPortType: SIGNAL_TYPE.LOGIC
4586      },
4587      'formula': {
4588          name: 'Формула',
4589          inputs: 2,
4590          outputs: 1,
4591          inputLabels: ['in₁', 'in₂'],
4592          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
4593          outputLabels: ['результат'],
4594          outputTypes: [SIGNAL_TYPE.NUMERIC],
4595          color: '#f59e0b',
4596          hasProperties: true,
4597          resizable: true,
4598          minWidth: 140,
4599          minHeight: 80,
4600          defaultProps: {
4601              expression: '',
4602              inputCount: 2
4603          },
4604          hasConditionPort: true,
4605          conditionPortType: SIGNAL_TYPE.LOGIC
4606      },
4607      'output': {
4608          name: 'Выход',
4609          inputs: 1,
4610          outputs: 0,
4611          inputLabels: ['сигнал'],
4612          inputTypes: [SIGNAL_TYPE.ANY],
4613          color: '#10b981',
4614          hasProperties: true,
4615          defaultProps: { label: 'Выход', outputGroup: '' },
4616          resizable: true,
```

```
4617          minWidth: 150,
4618          minHeight: 60,
4619      },  // ← важно, если предыдущий элемент не заканчивается запятой
4620      'group': {
4621          name: 'Группа',
4622          inputs: 0,
4623          outputs: 0,
4624          color: '#6b7280',
4625          resizable: true,
4626          minWidth: 200,
4627          minHeight: 120,
4628          hasProperties: true,
4629          defaultProps: { title: 'Группа' }
4630      }
4631  };
4632
4633  const VIEWPORT_CONFIG = {
4634      minZoom: 0.1,
4635      maxZoom: 3,
4636      zoomStep: 0.1,
4637      panSpeed: 1,
4638      canvasWidth: 5000,
4639      canvasHeight: 5000
4640  };
4641
4642  const MINIMAP_CONFIG = {
4643      width: 200,
4644      height: 150,
4645      padding: 10
4646  };
4647
4648  /**
4649   * Модуль работы с соединениями
4650   * connections.js
4651   */
4652
4653  const Connections = {
4654      /**
4655       * Настройка обработчиков порта
4656       */
4657      setupPortHandlers(port) {
4658          port.addEventListener('mousedown', (e) => {
4659              e.stopPropagation();
4660
4661              if (port.classList.contains('output')) {
4662                  const elemId = port.dataset.element;
4663                  const portName = port.dataset.port;
4664                  const signalType = getOutputPortType(elemId, portName);
4665
4666                  AppState.connectingFrom = {
4667                      element: elemId,
4668                      port: portName
4669                  };
4670                  AppState.connectingFromType = signalType;
4671
4672                  this.highlightCompatiblePorts(signalType);
4673
4674                  const svg = document.getElementById('connections-svg');
4675                  const startPos = this._getPortCanvasCenter(port);
4676
4677                  AppState.tempLine = document.createElementNS('http://www.w3.org/2000/
      svg', 'path');
4678                  AppState.tempLine.setAttribute('class', 'temp-connection');
4679                  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
                    );

                    if (!exists || allowMultipleInputs) {
                        AppState.connections.push({
                            fromElement: AppState.connectingFrom.element,
                            fromPort: AppState.connectingFrom.port,
                            toElement,
                            toPort: toPortName,
                            signalType: AppState.connectingFromType
                        });

                        port.classList.add('connected');
                        this.drawConnections();
                        this.clearConnectionState();
                        return;
                    }
                }
            }

            this.clearConnectionState();
        });

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

                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
                    if (AppState.tempLine) {
                        AppState.tempLine.classList.add('invalid');
                    }
                }
            }
        });

        port.addEventListener('mouseleave', () => {
            if (AppState.tempLine) {
                AppState.tempLine.classList.remove('invalid');
            }
        });
    },
```

```
4745
4746      /**
4747       * Подсветка совместимых портов
4748       */
4749      highlightCompatiblePorts(signalType) {
4750          document.querySelectorAll('.port.input').forEach(port => {
4751              const inputType = getInputPortType(port.dataset.element,
      port.dataset.port);
4752
4753              if (areTypesCompatible(signalType, inputType)) {
4754                  port.classList.add('compatible-highlight');
4755              } else {
4756                  port.classList.add('incompatible');
4757              }
4758          });
4759      },
4760
4761      /**
4762       * Очистка состояния соединения
4763       */
4764      clearConnectionState() {
4765          if (AppState.tempLine) {
4766              AppState.tempLine.remove();
4767              AppState.tempLine = null;
4768          }
4769          AppState.connectingFrom = null;
4770          AppState.connectingFromType = null;
4771
4772          document.querySelectorAll('.port').forEach(port => {
4773              port.classList.remove('compatible-highlight', 'incompatible');
4774          });
4775      },
4776
4777      /**
4778       * Отрисовка временной линии соединения
4779       */
4780      drawTempConnection(e) {
4781      if (!AppState.tempLine || !AppState.connectingFrom) return;
4782
4783      const fromElem = document.getElementById(AppState.connectingFrom.element);
4784      if (!fromElem) return;
4785
4786      const fromPort = fromElem.querySelector(`[data-port="$
      {AppState.connectingFrom.port}"]`);
4787      if (!fromPort) return;
4788
4789      const startPos = this._getPortCanvasCenter(fromPort);
4790      const endPos = screenToCanvas(e.clientX, e.clientY);
4791
4792      const horizontalDist = Math.abs(endPos.x - startPos.x);
4793      const controlDist = Math.max(horizontalDist * 0.4, 50);
4794
4795      // Тянем всегда от выхода (вектор 1, 0)
4796      const cx1 = startPos.x + controlDist;
4797      const cy1 = startPos.y;
4798
4799      // Вторая точка контроля для плавности за курсором
4800      const cx2 = endPos.x - controlDist;
4801      const cy2 = endPos.y;
4802
4803      AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} $
      {cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
4804      AppState.tempLine.setAttribute('fill', 'none');
4805  },
4806
```

```
4807      /**
4808       * Отрисовка всех соединений
4809       */
4810      drawConnections() {
4811      const svg = document.getElementById('connections-svg');
4812
4813      // 1. Очистка старых линий
4814      svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
4815
4816      // 2. Сброс визуального состояния портов
4817      document.querySelectorAll('.port.connected').forEach(port => {
4818          port.classList.remove('connected');
4819      });
4820
4821      // 3. Перебор всех соединений из AppState
4822      AppState.connections.forEach(conn => {
4823          const fromElem = document.getElementById(conn.fromElement);
4824          const toElem = document.getElementById(conn.toElement);
4825
4826          if (!fromElem || !toElem) return;
4827
4828          const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
4829          const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
4830
4831          if (!fromPort || !toPort) return;
4832
4833          fromPort.classList.add('connected');
4834          toPort.classList.add('connected');
4835
4836          const startPos = this._getPortCanvasCenter(fromPort);
4837          const endPos = this._getPortCanvasCenter(toPort);
4838
4839          if (!startPos || !endPos) return;
4840
4841          // Расстояние для изгиба кривой
4842          const horizontalDist = Math.abs(endPos.x - startPos.x);
4843          const verticalDist = Math.abs(endPos.y - startPos.y);
4844          const controlDist = Math.max(horizontalDist * 0.4, 50);
4845
4846          // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
4847          let d;
4848          let cx1 = startPos.x;
4849          let cy1 = startPos.y;
4850          let cx2 = endPos.x;
4851          let cy2 = endPos.y;
4852
4853          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
4854          cx1 = startPos.x + controlDist;
4855          cy1 = startPos.y;
4856
4857          // ВХОД (Target):
4858          if (conn.toPort === 'cond-0') {
4859              // Технический порт: Касательная (0, 1) в декартовой (вверх)
4860              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
4861              cx2 = endPos.x;
4862              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
4863          } else {
4864              // Обычный вход: Касательная (-1, 0) -> Слева направо
4865              cx2 = endPos.x - controlDist;
4866              cy2 = endPos.y;
4867          }
4868
4869          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
      ${endPos.y}`;
4870
```

```
4871          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
4872          path.setAttribute('d', d);
4873          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
4874
4875          // --- ЛОГИКА ЦВЕТА (Классы) ---
4876          let cssClass = 'connection';
4877          const type = conn.signalType;
4878
4879          // Приоритет новым типам сигналов
4880          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
4881          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
4882          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
4883          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
4884          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
4885
4886          path.setAttribute('class', cssClass);
4887
4888          // Обработчики событий
4889          path.style.pointerEvents = 'stroke';
4890          path.style.cursor = 'pointer';
4891          path.addEventListener('click', () => this.handleConnectionClick(conn));
4892
4893          svg.appendChild(path);
4894      });
4895
4896      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
4897          Outputs.updateOutputStatus();
4898      }
4899      Viewport.updateMinimap();
4900  },
4901      /**
4902       * Обработка клика по соединению (удаление)
4903       */
4904      handleConnectionClick(conn) {
4905          if (confirm('Удалить соединение?')) {
4906              AppState.connections = AppState.connections.filter(c =>
4907                  !(c.fromElement === conn.fromElement &&
4908                    c.fromPort === conn.fromPort &&
4909                    c.toElement === conn.toElement &&
4910                    c.toPort === conn.toPort)
4911              );
4912
4913              this.drawConnections();
4914          }
4915      },
4916
4917      /**
4918       * Получение центра порта в координатах Canvas
4919       */
4920      _getPortCanvasCenter(portEl) {
4921          if (!portEl) return null;
4922
4923          const rect = portEl.getBoundingClientRect();
4924          return screenToCanvas(
4925              rect.left + rect.width / 2,
4926              rect.top + rect.height / 2
4927          );
4928      }
4929  };
4930
4931  /**
4932   * Модуль работы с элементами схемы
4933   * elements.js
4934   */
4935
```

```
4936  const Elements = {
4937      /**
4938       * Генерация HTML для элемента
4939       */
4940          createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
4941              const config = ELEMENT_TYPES[elemType];
4942              if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
4943
4944              const safe = (value, fallback = '') => (value === null || value ===
      undefined) ? fallback : String(value);
4945              const w = width ?? config.minWidth ?? 120;
4946              const h = height ?? config.minHeight ?? 60;
4947
4948              const getPortClass = (signalType, direction) => {
4949                  const base = direction === 'output' ? 'port output' : 'port input';
4950                  if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
4951                  if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
4952                  return `${base} any-port`;
4953              };
4954
4955              // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
      innerHTML
4956              // Она тут остается, но ее результат не встраивается в HTML-строку
      напрямую, кроме формулы
4957              const buildConditionPortHTML = () => {
4958                  return `
4959                      <div class="condition-port-wrapper">
4960                          <div class="condition-port-label">условие</div>
4961                          <div class="port input condition-port"
4962                              data-port="cond-0"
4963                              data-element="${elemId}"
4964                              data-signal-type="${SIGNAL_TYPE.LOGIC}"
4965                              title="Техническое условие">
4966                          </div>
4967                      </div>`;
4968              };
4969
4970
4971              const buildInputPorts = (count, types = [], labels = []) => {
4972                  let html = '';
4973                  for (let i = 0; i < count; i++) {
4974                      const type = types[i] ?? types[types.length - 1] ??
      SIGNAL_TYPE.ANY;
4975                      html += `<div class="${getPortClass(type, 'input')}" data-
      port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Вход ${i+1}`}"></div>`;
4976                  }
4977                  return html;
4978              };
4979
4980              const buildOutputPorts = (count, types = [], labels = []) => {
4981                  let html = '';
4982                  for (let i = 0; i < count; i++) {
4983                      const type = types[i] ?? types[types.length - 1] ??
      SIGNAL_TYPE.ANY;
4984                      html += `<div class="${getPortClass(type, 'output')}" data-
      port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Выход ${i+1}`}"></div>`;
4985                  }
4986                  return html;
4987              };
4988
4989              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>` :
```

```
    '';
4990            // hasCondClass будет добавляться в addElement
4991            // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
    '';
4992
4993            let innerHTML = '';
4994
4995            if (elemType === 'input-signal') {
4996                const name = safe(props.name, 'Сигнал');
4997                const type = props.signalType || SIGNAL_TYPE.NUMBER;
4998                const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
4999                innerHTML = `
5000                    <div class="element-header" style="background:$
    {config.color};">Источник</div>
5001                    <div class="element-body">
5002                        <div class="element-symbol">
5003                            <span class="input-signal-icon">${symbol}</span>
5004                            <span class="input-signal-name">${name}</span>
5005                        </div>
5006                        <div class="ports-right">
5007                            ${buildOutputPorts(1, [type], ['Выход'])}
5008                        </div>
5009                    </div>`;
5010            }
5011            else if (elemType === 'const') {
5012                innerHTML = `
5013                    <div class="element-header" style="background:$
    {config.color};">Константа</div>
5014                    <div class="element-body">
5015                        <div class="element-symbol">${props.value ?? 0}</div>
5016                        <div class="ports-right">
5017                            ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
5018                        </div>
5019                    </div>`;
5020            }
5021            else if (elemType === 'separator') {
5022                innerHTML = `
5023                    <div class="element-header" style="background:$
    {config.color};">Сепаратор</div>
5024                    <div class="element-body">
5025                        <div class="ports-left">${buildInputPorts(1,
    config.inputTypes, config.inputLabels)}</div>
5026                        <div class="element-symbol">✓/✗</div>
5027                        <div class="ports-right">
5028                            <div class="port output logic-port true-port" data-
    port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
    title="ИСТИНА"></div>
5029                            <div class="port output logic-port false-port" data-
    port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
    title="ЛОЖЬ"></div>
5030                        </div>
5031                    </div>`;
5032            }
5033            else if (elemType === 'and' || elemType === 'or') {
5034                const gateSymbol = elemType === 'and' ? '∧' : '∨';
5035                const inputCount = props.inputCount || config.defaultProps?.inputCount
    || 2;
5036
5037                // Генерируем динамические входы
5038                let inputsHTML = '';
5039                for (let i = 0; i < inputCount; i++) {
5040                    inputsHTML += `<div class="port input logic-port" data-port="in-$
    {i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход $
    {i+1}"></div>`;
5041                }
```

```
5042
5043                 innerHTML = `
5044                     <div class="element-header" style="background:${config.color};">${config.name}</div>
5045                     <div class="element-body">
5046                         <div class="ports-left">
5047                             ${inputsHTML}
5048                         </div>
5049                         <div class="element-symbol">${gateSymbol}</div>
5050                         <div class="ports-right">
5051                             <div class="port output logic-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></div>
5052                         </div>
5053                     </div>`;
5054             }
5055             else if (elemType === 'if') {
5056                 const op = safe(props.operator, '=');
5057                 innerHTML = `
5058                     <div class="element-header" style="background:${config.color};">Условие</div>
5059                     <div class="element-body">
5060                         <div class="ports-left">${buildInputPorts(2, config.inputTypes, config.inputLabels)}</div>
5061                         <div class="element-symbol">${op}</div>
5062                         <div class="ports-right">
5063                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
5064                         </div>
5065                     </div>`;
5066             }
5067             else if (elemType === 'not') {
5068                 innerHTML = `
5069                     <div class="element-header" style="background:${config.color};">НЕ</div>
5070                     <div class="element-body">
5071                         <div class="ports-left">${buildInputPorts(1, [SIGNAL_TYPE.LOGIC], ['A'])}</div>
5072                         <div class="element-symbol">¬</div>
5073                         <div class="ports-right">
5074                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
5075                         </div>
5076                     </div>`;
5077             }
5078             else if (elemType === 'formula') {
5079                 const inputCount = props.inputCount || config.defaultProps?.inputCount || config.inputs || 2;
5080                 const expression = safe(props.expression);
5081                 const displayExpression = expression
5082                     ? (expression.length > 12 ? `${expression.slice(0, 12)}…` : expression)
5083                     : 'f(x)';
5084
5085                 innerHTML = `
5086                     ${buildConditionPortHTML()}
5087                     <div class="element-header" style="background:${config.color};">Формула</div>
5088                     <div class="element-body">
5089                         <div class="ports-left">${buildInputPorts(inputCount, config.inputTypes, config.inputLabels)}</div>
5090                         <div class="element-symbol">${displayExpression}</div>
5091                         <div class="ports-right">
5092                             ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Результат'])}
5093                         </div>
5094                     </div>`;
```

