```
#main.py
import os
import json
from typing import Dict, List
import pandas as pd
from fastapi import FastAPI, HTTPException, Request, Response
from fastapi.responses import JSONResponse
from fastapi.staticfiles import StaticFiles
import uuid
import pickle
from fastapi.middleware.cors import CORSMiddleware

import tempfile
from io import BytesIO

import pickle  # <-- отсутствовал







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")

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

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


def load_project_signals(folder: str) -> List[Dict]:
    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()   # ← КРИТИЧНО: берем code

            if not code:
                continue

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

            out.append({
                "Tagname": code,               # ← именно 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():
    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



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

    signals_map = {}  # Tagname -> dict
    for name in os.listdir(folder_abs):
        if not name.lower().endswith(".csv"):
            continue
        path = os.path.join(folder_abs, name)
        try:
            try:
                df = pd.read_csv(path, sep=';')[['Tagname', 'Description',
'Engineering Unit']]
            except KeyError:
                df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
            df = df.dropna(subset=['Tagname'])
            for _, row in df.iterrows():
                tag = str(row['Tagname']).strip()
                desc = "" if pd.isna(row['Description']) else
str(row['Description']).strip()
                try:
                    unit = "" if pd.isna(row['Engineering Unit']) else
str(row['Engineering Unit']).strip()
                except KeyError:
                    unit = ""
                desc_full = ", ".join([x for x in [desc, unit] if x])

                if tag:
                    signals_map[tag] = {
                        "Tagname": tag,
                        "Description": desc_full,
                        "EngineeringUnit": unit
```

```
126                     }
127         except Exception as e:
128             print(f"[WARN] failed to read {path}: {e}")
129 
130     out = list(signals_map.values())
131     out.sort(key=lambda x: x["Tagname"])
132     return out
133 
134 # main.py — добавь после существующих функций
135 
136 def extract_input_signals_from_project(project_data: Dict) -> List[str]:
137     """Извлекает имена входных сигналов из данных проекта"""
138     elements = project_data.get("elements", {})
139     input_signals = []
140 
141     for elem_id, elem_data in elements.items():
142         if elem_data.get("type") == "input-signal":
143             props = elem_data.get("props", {})
144             signal_name = props.get("name")
145             if signal_name:
146                 input_signals.append(signal_name)
147 
148     return input_signals
149 
150 
151 def load_project_by_code(code: str) -> Dict | None:
152     """Загружает проект по его коду (Tagname)"""
153     folder = STATE["settings"].get("projectDataFolder")
154     if not folder:
155         return None
156 
157     folder_abs = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
158     if not os.path.isdir(folder_abs):
159         return None
160 
161     for name in os.listdir(folder_abs):
162         if not name.endswith(".json"):
163             continue
164         path = os.path.join(folder_abs, name)
165         try:
166             with open(path, "r", encoding="utf-8") as f:
167                 payload = json.load(f)
168             proj = payload.get("project", {})
169             if proj.get("code") == code:
170                 return {
171                     "project": proj,
172                     "formula": payload.get("code", ""),  # сгенерированная формула
173                     "elements": payload.get("elements", {})
174                 }
175         except Exception as e:
176             print(f"[WARN] Error reading project {path}: {e}")
177             continue
178 
179     return None
180 
181 
182 def is_base_signal(signal_name: str) -> bool:
183     """Проверяет, есть ли сигнал в архиве (базовый сигнал с данными)"""
184     signal_index = STATE.get("signal_index", {})
185     return signal_name in signal_index
186 
187 
188 def resolve_signal_dependencies(
189     signal_names: List[str],
```

```
190      visited: set = None,
191      resolved: Dict[str, Dict] = None
192  ) -> tuple[set, Dict[str, Dict]]:
193      """
194      Рекурсивно разворачивает зависимости сигналов.
195
196      Returns:
197          base_signals: множество базовых сигналов (с данными в архиве)
198          synthetic_signals: словарь {code: {formula, dependencies}}
199      """
200      if visited is None:
201          visited = set()
202      if resolved is None:
203          resolved = {}
204
205      base_signals = set()
206
207      for signal_name in signal_names:
208          if not signal_name or signal_name in visited:
209              continue
210          visited.add(signal_name)
211
212          # Сначала проверяем, есть ли в архиве (базовый сигнал)
213          if is_base_signal(signal_name):
214              base_signals.add(signal_name)
215              continue
216
217          # Пробуем загрузить как проект (синтетический сигнал)
218          project = load_project_by_code(signal_name)
219          if project is None:
220              # Сигнал не найден ни в архиве, ни в проектах
221              # Добавляем в базовые — загрузчик потом вернёт "not found"
222              base_signals.add(signal_name)
223              print(f"[WARN] Signal '{signal_name}' not found in archive or projects")
224              continue
225
226          # Это синтетический сигнал!
227          formula = project.get("formula", "")
228          dependencies = extract_input_signals_from_project(project)
229
230          print(f"[INFO] Synthetic signal '{signal_name}' depends on: {dependencies}")
231
232          resolved[signal_name] = {
233              "formula": formula,
234              "dependencies": dependencies
235          }
236
237          # Рекурсивно обрабатываем зависимости
238          sub_base, _ = resolve_signal_dependencies(dependencies, visited, resolved)
239          base_signals.update(sub_base)
240
241      return base_signals, resolved
242
243
244  def topological_sort_signals(synthetic_signals: Dict[str, Dict]) -> List[str]:
245      """
246      Топологическая сортировка синтетических сигналов.
247      Возвращает порядок вычисления (сначала те, от которых зависят другие).
248      """
249      if not synthetic_signals:
250          return []
251
252      # Строим граф зависимостей (только между синтетическими сигналами)
253      in_degree = {name: 0 for name in synthetic_signals}
254      graph = {name: [] for name in synthetic_signals}
```

```
255
256     for name, data in synthetic_signals.items():
257         for dep in data.get("dependencies", []):
258             if dep in synthetic_signals:
259                 graph[dep].append(name)
260                 in_degree[name] += 1
261
262     # Алгоритм Кана для топологической сортировки
263     queue = [name for name, degree in in_degree.items() if degree == 0]
264     result = []
265
266     while queue:
267         node = queue.pop(0)
268         result.append(node)
269
270         for neighbor in graph[node]:
271             in_degree[neighbor] -= 1
272             if in_degree[neighbor] == 0:
273                 queue.append(neighbor)
274
275     # Проверка на циклы
276     if len(result) != len(synthetic_signals):
277         cyclic = [name for name in synthetic_signals if name not in result]
278         raise ValueError(f"Обнаружена циклическая зависимость между сигналами:
    {cyclic}")
279
280     return result
281
282
283
284 app = FastAPI()
285 app.add_middleware(
286     CORSMiddleware,
287     allow_origins=["http://localhost:8501"],
288     allow_credentials=True,
289     allow_methods=["*"],
290     allow_headers=["*"],
291 )
292
293 # Кэш сигналов в памяти
294 STATE = {
295     "settings": None,
296     "signals": None,
297     "signal_index": None
298 }
299
300
301 def build_signal_index(folder: str) -> Dict[str, List[str]]:
302     """
303     Проходим по всем CSV файлам и создаём индекс
304     signal_name -> list of files where it's present
305     """
306     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
307         os.path.join(BASE_DIR, folder)
308     )
309
310     if not os.path.isdir(folder_abs):
311         raise FileNotFoundError(f"Signal data folder not found: {folder_abs}")
312
313     signal_index = {}
314
315     print(f"[INFO] Building signal index from {folder_abs}...")