```
5095             }
5096             else if (elemType === 'output') {
5097                 innerHTML = `
5098                     <div class="element-header" style="background:$
     {config.color};">Выход</div>
5099                     <div class="element-body">
5100                         <div class="ports-left">
5101                             ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
5102                         </div>
5103                         <div class="element-symbol">${safe(props.label, 'Выход')}</
     div>
5104                         <div class="ports-right"></div>
5105                     </div>`;
5106 
5107             }
5108             else if (elemType === 'group') {
5109                 const title = props.title || 'Группа';
5110                 innerHTML = `
5111                     <div class="group-content">
5112                     <div class="group-title">${title}</div>
5113                     </div>`;
5114             }
5115 
5116             else { // Для любых других (fallback)
5117                 innerHTML = `
5118                     <div class="element-header" style="background:${config.color};">$
     {config.name}</div>
5119                     <div class="element-body">
5120                         <div class="ports-left">${buildInputPorts(config.inputs || 0,
     config.inputTypes, config.inputLabels)}</div>
5121                         <div class="element-symbol">${config.name}</div>
5122                         <div class="ports-right">
5123                             ${buildOutputPorts(config.outputs || 0,
     config.outputTypes, config.outputLabels)}
5124                         </div>
5125                     </div>`;
5126             }
5127             const commentHtml = `<div class="element-comment">${safe(props.comment,
     '')}</div>`;
5128 
5129 
5130             const html = `
5131                 <div class="element ${elemType}" id="${elemId}"
5132                     style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;"
     data-type="${elemType}">
5133                     ${innerHTML}
5134                     ${commentHtml}
5135                     ${resizeHandles}
5136                 </div>`;
5137 
5138             return { html, width: w, height: h };
5139         },
5140 
5141     /**
5142      * Добавление элемента
5143      */
5144         addElement(elemType, x, y, props = {}, elemId = null, customWidth = null,
     customHeight = null) {
5145             const config = ELEMENT_TYPES[elemType];
5146             if (!config) {
5147                 console.error(`Неизвестный тип элемента: ${elemType}`);
5148                 return null;
5149             }
5150 
5151             if (!elemId) {
```

```
5152                 elemId = `${elemType}_${++AppState.elementCounter}`;
5153             }
5154 
5155             let width = customWidth;
5156             let height = customHeight;
5157 
5158             if (width === null || width === undefined) {
5159                 width = config.minWidth || 140;
5160             }
5161             if (height === null || height === undefined) {
5162                 height = config.minHeight || 70;
5163             }
5164 
5165             try {
5166                 const result = this.createElementHTML(elemType, elemId, x, y, props,
      width, height);
5167                 if (!result || !result.html) {
5168                     console.error('createElementHTML вернул пустой результат');
5169                     return null;
5170                 }
5171 
5172                 const workspace = document.getElementById('workspace');
5173                 const wrapper = document.createElement('div');
5174                 wrapper.innerHTML = result.html.trim();
5175                 const element = wrapper.firstElementChild;
5176                 if (!element) {
5177                     console.error('Не удалось создать DOM элемент из HTML');
5178                     return null;
5179                 }
5180 
5181                 // Добавляем класс для отступа
5182                 if (config.hasConditionPort) {
5183                     element.classList.add('has-condition-port');
5184                 }
5185 
5186                 workspace.appendChild(element);
5187 
5188                 AppState.elements[elemId] = {
5189                     id: elemId,
5190                     type: elemType,
5191                     x,
5192                     y,
5193                     width: result.width || width,
5194                     height: result.height || height,
5195                     props: { ...(config.defaultProps || {}), ...(props || {}) }
5196                 };
5197 
5198                 // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ
      ЕГО В HTML)
5199                 if (config.hasConditionPort && elemType !== 'formula') {
5200                     const condPortWrapper = document.createElement('div');
5201                     condPortWrapper.innerHTML = `
5202                         <div class="condition-port-wrapper">
5203                             <div class="condition-port-label">условие</div>
5204                             <div class="port input condition-port"
5205                                 data-port="cond-0"
5206                                 data-element="${elemId}"
5207                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"
5208                                 title="Техническое условие">
5209                             </div>
5210                         </div>`;
5211                     element.prepend(condPortWrapper.firstElementChild); // Вставляем в
      самое начало элемента
5212                 }
5213 
```

```
5214
5215                this.setupElementHandlers(elemId); // Передаем ID элемента
5216
5217                // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
5218                // element.querySelectorAll('.port').forEach(port => {
5219                //     Connections.setupPortHandlers(port);
5220                // });
5221
5222                Connections.drawConnections(); // Перерисовываем соединения, чтобы
      учесть новые порты
5223                Viewport.updateMinimap();
5224                return elemId;
5225            } catch (err) {
5226                console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
5227                return null;
5228            }
5229        },
5230
5231    /**
5232     * Обновление входов логического элемента (AND, OR)
5233     */
5234    updateLogicGateInputs(elemId, inputCount) {
5235        const elem = document.getElementById(elemId);
5236        if (!elem) return;
5237
5238        const portsLeft = elem.querySelector('.ports-left');
5239        if (!portsLeft) return;
5240
5241        // Удаляем соединения к портам, которые больше не существуют
5242        AppState.connections = AppState.connections.filter(c => {
5243            if (c.toElement === elemId && c.toPort.startsWith('in-')) {
5244                const portNum = parseInt(c.toPort.split('-')[1], 10);
5245                return portNum < inputCount;
5246            }
5247            return true;
5248        });
5249
5250        // Генерируем новые входы
5251        let inputsHTML = '';
5252        for (let i = 0; i < inputCount; i++) {
5253            inputsHTML += `
5254                <div class="port input logic-port"
5255                    data-port="in-${i}"
5256                    data-element="${elemId}"
5257                    data-signal-type="${SIGNAL_TYPE.LOGIC}"
5258                    title="Вход ${i+1}">
5259                </div>
5260            `;
5261        }
5262        portsLeft.innerHTML = inputsHTML;
5263
5264        // Переподключаем обработчики
5265        portsLeft.querySelectorAll('.port').forEach(port =>
5266            Connections.setupPortHandlers(port)
5267        );
5268
5269        Connections.drawConnections();
5270    },
5271
5272    /**
5273     * Удаление элемента
5274     */
5275    deleteElement(elemId) {
5276        AppState.connections = AppState.connections.filter(c =>
5277            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];

        const elem = document.getElementById(elemId);
        if (elem) elem.classList.add('selected');

        const elemData = AppState.elements[elemId];
        if (elemData) {
            document.getElementById('selection-info').textContent =
                `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
        }
    },

    deselectAll() {
        // Снимаем класс со всех элементов на странице
        document.querySelectorAll('.element.selected').forEach(el =>
    el.classList.remove('selected'));

        AppState.selectedElement = null;
        AppState.selectedElements = [];
        if (document.getElementById('selection-info')) {
            document.getElementById('selection-info').textContent = '';
        }
    },
    /**
     * Настройка обработчиков элемента
     */
    setupElementHandlers(elemId) {
        try {
            const elem = document.getElementById(elemId);
            if (!elem) return;

            // elements.js -> setupElementHandlers
            elem.addEventListener('mousedown', (e) => {
                if (e.target.classList.contains('port')) return;
                if (e.target.classList.contains('resize-handle')) return;

                e.preventDefault();
                e.stopPropagation();

                // ПРАВКА ТУТ:
                // Если элемент НЕ в группе — выделяем только его.
```

```
                // Если элемент УЖЕ в группе — не трогаем группу, чтобы можно было тянуть всех.
                if (!AppState.selectedElements.includes(elemId)) {
                    this.selectElement(elemId);
                }

                AppState.draggingElement = elemId;
                const canvasPos = screenToCanvas(e.clientX, e.clientY);
                const elemData = AppState.elements[elemId];
                AppState.dragOffset.x = canvasPos.x - elemData.x;
                AppState.dragOffset.y = canvasPos.y - elemData.y;
            });

            elem.addEventListener('dblclick', (e) => {
                if (e.target.classList.contains('port')) return;
                const config = ELEMENT_TYPES[AppState.elements[elemId].type];
                if (config?.hasProperties) {
                    Modal.showPropertiesModal(elemId);
                }
            });

            elem.addEventListener('contextmenu', (e) => {
                e.preventDefault();
                this.showContextMenu(e.clientX, e.clientY, elemId);
            });

            const handles = elem.querySelectorAll('.resize-handle');
            handles.forEach(handle => this.setupResizeHandlers(handle, elemId));

            const ports = elem.querySelectorAll('.port');
            ports.forEach(port => Connections.setupPortHandlers(port));

        } catch (err) {
            console.error('setupElementHandlers error for', elemId, err);
        }
    },

    /**
     * Контекстное меню
     */
    showContextMenu(x, y, elemId) {
        const menu = document.getElementById('context-menu');
        menu.style.left = `${x}px`;
        menu.style.top = `${y}px`;
        menu.style.display = 'block';
        menu.dataset.elementId = elemId;
    },

    /**
     * Настройка resize
     */
    setupResizeHandlers(handle, elemId) {
        handle.addEventListener('mousedown', (e) => {
            e.stopPropagation();
            e.preventDefault();

            const elemData = AppState.elements[elemId];

            AppState.resizing = {
                elemId: elemId,
                handle: handle.dataset.direction,
                startX: e.clientX,
                startY: e.clientY,
                startWidth: elemData.width,
                startHeight: elemData.height,
```

```
                startLeft: elemData.x,
                startTop: elemData.y
            };
        });
    },
// elements.js — ЗАМЕНИ функцию copySelectedElements
    copySelectedElements() {
        const ids = (AppState.selectedElements && AppState.selectedElements.length > 0)
            ? [...AppState.selectedElements]
            : (AppState.selectedElement ? [AppState.selectedElement] : []);

        if (ids.length === 0) {
            console.log('Нечего копировать');
            return;
        }

        const originals = ids
            .map(id => AppState.elements[id])
            .filter(Boolean);

        if (originals.length === 0) return;

        const offsetX = 50;
        const offsetY = 50;

        const idMap = {};
        const newIds = [];

        originals.forEach(el => {
            // Копируем свойства элемента (глубокое копирование props)
            const newProps = JSON.parse(JSON.stringify(el.props || {}));

            // Используем существующую функцию addElement
            // Она сама сгенерирует ID и создаст DOM
            const createdId = this.addElement(
                el.type,                    // тип элемента
                el.x + offsetX,             // новая позиция X
                el.y + offsetY,             // новая позиция Y
                newProps,                   // скопированные свойства
                null,                       // ID = null, чтобы addElement сгенерировал сам
                el.width,                   // ширина
                el.height                   // высота
            );

            if (createdId) {
                idMap[el.id] = createdId;
                newIds.push(createdId);
            }
        });

        // Копируем связи ТОЛЬКО между скопированными элементами
        const newConnections = [];
        AppState.connections.forEach(conn => {
            if (idMap[conn.fromElement] && idMap[conn.toElement]) {
                newConnections.push({
                    fromElement: idMap[conn.fromElement],
                    fromPort: conn.fromPort,
                    toElement: idMap[conn.toElement],
                    toPort: conn.toPort,
                    signalType: conn.signalType || 'Boolean'
                });
            }
        });
```

```
5469
5470            AppState.connections.push(...newConnections);
5471            Connections.drawConnections();
5472
5473            // Выделяем новые элементы
5474            this.deselectAll();
5475            AppState.selectedElements = newIds;
5476            AppState.selectedElement = newIds[newIds.length - 1];
5477
5478            newIds.forEach(id => {
5479                const el = document.getElementById(id);
5480                if (el) el.classList.add('selected');
5481            });
5482
5483            document.getElementById('selection-info').textContent =
5484                `Скопировано: ${newIds.length} элемент(ов)`;
5485
5486            Viewport.updateMinimap();
5487            console.log(`Скопировано ${newIds.length} элементов`);
5488        },
5489
5490            // elements.js — добавь в объект Elements
5491        deleteSelectedElements() {
5492            const ids = (AppState.selectedElements && AppState.selectedElements.length >
      0)
5493                ? [...AppState.selectedElements]
5494                : (AppState.selectedElement ? [AppState.selectedElement] : []);
5495
5496            if (ids.length === 0) {
5497                console.log('Нечего удалять');
5498                return;
5499            }
5500
5501            // Удаляем каждый элемент
5502            ids.forEach(id => {
5503                this.deleteElement(id);
5504            });
5505
5506            // Сбрасываем выделение
5507            AppState.selectedElement = null;
5508            AppState.selectedElements = [];
5509            document.getElementById('selection-info').textContent = '';
5510
5511            console.log(`Удалено ${ids.length} элементов`);
5512        },
5513
5514        /**
5515         * Обработка resize
5516         */
5517        handleResize(e) {
5518            if (!AppState.resizing) return;
5519
5520            const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
      startTop } = AppState.resizing;
5521            const elem = document.getElementById(elemId);
5522            const elemData = AppState.elements[elemId];
5523            const config = ELEMENT_TYPES[elemData.type];
5524
5525            const dx = (e.clientX - startX) / AppState.viewport.zoom;
5526            const dy = (e.clientY - startY) / AppState.viewport.zoom;
5527
5528            let newWidth = startWidth;
5529            let newHeight = startHeight;
5530            let newLeft = startLeft;
5531            let newTop = startTop;
```

```
5532
5533          if (handle.includes('e')) {
5534              newWidth = Math.max(config.minWidth, startWidth + dx);
5535          }
5536          if (handle.includes('w')) {
5537              newWidth = Math.max(config.minWidth, startWidth - dx);
5538              newLeft = startLeft + (startWidth - newWidth);
5539          }
5540          if (handle.includes('s')) {
5541              newHeight = Math.max(config.minHeight, startHeight + dy);
5542          }
5543          if (handle.includes('n')) {
5544              newHeight = Math.max(config.minHeight, startHeight - dy);
5545              newTop = startTop + (startHeight - newHeight);
5546          }
5547
5548          elem.style.width = `${newWidth}px`;
5549          elem.style.height = `${newHeight}px`;
5550          elem.style.left = `${newLeft}px`;
5551          elem.style.top = `${newTop}px`;
5552
5553          elemData.width = newWidth;
5554          elemData.height = newHeight;
5555          elemData.x = newLeft;
5556          elemData.y = newTop;
5557
5558          Connections.drawConnections();
5559      },
5560
5561      /**
5562       * Обработка перетаскивания элемента
5563       */
5564      handleDrag(e) {
5565          if (!AppState.draggingElement) return;
5566
5567          const canvasPos = screenToCanvas(e.clientX, e.clientY);
5568          const elemId = AppState.draggingElement;
5569          const elemData = AppState.elements[elemId];
5570          if (!elemData) return;
5571
5572          const newX = canvasPos.x - AppState.dragOffset.x;
5573          const newY = canvasPos.y - AppState.dragOffset.y;
5574          const dx = newX - elemData.x;
5575          const dy = newY - elemData.y;
5576
5577          // если выделено несколько
5578          const group = AppState.selectedElements && AppState.selectedElements.length >
      1
5579              ? AppState.selectedElements
5580              : [elemId];
5581
5582          for (const id of group) {
5583              const elData = AppState.elements[id];
5584              if (!elData) continue;
5585              elData.x += dx;
5586              elData.y += dy;
5587              const el = document.getElementById(id);
5588              if (el) {
5589              el.style.left = elData.x + 'px';
5590              el.style.top = elData.y + 'px';
5591              }
5592          }
5593
5594          Connections.drawConnections();
5595      },
```

```
5596
5597        /**
5598         * Обновление входов формулы
5599         */
5600        updateFormulaInputs(elemId, inputCount) {
5601            const elem = document.getElementById(elemId);
5602            if (!elem) return;
5603
5604            const portsLeft = elem.querySelector('.ports-left');
5605            if (!portsLeft) return;
5606
5607            AppState.connections = AppState.connections.filter(c => {
5608                if (c.toElement === elemId && c.toPort.startsWith('in-')) {
5609                    const portNum = parseInt(c.toPort.split('-')[1], 10);
5610                    return portNum < inputCount;
5611                }
5612                return true;
5613            });
5614
5615            let inputsHTML = '';
5616            for (let i = 0; i < inputCount; i++) {
5617                inputsHTML += `
5618                    <div class="port input any-port"
5619                         data-port="in-${i}"
5620                         data-element="${elemId}"
5621                         data-signal-type="${SIGNAL_TYPE.ANY}"
5622                         title="in${i} (Любой)">
5623                    </div>
5624                `;
5625            }
5626            portsLeft.innerHTML = inputsHTML;
5627
5628            portsLeft.querySelectorAll('.port').forEach(port =>
5629                Connections.setupPortHandlers(port)
5630            );
5631
5632            Connections.drawConnections();
5633        },
5634
5635        /**
5636         * Рассчитать оптимальный размер элемента на основе количества портов
5637         */
5638        calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
5639            const elem = AppState.elements[elemId];
5640            if (!elem) return null;
5641
5642            const config = ELEMENT_TYPES[elem.type];
5643            if (!config || !config.resizable) return null;
5644
5645            // Базовая высота
5646            let baseHeight = config.minHeight || 60;
5647
5648            // Каждый порт требует примерно 25-30px высоты
5649            const portSpacing = 28;
5650            const maxPorts = Math.max(inputCount, outputCount);
5651
5652            // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
5653            const additionalHeight = (maxPorts - 1) * portSpacing;
5654            const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
5655
5656            return newHeight;
5657        },
5658
5659        /**
5660         * Обновление размера элемента при изменении портов
```

```
5661        */
5662       updateElementSize(elemId) {
5663           const elem = document.getElementById(elemId);
5664           const elemData = AppState.elements[elemId];
5665
5666           if (!elem || !elemData) return;
5667
5668           const config = ELEMENT_TYPES[elemData.type];
5669           if (!config || !config.resizable) return;
5670
5671           let inputCount = 0;
5672           let outputCount = config.outputs || 1;
5673
5674           // Определяем количество входов
5675           if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
      'formula') {
5676               inputCount = elemData.props.inputCount || config.inputs || 2;
5677           } else {
5678               inputCount = config.inputs || 0;
5679           }
5680
5681           // Рассчитываем новую высоту
5682           const newHeight = this.calculateOptimalHeight(elemId, inputCount,
      outputCount);
5683
5684           if (newHeight && newHeight !== elemData.height) {
5685               elemData.height = newHeight;
5686               elem.style.height = `${newHeight}px`;
5687
5688               // Перерисовываем соединения, т.к. изменился размер элемента
5689               Connections.drawConnections();
5690               Viewport.updateMinimap();
5691           }
5692       }
5693
5694
5695   };
5696
5697   /**
5698    * Модуль модальных окон
5699    * modal.js
5700    */
5701
5702   const Modal = {
5703       /**
5704        * Инициализация модальных окон
5705        */
5706       init() {
5707           // Модальное окно свойств элемента
5708           document.getElementById('modal-save').addEventListener('click', () => {
5709               this.saveElementProperties();
5710           });
5711
5712           document.getElementById('modal-cancel').addEventListener('click', () => {
5713               this.hideModal('modal-overlay');
5714           });
5715
5716           document.getElementById('modal-overlay').addEventListener('click', (e) => {
5717               if (e.target.id === 'modal-overlay') {
5718                   this.hideModal('modal-overlay');
5719               }
5720           });
5721
5722           // Модальное окно свойств проекта
5723           document.getElementById('project-modal-save').addEventListener('click', () =>
```

```
    {
5724            this.saveProjectProperties();
5725        });
5726
5727        document.getElementById('project-modal-cancel').addEventListener('click', ()
    => {
5728            this.hideModal('project-modal-overlay');
5729        });
5730
5731        document.getElementById('project-modal-overlay').addEventListener('click', (e)
    => {
5732            if (e.target.id === 'project-modal-overlay') {
5733                this.hideModal('project-modal-overlay');
5734            }
5735        });
5736    },
5737
5738    /**
5739     * Показать модальное окно
5740     */
5741    showModal(modalId) {
5742        document.getElementById(modalId).style.display = 'flex';
5743    },
5744
5745    /**
5746     * Скрыть модальное окно
5747     */
5748    hideModal(modalId) {
5749        document.getElementById(modalId).style.display = 'none';
5750        // Скрываем tooltip если он есть
5751        const tooltip = document.getElementById('template-tooltip');
5752        if (tooltip) {
5753            tooltip.classList.remove('visible');
5754        }
5755    },
5756
5757    /**
5758     * Показать свойства элемента
5759     */
5760    showPropertiesModal(elemId) {
5761        const elemData = AppState.elements[elemId];
5762        const elemType = elemData.type;
5763        const props = elemData.props;
5764        const config = ELEMENT_TYPES[elemType];
5765
5766        const modalOverlay = document.getElementById('modal-overlay');
5767        const modalTitle = document.getElementById('modal-title');
5768        const modalContent = document.getElementById('modal-content');
5769
5770        modalTitle.textContent = `Свойства: ${config.name}`;
5771
5772        let contentHTML = '';
5773
5774  if (elemType === 'input-signal') {
5775   const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
5776
5777   contentHTML = `
5778     <div class="modal-row">
5779       <label>Название сигнала:</label>
5780       <input type="text" id="prop-name" value="${props.name || ''}"
    placeholder="Например: 10LBA..." />
5781       <small style="color:#999;">
5782         Поиск по маске через * (например: *МАА*СР*)
5783       </small>
5784       <div id="signal-filter-results"
```

```
5785            style="max-height:160px; overflow-y:auto; background:#0f3460; border-
      radius:5px; margin-top:6px; display:none;">
5786          </div>
5787        </div>
5788
5789        <div class="modal-row">
5790          <label>Описание сигнала:</label>
5791          <textarea id="prop-description" readonly>${props.description || ''}</textarea>
5792        </div>
5793
5794        // modal.js в блоке input-signal
5795        <div class="modal-row">
5796            <label>Размерность:</label>
5797            <input type="text" id="prop-dimension" value="${props.dimension || ''}" />
5798        </div>
5799
5800        <div class="modal-row">
5801          <label>Тип сигнала:</label>
5802          <select id="prop-signal-type">
5803            <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
      'selected' : ''}>Числовой</option>
5804            <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
      'selected' : ''}>Логический</option>
5805          </select>
5806        </div>
5807      `;
5808
5809      // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
      DOM.
5810      // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
5811      // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
5812    } else if (elemType === 'if') {
5813                contentHTML = `
5814                    <div class="modal-row">
5815                        <label>Оператор сравнения:</label>
5816                        <select id="prop-operator">
5817                            <option value="=" ${props.operator === '=' ? 'selected' : ''}
      >=  (равно)</option>
5818                            <option value=">" ${props.operator === '>' ? 'selected' : ''}
      >>  (больше)</option>
5819                            <option value="<" ${props.operator === '<' ? 'selected' : ''}
      ><  (меньше)</option>
5820                            <option value=">=" ${props.operator === '>=' ? 'selected' :
      ''}>= (больше или равно)</option>
5821                            <option value="<=" ${props.operator === '<=' ? 'selected' :
      ''}>= (меньше или равно)</option>
5822                            <option value="!=" ${props.operator === '!=' ? 'selected' :
      ''}>!= (не равно)</option>
5823                        </select>
5824                    </div>
5825                `;
5826            } else if (elemType === 'and' || elemType === 'or') {
5827                contentHTML = `
5828                    <div class="modal-row">
5829                        <label>Количество входов:</label>
5830                        <input type="number" id="prop-input-count" value="$
      {props.inputCount || 2}" min="2" max="10">
5831                    </div>
5832                    <div class="modal-row">
5833                        <p style="color: #aaa; font-size: 12px;">
5834                            Измените количество входных портов для этого логического
      элемента.
5835                            Лишние соединения будут автоматически удалены.
5836                        </p>
5837                    </div>
```

```
5838             `;
5839         } else if (elemType === 'const') {
5840             contentHTML = `
5841                 <div class="modal-row">
5842                     <label>Значение:</label>
5843                     <input type="number" id="prop-value" value="${props.value ?? 0}"
      step="any">
5844                 </div>
5845             `;
5846         }
5847         else if (elemType === 'group') {
5848             contentHTML = `
5849             <div class="modal-row">
5850                 <label>Название группы:</label>
5851                 <input type="text" id="prop-title" value="${props.title || 'Группа'}">
5852             </div>`;
5853         }
5854 
5855         else if (elemType === 'formula') {
5856             let signalsHTML = '';
5857             AppState.connections.forEach(conn => {
5858                 if (conn.toElement === elemId) {
5859                     const fromElem = AppState.elements[conn.fromElement];
5860                     if (fromElem) {
5861                         const signalName = fromElem.props?.name || fromElem.id;
5862                         signalsHTML += `<div class="signal-item" data-signal="$
      {signalName}">${signalName} (${conn.toPort})</div>`;
5863                     }
5864                 }
5865             });
5866 
5867             // ... (где-то выше код сбора signalsHTML) ...
5868 
5869             contentHTML = `
5870                 <div class="modal-row">
5871                     <label>Количество входов:</label>
5872                     <input type="number" id="prop-input-count" value="$
      {props.inputCount || 2}" min="1" max="10">
5873                 </div>
5874 
5875                 <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->
5876                 <div style="display: flex; gap: 15px; margin-bottom: 15px; height:
      140px;">
5877                     <!-- Левая колонка: Сигналы -->
5878                     <div style="flex: 1; display: flex; flex-direction: column;">
5879                         <label style="margin-bottom: 5px; display:block;">Входные
      сигналы:</label>
5880                         <div class="signal-list" id="signal-list" style="flex: 1;
      overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px
      solid #4a90d9;">
5881                             ${signalsHTML || '<div style="color:#888;padding:5px;">Нет
      сигналов</div>'}
5882                         </div>
5883                     </div>
5884 
5885                     <!-- Правая колонка: Шаблоны -->
5886                     <div style="flex: 1; display: flex; flex-direction: column;">
5887                         <label style="margin-bottom: 5px; display:block;">Шаблоны:</
      label>
5888                         <div class="signal-list" id="template-list" style="flex: 1;
      overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px
      solid #4a90d9;">
5889                             <div style="color:#888;padding:5px;">Загрузка…</div>
5890                         </div>
5891                     </div>
```

```
5892                 </div>
5893
5894                 <!-- Нижний блок: Поле формулы (во всю ширину) -->
5895                 <div class="modal-row">
5896                     <label>Выражение формулы:</label>
5897                     <textarea id="prop-expression"
5898                             style="width: 100%; min-height: 80px; font-family:
      monospace; font-size: 14px; line-height: 1.4;"
5899                             spellcheck="false">${props.expression || ''}</textarea>
5900                     <small style="color:#999; display:block; margin-top:4px;">
5901                         Двойной клик по сигналу или шаблону вставит его в позицию
      курсора (или заменит выделенный текст).
5902                     </small>
5903                 </div>
5904             `;
5905         }
5906         if (!contentHTML) {
5907             contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных
      свойств.</div>`;
5908             }
5909         contentHTML += `
5910             <div class="modal-row">
5911                 <label>Комментарий:</label>
5912                 <textarea id="prop-comment" placeholder="Комментарий к элементу...">$
      {props.comment || ''}</textarea>
5913             </div>
5914             `;
5915
5916
5917         modalContent.innerHTML = contentHTML;
5918         // modal.js — внутри showPropertiesModal, блок if (elemType === 'formula')
5919         if (elemType === 'formula') {
5920             const listEl = document.getElementById('template-list');
5921
5922             // Создаём tooltip элемент (один на всю страницу)
5923             let tooltip = document.getElementById('template-tooltip');
5924             if (!tooltip) {
5925                 tooltip = document.createElement('div');
5926                 tooltip.id = 'template-tooltip';
5927                 tooltip.className = 'template-tooltip';
5928                 document.body.appendChild(tooltip);
5929             }
5930
5931             (async () => {
5932                 try {
5933                     const data = await Settings.fetchFormulaTemplates();
5934                     const items = data.templates || [];
5935
5936                     if (!items.length) {
5937                         listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет
      шаблонов</div>';
5938                         return;
5939                     }
5940
5941                     // Новый код
5942                     listEl.innerHTML = items.map(t => {
5943                         // --- НАЧАЛО ИЗМЕНЕНИЙ ---
5944                         let argList = [];
5945
5946                         if (Array.isArray(t.args)) {
5947                             // Если пришел старый формат (массив): ["p", "t"]
5948                             argList = t.args;
5949                         } else if (t.args && typeof t.args === 'object') {
5950                             // Если пришел новый формат (объект): {"p": {...}, "t":
      {...}}
```

```
                            // Берем только ключи (имена переменных)
                            argList = Object.keys(t.args);
                        }

                        // Формируем подпись функции: h(p, t)
                        const sig = `${t.name}(${argList.join(', ')})`;
                        // --- КОНЕЦ ИЗМЕНЕНИЙ ---

                        const desc = (t.description || '').replace(/"/g, '&quot;');
                        return `<div class="signal-item template-item"
                                data-insert="${sig}"
                                data-name="${t.name}"
                                data-description="${desc}">${sig}</div>`;
                    }).join('');

                    // Обработчики для каждого шаблона
                    listEl.querySelectorAll('.template-item').forEach(div => {
                        // Двойной клик — вставка
                        div.addEventListener('dblclick', () => {
                            const insert = div.dataset.insert;
                            const textarea = document.getElementById('prop-
expression');
                            insertAtCursor(textarea, insert);
                        });