316
317     for filename in os.listdir(folder_abs):
318         if not filename.lower().endswith(".csv"):
```

```
319             continue
320
321         filepath = os.path.join(folder_abs, filename)
322
323         try:
324             df_header = pd.read_csv(
325                 filepath,
326                 nrows=0,
327                 encoding="ISO-8859-2",
328                 sep=";"
329             )
330
331             columns = df_header.columns.tolist()
332             signal_columns = [c for c in columns if c not in ["DATE", "TIME",
    "datetime"]]
333
334             for signal_name in signal_columns:
335                 if signal_name not in signal_index:
336                     signal_index[signal_name] = []
337                 signal_index[signal_name].append(filepath)
338
339             print(f"  ✓ {filename}: {len(signal_columns)} signals")
340
341         except Exception as e:
342             print(f"  ✗ Failed to index {filename}: {e}")
343             continue
344
345     print(f"[OK] Total unique signals indexed: {len(signal_index)}")
346
347     # НЕ сохраняем здесь — это делает load_signal_index
348     return signal_index
349
350
351 def load_signal_index(folder: str) -> Dict[str, List[str]]:
352     """
353     Загружает индекс из кэша, но проверяет актуальность.
354     Перестраивает если:
355       - кэша нет
356       - папка изменилась (добавлены/удалены файлы)
357     """
358     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
359         os.path.join(BASE_DIR, folder)
360     )
361
362     # Получаем список CSV файлов и их время модификации
363     def get_folder_state(path: str) -> dict:
364         if not os.path.isdir(path):
365             return {}
366         state = {}
367         for name in os.listdir(path):
368             if name.lower().endswith(".csv"):
369                 filepath = os.path.join(path, name)
370                 state[name] = os.path.getmtime(filepath)
371         return state
372
373     current_state = get_folder_state(folder_abs)
374
375     # Пробуем загрузить кэш
376     if os.path.exists(SIGNAL_INDEX_PATH):
377         try:
378             with open(SIGNAL_INDEX_PATH, "rb") as f:
379                 cached_data = pickle.load(f)
380
381             # Проверяем, есть ли метаданные о состоянии папки
382             if isinstance(cached_data, dict) and "_folder_state" in cached_data:
```

```
383                 cached_state = cached_data["_folder_state"]
384                 cached_index = cached_data["index"]
385
386                 # Сравниваем состояния
387                 if cached_state == current_state:
388                     print(f"[OK] Signal index loaded from cache ({len(cached_index)}
    signals)")
389                     return cached_index
390                 else:
391                     print(f"[INFO] CSV files changed, rebuilding index...")
392             else:
393                 # Старый формат кэша — перестраиваем
394                 print(f"[INFO] Old cache format, rebuilding index...")
395
396         except Exception as e:
397             print(f"[WARN] Failed to load cached index: {e}")
398
399     # Перестраиваем индекс
400     index = build_signal_index(folder)
401
402     # Сохраняем с метаданными
403     try:
404         cache_data = {
405             "index": index,
406             "_folder_state": current_state
407         }
408         with open(SIGNAL_INDEX_PATH, "wb") as f:
409             pickle.dump(cache_data, f)
410         print(f"[OK] Signal index cached with folder state")
411     except Exception as e:
412         print(f"[WARN] Failed to cache signal index: {e}")
413
414     return index
415
416 def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
    pd.DataFrame]:
417     """
418     Загружает только нужные сигналы из только нужных файлов
419     Returns: {signal_name -> DataFrame}
420     """
421     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
422         os.path.join(BASE_DIR, folder)
423     )
424
425     signal_index = STATE.get("signal_index", {})
426     if not signal_index:
427         raise RuntimeError("Signal index not initialized")
428
429     signal_names_set = set(signal_names)
430     found_signals = {}