                        // 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');
```

```
6014
6015            let timer = null;
6016
6017            const renderList = (items) => {
6018                if (!items || items.length === 0) {
6019                results.innerHTML = '<div style="color:#666;padding:6px;">Нет
    совпадений</div>';
6020                results.style.display = 'block';
6021                return;
6022                }
6023
6024                results.innerHTML = items.map(s => `
6025                <div class="signal-result-item"
6026                    style="padding:6px 8px; cursor:pointer; border-bottom:1px solid
    rgba(255,255,255,0.08);">
6027                    <div style="font-weight:600;">${s.Tagname}</div>
6028                    <div style="color:#aaa; font-size:11px;">${s.Description || ''}</
    div>
6029                </div>
6030                `).join('');
6031
6032                results.style.display = 'block';
6033
6034                results.querySelectorAll('.signal-result-item').forEach((div, i) => {
6035                div.addEventListener('click', () => {
6036                    const chosen = items[i];
6037                    input.value = chosen.Tagname;
6038                    descField.value = chosen.Description || '';
6039                    const dimField = document.getElementById('prop-dimension');
6040                    if (dimField) dimField.value = chosen.EngineeringUnit ||
    chosen.Dimension || '';
6041                    results.style.display = 'none';
6042                });
6043                });
6044            };
6045
6046            const search = async () => {
6047                const mask = (input.value || '').trim();
6048
6049                // Показываем список только если пользователь реально использует маску
6050                if (!mask.includes('*')) {
6051                results.style.display = 'none';
6052                return;
6053                }
6054
6055                results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</
    div>';
6056                results.style.display = 'block';
6057
6058                try {
6059                // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
6060                const data = await Settings.fetchSignals(mask, 50);
6061                renderList(data.items || []);
6062                } catch (e) {
6063                results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка
    загрузки сигналов</div>';
6064                results.style.display = 'block';
6065                console.error(e);
6066                }
6067            };
6068
6069            input.addEventListener('input', () => {
6070                clearTimeout(timer);
6071                timer = setTimeout(search, 200); // debounce
6072            });
```

```
6073
6074            // опционально: закрывать список кликом вне
6075            document.addEventListener('mousedown', (e) => {
6076                if (!results.contains(e.target) && e.target !== input) {
6077                results.style.display = 'none';
6078                }
6079            }, { once: true });
6080        }
6081        modalOverlay.dataset.elementId = elemId;
6082        this.showModal('modal-overlay');
6083
6084        // Функция для умной вставки текста в позицию курсора
6085        const insertAtCursor = (field, text) => {
6086            if (!field) return;
6087
6088            // Получаем позиции выделения
6089            const startPos = field.selectionStart;
6090            const endPos = field.selectionEnd;
6091            const currentValue = field.value;
6092
6093            // Вставляем текст: (текст до) + (новый текст) + (текст после)
6094            field.value = currentValue.substring(0, startPos) +
6095                        text +
6096                        currentValue.substring(endPos, currentValue.length);
6097
6098            // Возвращаем фокус и ставим курсор сразу после вставленного текста
6099            field.focus();
6100            const newCursorPos = startPos + text.length;
6101            field.setSelectionRange(newCursorPos, newCursorPos);
6102        };
6103
6104        // Обработчик вставки сигналов для формулы
6105        if (elemType === 'formula') {
6106            document.querySelectorAll('.signal-item').forEach(item => {
6107            item.addEventListener('dblclick', () => {
6108                const signal = item.dataset.signal;
6109                const textarea = document.getElementById('prop-expression');
6110
6111                // БЫЛО: textarea.value += signal;
6112                // СТАЛО:
6113                insertAtCursor(textarea, signal);
6114                });
6115            });
6116        }
6117    },
6118
6119    /**
6120     * Сохранить свойства элемента
6121     */
6122  /**
6123   * Сохранить свойства элемента
6124   */
6125    saveElementProperties() {
6126        try {
6127            const modalOverlay = document.getElementById('modal-overlay');
6128            const elemId = modalOverlay.dataset.elementId;
6129            const elemData = AppState.elements[elemId];
6130            const elem = document.getElementById(elemId);
6131            if (!elemData) {
6132                alert('⚠ Элемент не найден — возможно, он был удалён или
    переименован.');
6133                console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
6134                this.hideModal('modal-overlay');
6135                return;
6136            }
```

```
6137
6138            const elemType = elemData.type;
6139
6140            if (elemType === 'input-signal') {
6141                const name = document.getElementById('prop-name').value || 'Сигнал';
6142                const description = document.getElementById('prop-description').value
      || '';
6143                const signalType = document.getElementById('prop-signal-type').value;
6144                const dimension = document.getElementById('prop-dimension').value ||
      '';
6145                elemData.props.dimension = dimension;
6146
6147                const oldSignalType = elemData.props.signalType;
6148                elemData.props.name = name;
6149                elemData.props.description = description;
6150                elemData.props.signalType = signalType;
6151
6152                if (oldSignalType !== signalType) {
6153                    AppState.connections = AppState.connections.filter(conn => {
6154                    if (conn.fromElement === elemId) {
6155                        const toPortIndex = parseInt(conn.toPort.split('-')[1]);
6156                        const inputType = getInputPortType(conn.toElement,
      toPortIndex);
6157                        return areTypesCompatible(signalType, inputType);
6158                    }
6159                    return true;
6160                    });
6161                }
6162
6163                const { html } = Elements.createElementHTML(
6164                    elemType, elemId, elemData.x, elemData.y, elemData.props,
      elemData.width, elemData.height
6165                );
6166                elem.outerHTML = html;
6167
6168                Elements.setupElementHandlers(elemId);
6169                Connections.drawConnections();
6170            } else if (elemType === 'if') {
6171                const operator = document.getElementById('prop-operator').value;
6172                elemData.props.operator = operator;
6173                const symbol = elem.querySelector('.element-symbol');
6174                if (symbol) symbol.textContent = operator;
6175
6176            } else if (elemType === 'const') {
6177                const value = parseFloat(document.getElementById('prop-value').value)
      || 0;
6178                elemData.props.value = value;
6179                const symbol = elem.querySelector('.element-symbol');
6180                if (symbol) symbol.textContent = String(value);
6181
6182            } else if (elemType === 'formula') {
6183                const expression = document.getElementById('prop-expression').value;
6184                const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
6185
6186                elemData.props.expression = expression;
6187                elemData.props.inputCount = inputCount;
6188
6189                const symbol = elem.querySelector('.element-symbol');
6190                if (symbol) {
6191                    symbol.textContent = expression.length > 12 ? `$
      {expression.slice(0, 12)}…` : (expression || 'f(x)');
6192                }
6193
6194                Elements.updateFormulaInputs(elemId, inputCount);
```

```
6195                 Elements.updateElementSize(elemId);  // ← Добавляем это
6196             } else if (elemType === 'and' || elemType === 'or') {
6197                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
6198                 elemData.props.inputCount = inputCount;
6199 
6200                 Elements.updateLogicGateInputs(elemId, inputCount);
6201                 Elements.updateElementSize(elemId);  // ← Добавляем это
6202 
6203                 const symbol = elem.querySelector('.element-symbol');
6204                 if (symbol) {
6205                     symbol.textContent = elemType === 'and' ? '∧' : '∨';
6206                 }
6207 
6208         } else if (elemType === 'output') {
6209                 const label = document.getElementById('prop-label').value || 'Выход';
6210                 const outputGroup = document.getElementById('prop-output-group').value
      || '';
6211 
6212                 elemData.props.label = label;
6213                 elemData.props.outputGroup = outputGroup;
6214 
6215                 const symbol = elem.querySelector('.element-symbol');
6216                 if (symbol) symbol.textContent = label;
6217             }
6218             else if (elemType === 'group') {
6219                 const title = document.getElementById('prop-title').value || 'Группа';
6220                 elemData.props.title = title;
6221                 const titleEl = elem.querySelector('.group-title');
6222                 if (titleEl) titleEl.textContent = title;
6223                 }
6224             const commentEl = document.getElementById('prop-comment');
6225             if (commentEl) elemData.props.comment = commentEl.value || '';
6226 
6227             this.hideModal('modal-overlay');
6228 
6229         } catch (error) {
6230             console.error('❌ Ошибка при сохранении свойств:', error);
6231             alert('Ошибка сохранения: ' + error.message);
6232         }
6233     },
6234 
6235     /**
6236      * Показать свойства проекта
6237      */
6238     showProjectPropertiesModal() {
6239         const content = document.getElementById('project-modal-content');
6240         const project = AppState.project;
6241 
6242         // Генерируем HTML для списка выходов только если модуль загружен
6243         let outputsHtml = '';
6244         if (typeof Outputs !== 'undefined' && AppState.outputs) {
6245             const logicalOutputsHtml = AppState.outputs.logical.length > 0
6246                 ? AppState.outputs.logical.map(output => `
6247                     <div class="output-item"
6248                         data-element-id="${output.elementId}"
6249                         onmouseenter="Outputs.highlightOutput('${output.elementId}',
      true)"
6250                         onmouseleave="Outputs.highlightOutput('${output.elementId}',
      false)"
6251                         onclick="Outputs.navigateToOutput('${output.elementId}');
      Modal.hideModal('project-modal-overlay');">
6252                         <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
      '❌'}</span>
6253                         <span class="output-name">${output.elementName}</span>
```

```
6254                         <span class="output-port">→ ${output.portLabel}</span>
6255                     </div>
6256                 `).join('')
6257                 : '<div class="no-outputs">Нет логических выходов</div>';
6258 
6259             const numericOutputsHtml = AppState.outputs.numeric.length > 0
6260                 ? AppState.outputs.numeric.map(output => `
6261                     <div class="output-item numeric"
6262                         data-element-id="${output.elementId}"
6263                         onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
6264                         onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
6265                         onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
6266                         <span class="output-icon">🔢</span>
6267                         <span class="output-name">${output.elementName}</span>
6268                         <span class="output-port">→ значение</span>
6269                     </div>
6270                 `).join('')
6271                 : '<div class="no-outputs">Нет числовых выходов</div>';
6272 
6273             outputsHtml = `
6274                 <div class="modal-row">
6275                     <label>Выходные сигналы схемы:</label>
6276                     <div class="outputs-container">
6277                         <div class="outputs-section">
6278                             <div class="outputs-section-title">
6279                                 <span class="section-icon">🔀</span>
6280                                 Логические выходы (${AppState.outputs.logical.length})
6281                             </div>
6282                             <div class="outputs-list">
6283                                 ${logicalOutputsHtml}
6284                             </div>
6285                         </div>
6286                         <div class="outputs-section">
6287                             <div class="outputs-section-title">
6288                                 <span class="section-icon">📐</span>
6289                                 Числовые выходы (${AppState.outputs.numeric.length})
6290                             </div>
6291                             <div class="outputs-list">
6292                                 ${numericOutputsHtml}
6293                             </div>
6294                         </div>
6295                     </div>
6296                     <div class="outputs-hint">
6297                         💡 Выходами автоматически становятся элементы, чьи выходные
    порты не подключены к другим элементам.
6298                         Кликните на выход, чтобы перейти к нему на схеме.
6299                     </div>
6300                 </div>
6301             `;
6302         }
6303 
6304         content.innerHTML = `
6305             <div class="modal-row">
6306                 <label>Код проекта:</label>
6307                 <input type="text" id="project-code" value="${project.code || ''}"
    placeholder="Уникальный идентификатор">
6308             </div>
6309 
6310             <div class="modal-row">
6311                 <label>Тип проекта:</label>
6312                 <div class="project-type-selector">
6313                     <div class="project-type-btn ${project.type ===
```

```
    PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
6314                        <div class="type-icon">📊</div>
6315                        <div class="type-name">Параметр</div>
6316                        <div class="type-desc">Вычисляемое значение</div>
6317                    </div>
6318                    <div class="project-type-btn ${project.type ===
    PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
6319                        <div class="type-icon">📋</div>
6320                        <div class="type-name">Правило</div>
6321                        <div class="type-desc">Логическое условие</div>
6322                    </div>
6323                </div>
6324            </div>
6325
6326            <div id="parameter-fields" class="conditional-fields ${project.type ===
    PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
6327                <div class="modal-row">
6328                    <label>Описание:</label>
6329                    <textarea id="project-description" placeholder="Описание
    сигнала">${project.description || ''}</textarea>
6330                </div>
6331                <div class="modal-row">
6332                    <label>Размерность:</label>
6333                    <input type="text" id="project-dimension" value="$
    {project.dimension || ''}" placeholder="Например: м/с, кг, °C">
6334                </div>
6335            </div>
6336
6337            <div id="rule-fields" class="conditional-fields ${project.type ===
    PROJECT_TYPE.RULE ? 'visible' : ''}">
6338                <div class="modal-row">
6339                    <label>Возможная причина:</label>
6340                    <textarea id="project-possible-cause" placeholder="Описание
    возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
6341                </div>
6342                <div class="modal-row">
6343                    <label>Методические указания:</label>
6344                    <textarea id="project-guidelines" placeholder="Инструкции и
    рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
6345                </div>
6346            </div>
6347
6348            ${outputsHtml}
6349        `;
6350
6351        // Обработчики переключения типа
6352        content.querySelectorAll('.project-type-btn').forEach(btn => {
6353            btn.addEventListener('click', () => {
6354                content.querySelectorAll('.project-type-btn').forEach(b =>
    b.classList.remove('active'));
6355                btn.classList.add('active');
6356
6357                const type = btn.dataset.type;
6358                document.getElementById('parameter-
    fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
6359                document.getElementById('rule-fields').classList.toggle('visible',
    type === PROJECT_TYPE.RULE);
6360            });
6361        });
6362
6363        this.showModal('project-modal-overlay');
6364    },
6365
6366    /**
6367     * Сохранить свойства проекта
```

```
6368        */
6369       saveProjectProperties() {
6370           const activeTypeBtn = document.querySelector('.project-type-btn.active');
6371           const type = activeTypeBtn ? activeTypeBtn.dataset.type :
     PROJECT_TYPE.PARAMETER;
6372
6373           AppState.project.code = document.getElementById('project-code').value;
6374           AppState.project.type = type;
6375
6376           if (type === PROJECT_TYPE.PARAMETER) {
6377               AppState.project.dimension = document.getElementById('project-
     dimension').value;
6378               AppState.project.description = document.getElementById('project-
     description').value || '';
6379               AppState.project.possibleCause = '';
6380               AppState.project.guidelines = '';
6381           } else {
6382               AppState.project.dimension = '';
6383               AppState.project.description = '';
6384               AppState.project.possibleCause = document.getElementById('project-
     possible-cause').value;
6385               AppState.project.guidelines = document.getElementById('project-
     guidelines').value;
6386           }
6387
6388           this.hideModal('project-modal-overlay');
6389       }
6390   };
6391
6392   /**
6393    * Модуль управления выходными сигналами
6394    */
6395
6396   const Outputs = {
6397       /**
6398        * Обновление статуса выходных элементов
6399        * Вызывается при каждом изменении схемы
6400        */
6401       updateOutputStatus() {
6402           this.clearAllOutputHighlights();
6403           AppState.outputs.logical = [];
6404           AppState.outputs.numeric = [];
6405           updateFrameChildren();
6406
6407           // Обработка элементов-выходов
6408           Object.values(AppState.elements).forEach(elem => {
6409               if (!elem || elem.type !== 'output') return;
6410
6411               // Проверяем, к чему подключен вход этого выхода
6412               const inputConns = AppState.connections.filter(c =>
6413                   c.toElement === elem.id && c.toPort === 'in-0'
6414               );
6415
6416               // Каждое соединение к выходу — это отдельный выход
6417               inputConns.forEach((conn, index) => {
6418                   const fromElem = AppState.elements[conn.fromElement];
6419                   if (!fromElem) return;
6420
6421                   const outputType = conn.signalType;
6422                   const outputInfo = {
6423                       id: `${elem.id}_conn_${index}`,
6424                       elementId: elem.id,
6425                       sourceElementId: conn.fromElement,
6426                       sourcePort: conn.fromPort,
6427                       portIndex: 0,
```

```
                    portId: 'in-0',
                    type: outputType,
                    label: elem.props?.label || 'Выход',
                    elementType: 'output',
                    elementName: elem.props?.label || 'Выход',
                    name: elem.props?.label || 'Выход'
                };

                if (outputType === SIGNAL_TYPE.LOGIC) {
                    AppState.outputs.logical.push(outputInfo);
                } else if (outputType === SIGNAL_TYPE.NUMBER) {
                    AppState.outputs.numeric.push(outputInfo);
                }

                // Подсветим входной порт
                this.highlightOutputPort(elem.id, 0, outputType);
            });
        });

        this.updateOutputCounter();
    },

    /**
     * Очистка всех выделений выходов
     */
    clearAllOutputHighlights() {
        document.querySelectorAll('.port.output-active').forEach(port => {
            port.classList.remove('output-active');
        });

        document.querySelectorAll('.element.has-output').forEach(elem => {
            elem.classList.remove('has-output');
        });

        document.querySelectorAll('.element.output-ambiguous').forEach(el =>
el.classList.remove('output-ambiguous'));
        document.querySelectorAll('.element.output-missing').forEach(el =>
el.classList.remove('output-missing'));
    },

    /**
     * Выделение выходного порта
     */
    highlightOutputPort(elemId, portIndex, portType) {
        const elem = document.getElementById(elemId);
        if (!elem) return;

        const port = elem.querySelector(`.port.output[data-port="out-${portIndex}"]`);
        if (port) {
            port.classList.add('output-active');
        }

        // Добавляем класс элементу (даёт общий визуал)
        elem.classList.add('has-output');
    },

    /**
     * Обновление счётчика выходов в меню
     */
    updateOutputCounter() {
        const counter = document.getElementById('output-counter');
        if (counter) {
            const total = AppState.outputs.logical.length +
AppState.outputs.numeric.length;
            counter.textContent = total;
```

```
6490              counter.style.display = total > 0 ? 'inline-block' : 'none';
6491          }
6492      },
6493
6494      /**
6495       * Получить все выходы для сохранения в проект
6496       */
6497      getOutputsForSave() {
6498          // Сохраняем информацию о frame/inner для рамок
6499          return {
6500              logical: AppState.outputs.logical.map(o => ({
6501                  id: o.id,
6502                  elementId: o.elementId,
6503                  frameId: o.frameId || null,
6504                  innerElementId: o.innerElementId || null,
6505                  portIndex: o.portIndex ?? o.innerPortIndex ?? null,
6506                  portLabel: o.label
6507              })),
6508              numeric: AppState.outputs.numeric.map(o => ({
6509                  id: o.id,
6510                  elementId: o.elementId,
6511                  frameId: o.frameId || null,
6512                  innerElementId: o.innerElementId || null,
6513                  portIndex: o.portIndex ?? o.innerPortIndex ?? null,
6514                  portLabel: o.label
6515              }))
6516          };
6517      },
6518
6519      /**
6520       * Подсветить конкретный выход (при наведении в списке)
6521       */
6522      highlightOutput(elementId, highlight = true) {
6523          const elem = document.getElementById(elementId);
6524          if (elem) {
6525              if (highlight) {
6526                  elem.classList.add('output-highlighted');
6527              } else {
6528                  elem.classList.remove('output-highlighted');
6529              }
6530          }
6531      },
6532
6533      /**
6534       * Перейти к элементу выхода на схеме (elementId — фокусируемый элемент; для рамок
      это id рамки)
6535       */
6536      navigateToOutput(elementId) {
6537          const elemData = AppState.elements[elementId];
6538          if (!elemData) return;
6539
6540          // Центрируем viewport на элементе
6541          const container = document.getElementById('workspace-container');
6542          const rect = container.getBoundingClientRect();
6543
6544          const centerX = elemData.x + elemData.width / 2;
6545          const centerY = elemData.y + elemData.height / 2;
6546
6547          AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
6548          AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;
6549
6550          Viewport.updateTransform();
6551
6552          // Выделяем элемент
6553          Elements.selectElement(elementId);
```

```
6554
6555          // Временная подсветка
6556          this.highlightOutput(elementId, true);
6557          setTimeout(() => this.highlightOutput(elementId, false), 2000);
6558      }
6559  };
6560
6561  /**
6562   * Модуль управления проектом (сохранение, загрузка)
6563   * project.js
6564   */
6565
6566  // --- миграция id: '-' -> '_' с обновлением всех ссылок ---
6567  function migrateIdsDashToUnderscore() {
6568    const map = {};
6569
6570    // 1) собрать map старых id → новых
6571    Object.values(AppState.elements).forEach(el => {
6572      if (typeof el.id === 'string' && el.id.includes('-')) {
6573        map[el.id] = el.id.replace(/-/g, '_');
6574      }
6575    });
6576
6577    if (!Object.keys(map).length) return;
6578
6579    // 2) DOM id + data-element
6580    Object.entries(map).forEach(([oldId, newId]) => {
6581      const dom = document.getElementById(oldId);
6582      if (dom) dom.id = newId;
6583
6584      if (dom) {
6585        dom.querySelectorAll('[data-element]').forEach(p => {
6586          if (p.dataset.element === oldId) p.dataset.element = newId;
6587        });
6588      }
6589    });
6590
6591    // 3) AppState.elements ключи
6592    Object.entries(map).forEach(([oldId, newId]) => {
6593      const el = AppState.elements[oldId];
6594      if (!el) return;
6595      el.id = newId;
6596      AppState.elements[newId] = el;
6597      delete AppState.elements[oldId];
6598    });
6599
6600    // 4) connections
6601    AppState.connections.forEach(c => {
6602      if (map[c.fromElement]) c.fromElement = map[c.fromElement];
6603      if (map[c.toElement]) c.toElement = map[c.toElement];
6604    });
6605
6606    // 5) формулы
6607    const escapeRegex = s => s.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
6608    Object.values(AppState.elements).forEach(el => {
6609      if (el.type === 'formula' && el.props?.expression) {
6610        let expr = el.props.expression;
6611        Object.entries(map).forEach(([oldId, newId]) => {
6612          const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
      z0-9_])`, 'g');
6613          expr = expr.replace(re, (m, p1) => `${p1}${newId}`);
6614        });
6615        el.props.expression = expr;
6616      }
6617    });
```

```
6618
6619    // 6) selected + modal
6620    if (map[AppState.selectedElement]) AppState.selectedElement =
      map[AppState.selectedElement];
6621    const modal = document.getElementById('modal-overlay');
6622    if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
      map[modal.dataset.elementId];
6623  }
6624
6625  const Project = {
6626      /**
6627       * Инициализация
6628       */
6629          /**
6630       * Инициализация
6631       */
6632  init() {
6633      document.getElementById('btn-new').addEventListener('click', () =>
      this.newProject());
6634      document.getElementById('btn-save').addEventListener('click', () =>
      this.saveProject());
6635      document.getElementById('btn-load').addEventListener('click', () =>
      this.openProjectListModal());
6636      document.getElementById('btn-project-settings').addEventListener('click', () => {
6637          Modal.showProjectPropertiesModal();
6638      });
6639
6640      // Работа с модалкой выбора проекта
6641      this.projectList = [];
6642      this.filteredProjectList = [];
6643      this.selectedProjectFilename = null;
6644
6645      document.getElementById('project-cancel').addEventListener('click', () =>
      this.closeProjectListModal());
6646      document.getElementById('project-refresh').addEventListener('click', () =>
      this.refreshProjectList());
6647
6648      document.getElementById('project-load').addEventListener('click', () => {
6649          if (this.selectedProjectFilename) {
6650              this.loadProjectFromList(this.selectedProjectFilename);
6651          }
6652      });
6653
6654      document.getElementById('project-search').addEventListener('input', (event) => {
6655          this.filterProjectList(event.target.value);
6656      });
6657  },
6658
6659      /**
6660       * Новый проект
6661       */
6662      newProject() {
6663          if (Object.keys(AppState.elements).length > 0) {
6664              if (!confirm('Создать новый проект? Несохранённые изменения будут
      потеряны.')) {
6665                  return;
6666              }
6667          }
6668
6669          document.getElementById('workspace').innerHTML = '';
6670          document.getElementById('connections-svg').innerHTML = '';
6671
6672          resetState();
6673          Viewport.updateTransform();
6674      },
```

```
6675
6676          /**
6677       * Запрос имени файла и загрузка с сервера
6678       */
6679      async loadProjectPrompt() {
6680          const filename = window.prompt(
6681              "Введите имя файла проекта для загрузки (с сервера). Пример:
      scheme_logic.json",
6682              AppState.project.code ? `${AppState.project.code}_$
      {AppState.project.type}.json` : "scheme_type.json"
6683          );
6684
6685          if (!filename) return; // Отмена
6686
6687          try {
6688              // Используем обертку из Settings.js для запроса к /api/project/load
6689              const data = await Settings.loadProject(filename);
6690
6691              // Если загрузка успешна, вызываем основную функцию обработки данных
6692              this._processLoadedData(data);
6693              alert(`Проект "${filename}" успешно загружен с сервера.`);
6694
6695          } catch (error) {
6696              console.error('Ошибка загрузки проекта:', error);
6697              alert(`Ошибка загрузки проекта: ${error.message}`);
6698          }
6699      },
6700
6701      /**
6702       * Сохранение проекта
6703       */
6704          async saveProject() {
6705      // 1. Проверяем свойства проекта
6706      if (!AppState.project.code) {
6707          Modal.showProjectPropertiesModal();
6708          alert('Пожалуйста, укажите код проекта перед сохранением.');
6709          return;
6710      }
6711
6712      // Обновляем размеры рамок перед сохранением
6713      updateFrameChildren();
6714      // нормализуем id
6715      migrateIdsDashToUnderscore();
6716
6717      // подчистим связи прямо перед сохранением
6718      const exists = (id) => !!AppState.elements[id];
6719      AppState.connections = (AppState.connections || [])
6720        .map(c => ({
6721          ...c,
6722          fromElement: exists(c.fromElement) ? c.fromElement : c.fromElement.replace(/-/
      g, '_'),
6723          toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g, '_')
6724        }))
6725        .filter(c => exists(c.fromElement) && exists(c.toElement))
6726        .filter((c, idx, arr) => {
6727          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
6728          return arr.findIndex(x =>
6729            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
6730          ) === idx;
6731        });
6732
6733      // Генерируем код заранее
6734      let generatedCode = '';
6735      if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function') {
6736          try {
```

```
            generatedCode = CodeGen.generate() || '';
        } catch (err) {
            console.error('Code generation failed:', err);
        }
    }

    // НОВОЕ: получаем состояние визуализатора перед сохранением
    let visualizerState = AppState.project?.visualizer_state || null;

    if (AppState.currentVisualizerToken) {
        try {
            const freshState = await App.fetchVisualizerState();
            if (freshState) {
                visualizerState = freshState;
                console.log('Состояние визуализатора обновлено перед сохранением');
            }
        } catch (err) {
            console.warn('Не удалось получить состояние визуализатора:', err);
        }
    }

    // 2. Сборка объекта проекта
    const project = {
        version: '1.0',
        project: AppState.project,
        elements: AppState.elements,
        connections: AppState.connections,
        counter: AppState.elementCounter,
        viewport: {
            zoom: AppState.viewport.zoom,
            panX: AppState.viewport.panX,
            panY: AppState.viewport.panY
        },
        code: generatedCode,
        visualizer_state: visualizerState  // НОВОЕ: сохраняем состояние визуализатора
    };

    const filename = `${AppState.project.code || 'scheme'}_$
{AppState.project.type}.json`;

    // 3. Сохранение на сервер
    try {
        await Settings.saveProject(filename, project);

        // НОВОЕ: обновляем состояние в AppState после успешного сохранения
        AppState.project.visualizer_state = visualizerState;

        alert(`Проект успешно сохранен на сервере как: ${filename}`);
    } catch (error) {
        console.error('Ошибка сохранения проекта:', error);
        alert(`Ошибка сохранения проекта: ${error.message}`);
    }
},

    async showProjectList() {
        try {
            const result = await Settings.listProjects(); // нужно реализовать в
settings.js
            const list = result.projects || [];

            if (list.length === 0) {
                alert('Проекты в папке не найдены.');
                return;
            }

```

```
6800              const choice = window.prompt(
6801                  'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code ||
      p.filename} — ${p.description}`).join('\n') +
6802                  '\n\nВведите номер проекта для загрузки:',
6803                  '1'
6804              );
6805              const index = parseInt(choice, 10) - 1;
6806              if (isNaN(index) || !list[index]) return;
6807
6808              await this.loadProjectByFilename(list[index].filename);
6809          } catch (error) {
6810              console.error(error);
6811              alert('Не удалось получить список проектов: ' + error.message);
6812          }
6813      },
6814
6815      async loadProjectByFilename(filename) {
6816          try {
6817              const data = await Settings.loadProject(filename);
6818              this._processLoadedData(data);
6819              alert(`Проект "${filename}" загружен.`);
6820          } catch (error) {
6821              console.error(error);
6822              alert('Ошибка загрузки проекта: ' + error.message);
6823          }
6824      },
6825
6826      openProjectListModal() {
6827    const modal = document.getElementById('modal-project-list');
6828    modal.classList.remove('hidden');
6829    document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
6830    this.refreshProjectList();
6831  },
6832
6833  closeProjectListModal() {
6834    const modal = document.getElementById('modal-project-list');
6835    modal.classList.add('hidden');
6836    document.body.classList.remove('modal-open');
6837  },
6838
6839  async refreshProjectList() {
6840    const tbody = document.getElementById('project-list-body');
6841    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
6842    try {
6843      const result = await Settings.listProjects();
6844      this.projectList = result.projects || [];
6845      this.filteredProjectList = [...this.projectList];
6846      this.renderProjectList();
6847    } catch (err) {
6848      console.error(err);
6849      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
6850    }
6851  },
6852
6853  renderProjectList() {
6854    const tbody = document.getElementById('project-list-body');
6855    const loadBtn = document.getElementById('project-load');
6856    loadBtn.disabled = true;
6857    this.selectedProjectFilename = null;
6858
6859    if (!this.filteredProjectList.length) {
6860      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
```