431     files_to_load = set()
432
433     # Определяем, какие файлы нужно загружать
434     for signal_name in signal_names_set:
435         if signal_name in signal_index:
436             files_to_load.update(signal_index[signal_name])
437
438     print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)}
    files")
439
440     # Загружаем данные из файлов
441     for filepath in files_to_load:
442         try:
443             df = pd.read_csv(
444                 filepath,
```

```
                encoding="ISO-8859-2",
                sep=";"
            )

            # Обработка даты/времени
            df["TIME"] = df["TIME"].str.replace(",", ".", regex=False)
            df["TIME"] = df["TIME"].str.split(".").str[0]
            combined = df["DATE"] + " " + df["TIME"]
            df["datetime"] = pd.to_datetime(
                combined, format="%d.%m.%Y %H:%M:%S", errors="coerce"
            )
            df = df.dropna(subset=["datetime"])
            df = df.drop(['DATE', 'TIME'], axis=1)

            # Сортируем по datetime
            df = df.sort_values("datetime")

            # Извлекаем только нужные сигналы
            available_columns = set(df.columns) & signal_names_set
            for signal_name in available_columns:
                if signal_name not in found_signals:
                    # Сохраняем datetime и значение сигнала
                    found_signals[signal_name] = df[["datetime", signal_name]].copy()
                    found_signals[signal_name].columns = ["datetime", "value"]

        except Exception as e:
            print(f"[WARN] Failed to read {filepath}: {e}")
            continue

    return found_signals




@app.on_event("startup")
def startup():
    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', []))}")

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

@app.get("/api/signals")
def api_signals(q: str = "", limit: int = 50):
    """
    q — маска со * (например *МАА*СР*)
    """
    signals = STATE["signals"] or []
    if not q:
        result = {"items": signals[:limit], "total": len(signals)}
    else:
        import re
```

```
510         escaped = re.escape(q).replace(r"\*", ".*")
511         rx = re.compile("^" + escaped + "$", re.IGNORECASE)
512         items = [s for s in signals if rx.match(s["Tagname"])]
513         result = {"items": items[:max(1, min(limit, 500))], "total": len(items)}
514
515     # Возвращаем с заголовками против кэширования
516     return JSONResponse(
517         content=result,
518         headers={
519             "Cache-Control": "no-cache, no-store, must-revalidate",
520             "Pragma": "no-cache",
521             "Expires": "0"
522         }
523     )
524
525 # Helper: возвращает абсолютный путь к файлу проекта
526 def get_project_path(filename: str):
527     folder = STATE["settings"].get("projectDataFolder")
528     if not folder:
529         raise RuntimeError("projectDataFolder not configured")
530
531     # Нормализуем путь к папке проектов
532     project_dir = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
533
534     # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
535     if '..' in filename or '/' in filename or '\\' in filename:
536         raise HTTPException(status_code=400, detail="Invalid filename")
537
538     path = os.path.join(project_dir, filename)
539     # Проверяем, что итоговый путь лежит внутри разрешенной директории
540     if not path.startswith(project_dir):
541          raise HTTPException(status_code=400, detail="Path traversal attempt")
542
543     return path
544
545 # main.py — исправь endpoint save_project
546
547 @app.post("/api/project/save")
548 async def save_project(request: Request):
549     try:
550         data = await request.json()
551         filename = data.get("filename")
552         content = data.get("content")
553
554         if not filename or not content:
555             raise HTTPException(status_code=400, detail="Filename and content are
    required")
556
557         path = get_project_path(filename)
558
559         # Сохраняем как JSON
560         with open(path, "w", encoding="utf-8") as f:
561             json.dump(content, f, indent=2)
562
563         # ВАЖНО: обновляем кэш ПОСЛЕ закрытия файла!