```
      найдено</td></tr>`;
6861      return;
6862    }
6863
6864    tbody.innerHTML = '';
6865    this.filteredProjectList.forEach((item) => {
6866      const tr = document.createElement('tr');
6867      tr.innerHTML = `
6868        <td>${item.filename}</td>
6869        <td>${item.code || ''}</td>
6870        <td>${item.description || ''}</td>
6871        <td>${item.type || ''}</td>
6872      `;
6873      tr.addEventListener('click', () => {
6874        this.highlightRow(tr);
6875        this.selectedProjectFilename = item.filename;
6876        loadBtn.disabled = false;
6877      });
6878      tr.addEventListener('dblclick', () => {
6879        this.highlightRow(tr);
6880        this.selectedProjectFilename = item.filename;
6881        loadBtn.disabled = false;
6882        this.loadProjectFromList(item.filename);
6883      });
6884      tbody.appendChild(tr);
6885    });
6886  },
6887
6888  highlightRow(row) {
6889    const tbody = row.parentElement;
6890    [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
6891    row.classList.add('selected');
6892  },
6893
6894
6895  // Фильтр по поисковой строке
6896  filterProjectList(query) {
6897    const q = (query || '').trim().toLowerCase();
6898    if (!q) {
6899      this.filteredProjectList = [...this.projectList];
6900    } else {
6901      this.filteredProjectList = this.projectList.filter((item) => {
6902        return [
6903          item.filename,
6904          item.code,
6905          item.description,
6906          item.type
6907        ].some((field) => (field || '').toLowerCase().includes(q));
6908      });
6909    }
6910    this.renderProjectList();
6911  },
6912
6913  async loadProjectFromList(filename) {
6914    try {
6915      const data = await Settings.loadProject(filename);
6916      this._processLoadedData(data);
6917      this.closeProjectListModal();
6918      alert(`Проект "${filename}" успешно загружен.`);
6919    } catch (error) {
6920      console.error(error);
6921      alert('Ошибка загрузки проекта: ' + error.message);
6922    }
6923  },
6924
```

```
6925
6926
6927
6928
6929
6930      /**
6931       * Загрузка проекта
6932       */
6933  _processLoadedData(data) {
6934    try {
6935      document.getElementById('workspace').innerHTML = '';
6936      document.getElementById('connections-svg').innerHTML = '';
6937      resetState();
6938
6939      if (data.project) {
6940        AppState.project = { ...AppState.project, ...data.project };
6941      }
6942
6943      // НОВОЕ: загружаем состояние визуализатора
6944      if (data.visualizer_state) {
6945        AppState.project.visualizer_state = data.visualizer_state;
6946        console.log('Загружено состояние визуализатора:', data.visualizer_state);
6947      } else {
6948        AppState.project.visualizer_state = null;
6949      }
6950
6951      // НОВОЕ: сбрасываем токен предыдущей сессии визуализатора
6952      AppState.currentVisualizerToken = null;
6953
6954      AppState.elementCounter = data.counter || 0;
6955
6956      if (data.viewport) {
6957        AppState.viewport.zoom = data.viewport.zoom || 1;
6958        AppState.viewport.panX = data.viewport.panX || 0;
6959        AppState.viewport.panY = data.viewport.panY || 0;
6960      }
6961
6962      const elements = data.elements || {};
6963      Object.values(elements)
6964        .filter(e => e.type === 'output-frame')
6965        .forEach(elemData => {
6966          Elements.addElement(
6967            elemData.type,
6968            elemData.x,
6969            elemData.y,
6970            elemData.props,
6971            elemData.id,
6972            elemData.width,
6973            elemData.height
6974          );
6975        });
6976
6977      Object.values(elements)
6978        .filter(e => e.type !== 'output-frame')
6979        .forEach(elemData => {
6980          Elements.addElement(
6981            elemData.type,
6982            elemData.x,
6983            elemData.y,
6984            elemData.props,
6985            elemData.id,
6986            elemData.width,
6987            elemData.height
6988          );
6989        });
```

```
6990
6991      AppState.connections = data.connections || [];
6992
6993      // Миграция id: '-' -> '_'
6994      migrateIdsDashToUnderscore();
6995
6996      // очистка соединений: удалить битые и дубликаты
6997      const exists = (id) => !!AppState.elements[id];
6998
6999      AppState.connections = (AppState.connections || [])
7000        .filter(c => exists(c.fromElement) && exists(c.toElement))
7001        .filter((c, idx, arr) => {
7002          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
7003          return arr.findIndex(x =>
7004            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
7005          ) === idx;
7006        });
7007
7008      // корректно восстанавливаем счётчик
7009      const counterFromFile = Number(data.counter);
7010      AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
7011
7012      const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
7013          if (!el?.id) return max;
7014          const match = String(el.id).match(/_(\d+)$/);
7015          const num = match ? parseInt(match[1], 10) : NaN;
7016          return Number.isFinite(num) ? Math.max(max, num) : max;
7017      }, 0);
7018
7019      AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
7020
7021      Viewport.updateTransform();
7022      Connections.drawConnections();
7023      updateFrameChildren();
7024
7025    } catch (e) {
7026      alert('Ошибка обработки данных проекта: ' + e.message);
7027      console.error(e);
7028    }
7029  }
7030  };
7031
7032  // settings.js — ПОЛНАЯ ИСПРАВЛЕННАЯ ВЕРСИЯ
7033
7034  const Settings = {
7035    config: null,
7036    templates: null,
7037    apiUrl: '',  // ← Добавь это! Пустая строка = относительные пути
7038
7039    async init() {
7040      try {
7041        const r = await fetch('/api/settings');
7042        if (r.ok) this.config = await r.json();
7043      } catch (e) {
7044        console.warn('Settings load failed:', e);
7045      }
7046      try {
7047        const t = await this.fetchFormulaTemplates();
7048        this.templates = t.templates || [];
7049      } catch (e) {
7050        this.templates = [];
7051      }
7052    },
7053
7054    getTemplatesMap() {
```

```
7055      const map = {};
7056      (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
7057      return map;
7058    },
7059
7060    // ← ОДНА функция fetchSignals с cache-busting
7061    async fetchSignals(mask, limit = 50) {
7062      const timestamp = Date.now();
7063      const url = `${this.apiUrl}/api/signals?q=${encodeURIComponent(mask || '')}
      &limit=${limit}&_t=${timestamp}`;
7064      const r = await fetch(url);
7065      if (!r.ok) throw new Error('Failed to fetch signals');
7066      return await r.json();
7067    },
7068
7069    async saveProject(filename, projectData) {
7070      if (!filename.endsWith('.json')) {
7071        filename += '.json';
7072      }
7073      const r = await fetch(`${this.apiUrl}/api/project/save`, {
7074        method: 'POST',
7075        headers: { 'Content-Type': 'application/json' },
7076        body: JSON.stringify({
7077          filename: filename,
7078          content: projectData
7079        })
7080      });
7081      if (!r.ok) throw new Error('Failed to save project');
7082      return r.json();
7083    },
7084
7085    async listProjects() {
7086      const r = await fetch(`${this.apiUrl}/api/project/list`);
7087      if (!r.ok) throw new Error('Failed to list projects');
7088      return r.json();
7089    },
7090
7091    async fetchFormulaTemplates() {
7092      const r = await fetch(`${this.apiUrl}/api/formula-templates`);
7093      if (!r.ok) throw new Error('Failed to fetch formula templates');
7094      return await r.json();
7095    },
7096
7097    async loadProject(filename) {
7098      if (!filename.endsWith('.json')) {
7099        filename += '.json';
7100      }
7101      const r = await fetch(`${this.apiUrl}/api/project/load/$
      {encodeURIComponent(filename)}`);
7102      if (!r.ok) {
7103        if (r.status === 404) {
7104          throw new Error(`Project "${filename}" not found (404)`);
7105        }
7106        throw new Error('Failed to load project');
7107      }
7108      return r.json();
7109    }
7110  };
7111
7112   /**
7113   * Глобальное состояние приложения
7114   * state.js
7115   */
7116
7117  const AppState = {
```

```
    // Элементы схемы
    elements: {},
    connections: [],
    elementCounter: 0,

    // Выделение
    selectedElement: null,
    selectedElements: [],

    // Перетаскивание
    draggingElement: null,
    dragOffset: { x: 0, y: 0 },
    isDraggingFromPalette: false,
    dragPreview: null,
    dragType: null,

    // Соединения
    connectingFrom: null,
    connectingFromType: null,
    tempLine: null,

    // Resize
    resizing: null,

    // Viewport (масштабирование и перемещение)
    viewport: {
        zoom: 1,
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    },

    // Свойства проекта
    project: {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        description: '',
        dimension: '',
        possibleCause: '',
        guidelines: '',
        visualizer_state: null  // НОВОЕ: состояние визуализатора
    },

    // Выходные сигналы (автоматически определяются)
    outputs: {
        logical: [],
        numeric: []
    },

    // НОВОЕ: токен текущей сессии визуализатора
    currentVisualizerToken: null
};

/**
 * Сброс состояния
 */
function resetState() {
    AppState.elements = {};
    AppState.connections = [];
    AppState.elementCounter = 0;
    AppState.selectedElement = null;
    AppState.selectedElements = [];
    AppState.draggingElement = null;
```

```
    AppState.connectingFrom = null;
    AppState.tempLine = null;
    AppState.resizing = null;

    AppState.viewport = {
        zoom: 1,
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    };

    AppState.project = {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        description: '',
        dimension: '',
        possibleCause: '',
        guidelines: '',
        visualizer_state: null  // НОВОЕ: сбрасываем состояние визуализатора
    };

    AppState.outputs = {
        logical: [],
        numeric: []
    };

    // НОВОЕ: сбрасываем токен визуализатора
    AppState.currentVisualizerToken = null;
}

/**
 * Вспомогательные функции
 * utils.js
 */

/**
 * Генерация уникального ID
 */
function generateId() {
    AppState.elementCounter++;
    return `elem_${AppState.elementCounter}`;
}

function getInputPortType(elementId, portIdentifier) {
    const element = AppState.elements[elementId];
    if (!element) return SIGNAL_TYPE.ANY;

    const config = ELEMENT_TYPES[element.type];
    if (!config) return SIGNAL_TYPE.ANY;

    let portIndex = portIdentifier;

    // Обработка технического порта условия
    if (typeof portIdentifier === 'string') {
        if (portIdentifier === 'cond-0' && config.hasConditionPort) {
            return config.conditionPortType || SIGNAL_TYPE.LOGIC;
        }

        if (portIdentifier.startsWith('in-')) {
            portIndex = parseInt(portIdentifier.split('-')[1], 10);
        }
    }

```

```
7248      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
7249          portIndex = 0;
7250      }
7251
7252      // Динамические входы для AND/OR берут тип из конфига
7253      if ((element.type === 'and' || element.type === 'or')) {
7254          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
7255      }
7256
7257      if (element.type === 'formula') {
7258          return SIGNAL_TYPE.ANY;
7259      }
7260
7261      const types = config.inputTypes || [];
7262      if (types.length === 0) return SIGNAL_TYPE.ANY;
7263
7264      if (portIndex < types.length) {
7265          return types[portIndex] || SIGNAL_TYPE.ANY;
7266      }
7267
7268      return types[types.length - 1] || SIGNAL_TYPE.ANY;
7269  }
7270
7271  function getOutputPortType(elementId, portIdentifier) {
7272      const element = AppState.elements[elementId];
7273      if (!element) return SIGNAL_TYPE.ANY;
7274
7275      const config = ELEMENT_TYPES[element.type];
7276      if (!config) return SIGNAL_TYPE.ANY;
7277
7278      let portIndex = portIdentifier;
7279
7280      if (typeof portIdentifier === 'string') {
7281          if (portIdentifier.startsWith('out-')) {
7282              portIndex = parseInt(portIdentifier.split('-')[1], 10);
7283          }
7284      }
7285
7286      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
7287          portIndex = 0;
7288      }
7289
7290      const types = config.outputTypes || [];
7291      if (types.length === 0) return SIGNAL_TYPE.ANY;
7292
7293      if (portIndex < types.length) {
7294          return types[portIndex] || SIGNAL_TYPE.ANY;
7295      }
7296
7297      return types[types.length - 1] || SIGNAL_TYPE.ANY;
7298  }
7299  /**
7300   * Проверка совместимости типов сигналов
7301   *
7302   * Новая логика:
7303   * - ANY совместим со всем
7304   * - TRUE совместим с LOGIC, TRUE, ANY
7305   * - FALSE совместим с LOGIC, FALSE, ANY
7306   * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
7307   * - NUMERIC совместим с NUMERIC, ANY
7308   */
7309  function areTypesCompatible(outputType, inputType) {
7310      // Если один из типов ANY - совместимы
7311      if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
```

```
7312          return true;
7313      }
7314
7315      // Если типы одинаковые - совместимы
7316      if (outputType === inputType) {
7317          return true;
7318      }
7319
7320      // TRUE/FALSE совместимы с LOGIC
7321      if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
7322          inputType === SIGNAL_TYPE.LOGIC) {
7323          return true;
7324      }
7325
7326      // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
7327      if (outputType === SIGNAL_TYPE.LOGIC &&
7328          (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
7329          return true;
7330      }
7331
7332      return false;
7333  }
7334
7335  /**
7336   * Проверка, находится ли элемент внутри рамки
7337   */
7338  function isInsideFrame(elemId, frameId) {
7339      const elem = AppState.elements[elemId];
7340      const frame = AppState.elements[frameId];
7341
7342      if (!elem || !frame || frame.type !== 'output-frame') return false;
7343
7344      const elemCenterX = elem.x + elem.width / 2;
7345      const elemCenterY = elem.y + elem.height / 2;
7346
7347      return elemCenterX > frame.x &&
7348             elemCenterX < frame.x + frame.width &&
7349             elemCenterY > frame.y &&
7350             elemCenterY < frame.y + frame.height;
7351  }
7352
7353  /**
7354   * Обновить принадлежность элементов к рамкам
7355   */
7356  function updateFrameChildren() {
7357      // Сначала очистим children у рамок и parentFrame у всех элементов
7358      Object.values(AppState.elements).forEach(elem => {
7359          if (elem.type === 'output-frame') {
7360              elem.children = [];
7361          } else {
7362              // удаляем parentFrame по умолчанию (пересчитаем ниже)
7363              if (elem.parentFrame) delete elem.parentFrame;
7364          }
7365      });
7366
7367      // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
     попадает
7368      Object.values(AppState.elements).forEach(elem => {
7369          if (!elem || elem.type === 'output-frame') return;
7370
7371          Object.values(AppState.elements).forEach(frame => {
7372              if (!frame || frame.type !== 'output-frame') return;
7373
7374              if (isInsideFrame(elem.id, frame.id)) {
7375                  // 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 };
}

/**
 * Проверка, является ли порт выходным (не подключен к другим элементам)
 */
function isOutputPort(elemId, portIndex) {
    const portKey = `out-${portIndex}`;

    // Проверяем, есть ли соединения от этого порта
    const hasConnection = AppState.connections.some(conn =>
        conn.fromElement === elemId && conn.fromPort === portKey
    );

    return !hasConnection;
}

/**
 * Получить информацию о выходном порте
 */
function getOutputPortInfo(elemId, portIndex) {
    const elem = AppState.elements[elemId];
    if (!elem) return null;

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

    return {
        elementId: elemId,
        elementType: elem.type,
        elementName: config.name,
        portIndex: portIndex,
```

```
7441          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
7442          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
7443          // Дополнительная информация для идентификации
7444          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
7445      };
7446  }
7447
7448  function splitArgsTopLevel(argStr) {
7449    const out = [];
7450    let cur = '';
7451    let depth = 0;
7452    for (let i = 0; i < argStr.length; i++) {
7453      const ch = argStr[i];
7454      if (ch === '(') depth++;
7455      if (ch === ')') depth--;
7456      if (ch === ',' && depth === 0) {
7457        out.push(cur.trim());
7458        cur = '';
7459      } else {
7460        cur += ch;
7461      }
7462    }
7463    if (cur.trim()) out.push(cur.trim());
7464    return out;
7465  }
7466
7467  // js/codegen.js
7468
7469  function expandFormulaTemplates(expr, templatesMap) {
7470      if (!expr) return expr;
7471      if (!templatesMap) return expr;
7472
7473      // несколько проходов на случай вложенных шаблонов
7474      for (let pass = 0; pass < 10; pass++) {
7475          let changed = false;
7476
7477          // Регулярка ищет вызовы функций: funcName(arg1, arg2, ...)
7478          expr = expr.replace(/([A-Za-z_]\w*)\s*\(([^()]|\([^()]*\))*\)/g, (match, name)
      => {
7479              const tpl = templatesMap[name];
7480              if (!tpl) return match;
7481
7482              // 1. Извлекаем аргументы из вызова: h(10, 20, 30) -> ["10", "20", "30"]
7483              const open = match.indexOf('(');
7484              const close = match.lastIndexOf(')');
7485              const inside = match.slice(open + 1, close);
7486              const callArgs = splitArgsTopLevel(inside);
7487
7488              // 2. Определяем формальные параметры (ключи) и конфиг
7489              let formalArgs = [];
7490              let argsConfig = {};
7491
7492              if (Array.isArray(tpl.args)) {
7493                  // Старый формат: "args": ["p", "t"]
7494                  formalArgs = tpl.args;
7495              } else if (typeof tpl.args === 'object' && tpl.args !== null) {
7496                  // Новый формат: "args": { "p": {"min":...}, ... }
7497                  formalArgs = Object.keys(tpl.args);
7498                  argsConfig = tpl.args;
7499              }
7500
7501              // Если количество аргументов не совпало — не трогаем (чтобы не сломать)
7502              if (callArgs.length !== formalArgs.length) return match;
7503
```

```
7504              // 3. Подготовка тела функции
7505              let body = String(tpl.body || '0');
7506
7507              // 4. Сбор условий валидации (min/max)
7508              let conditions = [];
7509
7510              formalArgs.forEach((fName, i) => {
7511                  const actualVal = callArgs[i]; // То, что передали: "10" или
      "sensor_1"
7512
7513                  // Подстановка значения в тело: заменяем параметр p на 10
7514                  const re = new RegExp(`\\b${fName}\\b`, 'g');
7515                  body = body.replace(re, `(${actualVal})`);
7516
7517                  // Проверка ограничений (только для нового формата)
7518                  if (argsConfig[fName]) {
7519                      const conf = argsConfig[fName];
7520
7521                      // Проверка min
7522                      if (conf.min !== undefined && conf.min !== null) {
7523                          conditions.push(`(${actualVal} >= ${conf.min})`);
7524                      }
7525                      // Проверка max
7526                      if (conf.max !== undefined && conf.max !== null) {
7527                          conditions.push(`(${actualVal} <= ${conf.max})`);
7528                      }
7529                  }
7530              });
7531
7532              // 5. Формирование результата
7533              let resultExpr = `(${body})`;
7534
7535              // Если есть условия, заворачиваем в WHEN
7536              if (conditions.length > 0) {
7537                  const conditionString = conditions.join(' AND ');
7538                  const fallbackValue = tpl.return_value !== undefined ?
      tpl.return_value : 0;
7539
7540                  // WHEN(условия, формула, значение_по_умолчанию)
7541                  resultExpr = `WHEN(${conditionString}, ${body}, ${fallbackValue})`;
7542              }
7543
7544              changed = true;
7545              return resultExpr;
7546          });
7547
7548          if (!changed) break;
7549      }
7550
7551      return expr;
7552  }
7553
7554  /**
7555   * Модуль управления viewport (масштабирование и перемещение)
7556   * viewport.js
7557   */
7558
7559  const Viewport = {
7560      /**
7561       * Инициализация viewport
7562       */
7563      init() {
7564          this.setupZoomControls();
7565          this.setupPanning();
7566          this.setupMouseWheel();
```

```
7567          this.setupMinimap();
7568          this.setupCursorPosition();
7569          this.updateTransform();
7570          const container = document.getElementById('workspace-container');
7571          const rect = container.getBoundingClientRect();
7572          AppState.viewport.panX = 100; // немного отступить от левого края
7573          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
      AppState.viewport.zoom;
7574          this.updateTransform();
7575      },
7576
7577      /**
7578       * Настройка кнопок масштабирования
7579       */
7580      setupZoomControls() {
7581          document.getElementById('btn-zoom-in').addEventListener('click', () => {
7582              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
7583          });
7584
7585          document.getElementById('btn-zoom-out').addEventListener('click', () => {
7586              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
7587          });
7588
7589          document.getElementById('btn-zoom-reset').addEventListener('click', () => {
7590              this.setZoom(1);
7591              this.setPan(0, 0);
7592          });
7593
7594          document.getElementById('btn-zoom-fit').addEventListener('click', () => {
7595              this.fitToContent();
7596          });
7597      },
7598
7599      /**
7600       * Настройка перемещения (pan)
7601       */
7602      setupPanning() {
7603          const container = document.getElementById('workspace-container');
7604
7605          container.addEventListener('mousedown', (e) => {
7606              // Средняя кнопка мыши или пробел + левая кнопка
7607              if (e.button === 1 || (e.button === 0 && e.target === container)) {
7608                  e.preventDefault();
7609                  AppState.viewport.isPanning = true;
7610                  AppState.viewport.lastMouseX = e.clientX;
7611                  AppState.viewport.lastMouseY = e.clientY;
7612                  container.style.cursor = 'grabbing';
7613              }
7614          });
7615
7616          document.addEventListener('mousemove', (e) => {
7617              if (AppState.viewport.isPanning) {
7618                  const dx = e.clientX - AppState.viewport.lastMouseX;
7619                  const dy = e.clientY - AppState.viewport.lastMouseY;
7620
7621                  this.setPan(
7622                      AppState.viewport.panX + dx,
7623                      AppState.viewport.panY + dy
7624                  );
7625
7626                  AppState.viewport.lastMouseX = e.clientX;
7627                  AppState.viewport.lastMouseY = e.clientY;
7628              }
7629          });
7630
```

```
7631          document.addEventListener('mouseup', (e) => {
7632              if (AppState.viewport.isPanning) {
7633                  AppState.viewport.isPanning = false;
7634                  document.getElementById('workspace-container').style.cursor = '';
7635              }
7636          });
7637
7638          // Клавиша пробел для режима перемещения
7639          document.addEventListener('keydown', (e) => {
7640              if (e.code === 'Space' && !e.repeat) {
7641                  document.getElementById('workspace-container').style.cursor = 'grab';
7642              }
7643          });
7644
7645          document.addEventListener('keyup', (e) => {
7646              if (e.code === 'Space') {
7647                  document.getElementById('workspace-container').style.cursor = '';
7648              }
7649          });
7650      },
7651
7652      /**
7653       * Настройка масштабирования колесом мыши
7654       */
7655      setupMouseWheel() {
7656          const container = document.getElementById('workspace-container');
7657
7658          container.addEventListener('wheel', (e) => {
7659              e.preventDefault();
7660
7661              const rect = container.getBoundingClientRect();
7662              const mouseX = e.clientX - rect.left;
7663              const mouseY = e.clientY - rect.top;
7664
7665              // Позиция мыши на холсте до масштабирования
7666              const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
      AppState.viewport.zoom;
7667              const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
      AppState.viewport.zoom;
7668
7669              // Новый масштаб
7670              const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
      VIEWPORT_CONFIG.zoomStep;
7671              const newZoom = Math.max(
7672                  VIEWPORT_CONFIG.minZoom,
7673                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
7674              );
7675
7676              // Корректируем pan, чтобы точка под курсором осталась на месте
7677              const newPanX = mouseX - canvasPosBeforeX * newZoom;
7678              const newPanY = mouseY - canvasPosBeforeY * newZoom;
7679
7680              AppState.viewport.zoom = newZoom;
7681              AppState.viewport.panX = newPanX;
7682              AppState.viewport.panY = newPanY;
7683
7684              this.updateTransform();
7685          }, { passive: false });
7686      },
7687
7688      /**
7689       * Установить масштаб
7690       */
7691      setZoom(zoom) {
7692          const container = document.getElementById('workspace-container');
```

```
7693          const rect = container.getBoundingClientRect();
7694
7695          // Центр экрана
7696          const centerX = rect.width / 2;
7697          const centerY = rect.height / 2;
7698
7699          // Позиция центра на холсте
7700          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
7701          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
7702
7703          // Новый масштаб
7704          const newZoom = Math.max(
7705              VIEWPORT_CONFIG.minZoom,
7706              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
7707          );
7708
7709          // Корректируем pan
7710          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
7711          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
7712          AppState.viewport.zoom = newZoom;
7713
7714          this.updateTransform();
7715      },
7716
7717      /**
7718       * Установить смещение
7719       */
7720      setPan(x, y) {
7721          AppState.viewport.panX = x;
7722          AppState.viewport.panY = y;
7723          this.updateTransform();
7724      },
7725
7726      /**
7727       * Вписать содержимое в экран
7728       */
7729      fitToContent() {
7730          const elements = Object.values(AppState.elements);
7731          if (elements.length === 0) {
7732              this.setZoom(1);
7733              this.setPan(0, 0);
7734              return;
7735          }
7736
7737          // Находим границы содержимого
7738          let minX = Infinity, minY = Infinity;
7739          let maxX = -Infinity, maxY = -Infinity;
7740
7741          elements.forEach(elem => {
7742              minX = Math.min(minX, elem.x);
7743              minY = Math.min(minY, elem.y);
7744              maxX = Math.max(maxX, elem.x + elem.width);
7745              maxY = Math.max(maxY, elem.y + elem.height);
7746          });
7747
7748          const contentWidth = maxX - minX;
7749          const contentHeight = maxY - minY;
7750
7751          const container = document.getElementById('workspace-container');
7752          const rect = container.getBoundingClientRect();
7753
7754          const padding = 50;
7755          const availableWidth = rect.width - padding * 2;
```

```
7756          const availableHeight = rect.height - padding * 2;
7757
7758          const zoomX = availableWidth / contentWidth;
7759          const zoomY = availableHeight / contentHeight;
7760          const newZoom = Math.min(zoomX, zoomY, 1);
7761
7762          AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
7763          AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
      (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
7764          AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
      (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
7765
7766          this.updateTransform();
7767      },
7768
7769      /**
7770       * Обновить трансформацию
7771       */
7772      updateTransform() {
7773          const workspace = document.getElementById('workspace');
7774          const svg = document.getElementById('connections-svg');
7775
7776          const transform = `translate(${AppState.viewport.panX}px, $
      {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
7777
7778          workspace.style.transform = transform;
7779          svg.style.transform = transform;
7780
7781          // Обновляем отображение масштаба
7782          document.getElementById('zoom-level').textContent = `$
      {Math.round(AppState.viewport.zoom * 100)}%`;
7783
7784          // Обновляем мини-карту
7785          this.updateMinimap();
7786      },
7787
7788      /**
7789       * Настройка мини-карты
7790       */
7791      setupMinimap() {
7792          const minimap = document.getElementById('minimap');
7793          const canvas = document.getElementById('minimap-canvas');
7794
7795          canvas.width = MINIMAP_CONFIG.width;
7796          canvas.height = MINIMAP_CONFIG.height;
7797
7798          // Клик по мини-карте для перемещения
7799          minimap.addEventListener('click', (e) => {
7800              const rect = minimap.getBoundingClientRect();
7801              const x = e.clientX - rect.left;
7802              const y = e.clientY - rect.top;
7803
7804              this.navigateToMinimapPosition(x, y);
7805          });
7806      },
7807
7808      /**
7809       * Обновить мини-карту
7810       */
7811      updateMinimap() {
7812          const canvas = document.getElementById('minimap-canvas');
7813          const ctx = canvas.getContext('2d');
7814          const viewportEl = document.getElementById('minimap-viewport');
7815
7816          // Очищаем
```

```
7817            ctx.fillStyle = '#0a0a1a';
7818            ctx.fillRect(0, 0, canvas.width, canvas.height);
7819
7820            // Масштаб мини-карты
7821            const scale = Math.min(
7822                canvas.width / VIEWPORT_CONFIG.canvasWidth,
7823                canvas.height / VIEWPORT_CONFIG.canvasHeight
7824            );
7825
7826            // Рисуем элементы
7827            Object.values(AppState.elements).forEach(elem => {
7828                const x = elem.x * scale;
7829                const y = elem.y * scale;
7830                const w = Math.max(elem.width * scale, 2);
7831                const h = Math.max(elem.height * scale, 2);
7832
7833                ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
7834                ctx.fillRect(x, y, w, h);
7835            });
7836
7837            // Рисуем viewport
7838            const container = document.getElementById('workspace-container');
7839            const rect = container.getBoundingClientRect();
7840
7841            const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
7842            const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
7843            const vpW = (rect.width / AppState.viewport.zoom) * scale;
7844            const vpH = (rect.height / AppState.viewport.zoom) * scale;
7845
7846            viewportEl.style.left = `${vpX}px`;
7847            viewportEl.style.top = `${vpY}px`;
7848            viewportEl.style.width = `${vpW}px`;
7849            viewportEl.style.height = `${vpH}px`;
7850        },
7851
7852        /**
7853         * Перейти к позиции на мини-карте
7854         */
7855        navigateToMinimapPosition(minimapX, minimapY) {
7856            const canvas = document.getElementById('minimap-canvas');
7857            const container = document.getElementById('workspace-container');
7858            const rect = container.getBoundingClientRect();
7859
7860            const scale = Math.min(
7861                canvas.width / VIEWPORT_CONFIG.canvasWidth,
7862                canvas.height / VIEWPORT_CONFIG.canvasHeight
7863            );
7864
7865            const canvasX = minimapX / scale;
7866            const canvasY = minimapY / scale;
7867
7868            // Центрируем viewport на этой точке
7869            AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
7870            AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
7871
7872            this.updateTransform();
7873        },
7874
7875        /**
7876         * Отслеживание позиции курсора
7877         */
7878        setupCursorPosition() {
7879            const container = document.getElementById('workspace-container');
7880
7881            container.addEventListener('mousemove', (e) => {
```

```
            const pos = screenToCanvas(e.clientX, e.clientY);
            document.getElementById('cursor-pos').textContent =
                `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
        });
    }
};

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;
```

```
7947      gap: 8px;
7948      background: #0a0a1a;
7949      padding: 5px 10px;
7950      border-radius: 5px;
7951  }
7952
7953  .zoom-btn {
7954      width: 30px;
7955      height: 30px;
7956      padding: 0;
7957      font-size: 18px;
7958      font-weight: bold;
7959  }
7960
7961  #zoom-level {
7962      min-width: 50px;
7963      text-align: center;
7964      font-size: 12px;
7965      color: #aaa;
7966  }
7967
7968  /* ====== ОСНОВНАЯ ОБЛАСТЬ ====== */
7969  #main {
7970      display: flex;
7971      flex: 1;
7972      overflow: hidden;
7973  }
7974
7975  /* ====== ПАЛИТРА ====== */
7976  #palette {
7977      width: 200px;
7978      background: #16213e;
7979      padding: 15px;
7980      border-right: 2px solid #0f3460;
7981      overflow-y: auto;
7982      z-index: 10;
7983      flex-shrink: 0;
7984  }
7985
7986  #palette h3 {
7987      margin-bottom: 15px;
7988      color: #e94560;
7989      text-align: center;
7990      font-size: 14px;
7991  }
7992
7993  .palette-section {
7994      margin-bottom: 15px;
7995  }
7996
7997  .palette-section-title {
7998      font-size: 11px;
7999      color: #888;
8000      margin-bottom: 8px;
8001      padding-bottom: 3px;
8002      border-bottom: 1px solid #333;
8003  }
8004
8005  .palette-item {
8006      background: #0f3460;
8007      padding: 8px;
8008      margin-bottom: 6px;
8009      border-radius: 8px;
8010      cursor: grab;
8011      text-align: center;
```

```
    transition: all 0.3s;
    border: 2px solid transparent;
    user-select: none;
}

.palette-item:hover {
    border-color: #e94560;
    transform: scale(1.02);
}

.palette-item:active {
    cursor: grabbing;
}

.palette-item svg {
    width: 50px;
    height: 32px;
    margin-bottom: 2px;
    pointer-events: none;
}

.palette-item-name {
    font-size: 10px;
    color: #aaa;
    pointer-events: none;
}

.type-legend {
    margin-top: 15px;
    padding-top: 10px;
    border-top: 1px solid #333;
    font-size: 10px;
}

.type-legend-item {
    display: flex;
    align-items: center;
    gap: 8px;
    margin-bottom: 5px;
}

.type-legend-dot {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    border: 2px solid #fff;
}
.type-legend-dot.logic { background: #a855f7; }
.type-legend-dot.number { background: #3b82f6; }

/* ====== РАБОЧАЯ ОБЛАСТЬ ====== */
#workspace-container {
    flex: 1;
    position: relative;
    overflow: hidden;
    background-color: #0a0a1a;
    background-image:
        linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
        linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);
    background-size: 25px 25px;
}

#workspace {
    position: absolute;
    transform-origin: 0 0;
```

```
8077      width: 5000px;
8078      height: 5000px;
8079  }
8080
8081  #connections-svg {
8082      position: absolute;
8083      transform-origin: 0 0;
8084      pointer-events: none;
8085      z-index: 5;
8086      width: 5000px;
8087      height: 5000px;
8088  }
8089
8090  #connections-svg path {
8091      pointer-events: stroke;
8092  }
8093
8094  /* ====== ЭЛЕМЕНТЫ ====== */
8095  .element {
8096      position: absolute;
8097      background: #0f3460;
8098      border: 2px solid #4a90d9;
8099      border-radius: 8px;
8100      cursor: move;
8101      user-select: none;
8102      z-index: 10;
8103      display: flex;
8104      flex-direction: column;
8105  }
8106
8107  .element.selected {
8108      border-color: #e94560;
8109      box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
8110  }
8111
8112  .element-header {
8113      background: #4a90d9;
8114      padding: 5px 10px;
8115      border-radius: 5px 5px 0 0;
8116      font-size: 11px;
8117      font-weight: bold;
8118      text-align: center;
8119      white-space: nowrap;
8120      overflow: hidden;
8121      text-overflow: ellipsis;
8122  }
8123
8124  .element-body {
8125      padding: 10px;
8126      display: flex;
8127      justify-content: space-between;
8128      align-items: center;
8129      flex: 1;
8130      gap: 8px;
8131  }
8132
8133  .element-symbol {
8134      font-size: 16px;
8135      font-weight: bold;
8136      flex: 1;
8137      text-align: center;
8138      padding: 0 5px;
8139      word-break: break-all;
8140      color: #eee;
8141  }
```

```
8142
8143  /* ====== ПОРТЫ ====== */
8144  .ports-left, .ports-right {
8145      display: flex;
8146      flex-direction: column;
8147      justify-content: space-around;
8148      gap: 10px;
8149      height: 100%;
8150  }
8151
8152  .port {
8153      width: 14px;
8154      height: 14px;
8155      border-radius: 50%;
8156      border: 2px solid #fff;
8157      cursor: crosshair;
8158      transition: all 0.2s;
8159      position: relative;
8160      flex-shrink: 0;
8161  }
8162
8163  .port:hover { transform: scale(1.3); }
8164  .port.input { margin-left: -8px; }
8165  .port.output { margin-right: -8px; }
8166  .port.connected { background: #4ade80; }
8167
8168  /* Типы портов */
8169  .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
8170  .port.logic-port:hover { background: #c084fc; }
8171  .port.logic-port.connected { background: #7c3aed; }
8172
8173  .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
8174  .port.number-port:hover { background: #60a5fa; }
8175  .port.number-port.connected { background: #2563eb; }
8176
8177  .port.any-port { background: #6b7280; border-color: #d1d5db; }
8178  .port.any-port:hover { background: #9ca3af; }
8179  .port.any-port.connected { background: #4b5563; }
8180
8181  .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !
      important; }
8182  .port.output.no-port { background: #f87171 !important; border-color: #fecaca !
      important; }
8183
8184  .port.incompatible { opacity: 0.3; cursor: not-allowed; }
8185  .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
8186
8187  /* ====== RESIZE HANDLES ====== */
8188  .resize-handle {
8189      position: absolute;
8190      width: 12px;
8191      height: 12px;
8192      background: #e94560;
8193      border: 1px solid #fff;
8194      border-radius: 3px;
8195      z-index: 20;
8196      opacity: 0;
8197      transition: opacity 0.2s;
8198  }
8199  .element.selected .resize-handle { opacity: 0.8; }
8200  .resize-handle:hover { opacity: 1; }
8201  .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
8202  .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor:
      ew-resize; }
8203  .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%);
```

```
      cursor: ns-resize; }
8204
8205
8206  /* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
8207  .element.input-signal {
8208      background: transparent;
8209      border: none;
8210  }
8211
8212  .element.input-signal .element-header {
8213      display: none; /* У трапеции нет заголовка */
8214  }
8215
8216  .element.input-signal .element-body {
8217      padding: 0;
8218      background: #0f3460;
8219      border: 2px solid #4a90d9;
8220      clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
8221      display: flex;
8222      justify-content: space-between;
8223      align-items: center;
8224      padding-left: 15px;
8225      padding-right: 25px;
8226  }
8227
8228  .element.input-signal .element-symbol {
8229      text-align: left;
8230      color: #eee;
8231  }
8232
8233  .element.input-signal.selected .element-body {
8234      border-color: #e94560;
8235  }
8236
8237  /* ====== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ====== */
8238  .element.output {
8239      background: rgba(16, 185, 129, 0.1);
8240      border: 2px dashed #10b981;
8241  }
8242
8243  .element.output .element-header {
8244      display: none; /* У выхода нет заголовка */
8245  }
8246
8247  .element.output .element-body {
8248      padding-left: 20px;
8249  }
8250
8251  .element.output .element-symbol {
8252      color: #10b981;
8253      font-size: 14px;
8254  }
8255
8256  .element.output.selected {
8257      border-color: #e94560;
8258      border-style: dashed;
8259  }
8260
8261
8262  /* Formula condition port */
8263  /* Универсальный стиль для технического порта (сверху) */
8264  .element.has-condition-port {
8265      margin-top: 30px; /* Даем место порту над элементом */
8266  }
8267
```

```
.condition-port-wrapper {
    position: absolute;
    top: -28px;
    left: 50%;
    transform: translateX(-50%);
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 4px;
    pointer-events: none;
    z-index: 21;
}

.condition-port-label {
    font-size: 10px;
    color: #f59e0b;
    font-weight: 600;
    white-space: nowrap;
}

.port.condition-port {
    pointer-events: auto;
    width: 16px;
    height: 16px;
    border-radius: 50%;
    border: 2px solid #f59e0b;
    background: #fff7ed;
    margin: 0; /* Сбрасываем лишние отступы */
}
.element.formula .condition-port:hover { background: #fde68a; }


/* ====== СОЕДИНЕНИЯ ====== */
.connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #4a90d9;
    stroke-width: 2.5;
}
.connection:hover {
    stroke: #e94560;
    stroke-width: 4;
}

.connection.logic-conn { stroke: #a855f7; }
.connection.numeric-conn { stroke: #3b82f6; }
.connection.any-conn { stroke: #6b7280; }
.connection.true-conn { stroke: #4ade80; }
.connection.false-conn { stroke: #f87171; }

.connection.yes-conn { stroke: #4ade80; }
.connection.no-conn { stroke: #f87171; }

.temp-connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #e94560;
    stroke-width: 2;
    stroke-dasharray: 5, 5;
}
.temp-connection.invalid { stroke: #ef4444; }

/* ====== ПРОЧЕЕ ====== */
.drag-preview {
    position: fixed;
    pointer-events: none;
    opacity: 0.8;
```

```
    z-index: 1000;
    background: #0f3460;
    border: 2px solid #e94560;
    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; }
.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; }
.output-counter:empty, .output-counter[style*="display: none"] { display: none; }
.element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
.element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important; border-color: #fbbf24 !important; }
.port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation: pulse-output 1.5s infinite; }
@keyframes pulse-output {
    0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
    50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
}

.outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-height: 250px; overflow-y: auto; }
.outputs-section { margin-bottom: 15px; }
.outputs-section:last-child { margin-bottom: 0; }
.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; }
.outputs-section-title .section-icon { font-size: 16px; }
.outputs-list { display: flex; flex-direction: column; gap: 5px; }
.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; }
.output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981; transform: translateX(5px); }
.output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59, 130, 246, 0.3); }
.output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color: #3b82f6; }
.output-icon { font-size: 14px; }
.output-name { font-weight: bold; color: #eee; }
.output-port { color: #888; font-size: 12px; margin-left: auto; }
.no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
.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; }
.element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-color: rgba(240,80,80,0.8) !important; }
.element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-color: rgba(250,200,30,0.8) !important; }
/* TRUE/FALSE порты (для сепаратора) */
.port.true-port {
    background: #4ade80 !important;
    border-color: #bbf7d0 !important;
}
.port.true-port:hover {
    background: #22c55e !important;
}
.port.true-port.connected {
    background: #16a34a !important;
}

.port.false-port {
    background: #f87171 !important;
    border-color: #fecaca !important;
}
.port.false-port:hover {
    background: #ef4444 !important;
}
```

```
8504  .port.false-port.connected {
8505      background: #dc2626 !important;
8506  }
8507
8508  /* Сепаратор стиль */
8509  .element.separator {
8510      background: #0f3460;
8511      border: 2px solid #f59e0b;
8512  }
8513
8514  .element.separator.selected {
8515      border-color: #e94560;
8516      box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
8517  }
8518
8519  /* === Выделение рамкой === */
8520  #selection-rect {
8521    position: absolute;
8522    border: 1px dashed #e94560;
8523    background: rgba(233, 69, 96, 0.1);
8524    pointer-events: none;
8525    display: none;
8526    z-index: 200;
8527  }
8528
8529  /* === Кастомный элемент “Группа” === */
8530  .element.group {
8531    background: rgba(107, 114, 128, 0.12);
8532    border: 2px dashed #6b7280;
8533    border-radius: 8px;
8534    position: absolute;
8535    z-index: 1;              /* ниже обычных элементов (у них z-index: 10) */
8536  }
8537
8538  .element.group .group-title {
8539    pointer-events: auto;
8540  }
8541
8542  .group-title {
8543    position: absolute;
8544    top: -20px;
8545    left: 5px;
8546    font-size: 11px;
8547    color: #ccc;
8548    background: #16213e;
8549    padding: 2px 6px;
8550    border-radius: 4px;
8551    pointer-events: auto; /* можно кликнуть для выбора */
8552  }
8553
8554  .modal.hidden { display: none; }
8555  .modal { position: fixed; inset: 0; display: flex; align-items: center; justify-
      content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
8556  .modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px;
      max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow:
      hidden; }
8557  .modal__content--wide { width: 800px; }
8558  .modal__title { margin: 0; }
8559
8560  .project-list__toolbar { display: flex; gap: 12px; }
8561  .project-list__toolbar input { flex: 1; padding: 6px 10px; }
8562  .project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd;
      border-radius: 6px; }
8563  .project-list__table { width: 100%; border-collapse: collapse; }
8564  .project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom:
```

```
      1px solid #eee; }
8565  .project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
8566  .project-list__table tbody tr:hover { background: #f0f6ff; }
8567  .project-list__empty { text-align: center; color: #888; padding: 16px; }
8568  .modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
8569  .project-list__table th,
8570  .project-list__table td {
8571    color: #111;          /* насыщённый чёрный текст */
8572    padding: 8px 12px;
8573    border-bottom: 1px solid #eee;
8574  }
8575  .modal__content--wide {
8576    width: 860px;
8577    max-height: 90vh;    /* занимает 90% экрана */
8578  }
8579
8580  .project-list__table-container {
8581    flex: 1;
8582    overflow: auto;
8583    border: 1px solid #ddd;
8584    border-radius: 6px;
8585    max-height: 60vh;     /* много строк */
8586  }
8587
8588  .element-comment {
8589    padding: 6px 10px 10px;
8590    font-size: 11px;
8591    color: #cbd5e1;
8592    opacity: 0.9;
8593    border-top: 1px solid rgba(255,255,255,0.08);
8594    white-space: pre-wrap;
8595    word-break: break-word;
8596  }
8597
8598  .element-comment:empty { display: none; }
8599
8600  /* Tooltip для шаблонов формул */
8601  .template-item {
8602      position: relative;
8603  }
8604
8605  .template-tooltip {
8606      position: fixed;
8607      background: #1a1a2e;
8608      border: 1px solid #4a90d9;
8609      border-radius: 6px;
8610      padding: 8px 12px;
8611      color: #e0e0e0;
8612      font-size: 12px;
8613      max-width: 280px;
8614      line-height: 1.4;
8615      box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
8616      z-index: 10000;
8617      pointer-events: none;
8618      opacity: 0;
8619      transition: opacity 0.15s ease;
8620  }
8621
8622  .template-tooltip.visible {
8623      opacity: 1;
8624  }
8625
8626  .template-tooltip::before {
8627      content: '';
8628      position: absolute;
```

```
8629      top: -6px;
8630      left: 20px;
8631      border-left: 6px solid transparent;
8632      border-right: 6px solid transparent;
8633      border-bottom: 6px solid #4a90d9;
8634  }
8635
8636  .template-tooltip-title {
8637      font-weight: 600;
8638      color: #4a90d9;
8639      margin-bottom: 4px;
8640  }
```