564         refresh_signals_cache()
565
566         print(f"[OK] Project saved: {filename}, signals cache refreshed:
    {len(STATE['signals'])} signals")
567
568         return {"status": "ok", "message": f"Project saved to {filename}"}
569
570     except HTTPException as e:
571         raise e
```

```
572      except Exception as e:
573          print(f"Error saving project: {e}")
574          raise HTTPException(status_code=500, detail="Internal server error during
     save")
575
576  @app.get("/api/project/load/{filename}")
577  def load_project(filename: str):
578      try:
579          path = get_project_path(filename)
580
581          if not os.path.exists(path):
582              raise HTTPException(status_code=404, detail="Project not found")
583
584          with open(path, "r", encoding="utf-8") as f:
585              content = json.load(f)
586
587          return content
588
589      except HTTPException as e:
590          raise e
591      except Exception as e:
592          print(f"Error loading project: {e}")
593          raise HTTPException(status_code=500, detail="Internal server error during
     load")
594
595  @app.get("/api/formula-templates")
596  def api_formula_templates():
597      return STATE.get("templates") or {"templates": []}
598
599  @app.get("/api/project/list")
600  def list_projects():
601      folder = STATE["settings"].get("projectDataFolder")
602      if not folder:
603          raise HTTPException(status_code=500, detail="Project folder not configured")
604
605      project_dir = folder if os.path.isabs(folder) else
     os.path.normpath(os.path.join(BASE_DIR, folder))
606      os.makedirs(project_dir, exist_ok=True)
607
608      projects = []
609      for fname in sorted(os.listdir(project_dir)):
610          if not fname.endswith(".json"):
611              continue
612          path = os.path.join(project_dir, fname)
613          try:
614              with open(path, "r", encoding="utf-8") as f:
615                  payload = json.load(f)
616          except Exception:
617              continue
618          project_meta = payload.get("project", {})
619          projects.append({
620              "filename": fname,
621              "code": project_meta.get("code") or project_meta.get("tagname") or "",
622              "description": project_meta.get("description") or "",
623              "type": project_meta.get("type") or ""
624          })
625      return {"projects": projects}
626
627  @app.post("/api/signal-data")
628  async def api_signal_data(request: Request):
629      """
630      POST с JSON телом:
631      {
632          "signal_names": ["SIGNAL1", "SIGNAL2", ...],
633          "format": "parquet"  # или "json"
```

```
634      }
635
636      Returns: Parquet файл с данными или JSON
637      """
638      try:
639          data = await request.json()
640          signal_names = data.get("signal_names", [])
641          output_format = data.get("format", "parquet")  # По умолчанию Parquet
642
643          if not signal_names:
644              raise HTTPException(status_code=400, detail="signal_names is required")
645
646          folder = STATE["settings"].get("signalArchiveFolder")
647          if not folder:
648              raise HTTPException(status_code=500, detail="signalArchiveFolder not
     configured")
649
650          # Загружаем данные сигналов
651          signals_data = load_signal_data_optimized(signal_names, folder)
652
653          # Подготавливаем ответ
654          response = {
655              "found": list(signals_data.keys()),
656              "not_found": [s for s in signal_names if s not in signals_data],
657              "format": output_format
658          }
659
660          if not signals_data:
661              raise HTTPException(status_code=404, detail="No signals found")
662
663          # Экспортируем данные в зависимости от формата
664          if output_format == "parquet":
665              return await _export_parquet(signals_data, response)
666          else:
667              return await _export_json(signals_data, response)
668
669      except HTTPException as e:
670          raise e
671      except Exception as e:
672          print(f"Error in api_signal_data: {e}")
673          raise HTTPException(status_code=500, detail=str(e))
674
675  async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
676      """
677      Экспортирует данные в Parquet (НАМНОГО меньше чем JSON!)
678      """
679      from fastapi.responses import FileResponse
680
681      try:
682          # Создаем временный файл
683          with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
684              tmp_path = tmp.name
685
686          # Каждый сигнал сохраняем как отдельную таблицу в Parquet
687          # Используем структуру: datetime, signal_name, value
688          rows = []
689          for signal_name, df in signals_data.items():
690              df_copy = df.copy()
691              df_copy["signal_name"] = signal_name
692              rows.append(df_copy)
693
694          combined = pd.concat(rows, ignore_index=True)
695          combined.to_parquet(tmp_path, compression='snappy', index=False)
696
697          file_size = os.path.getsize(tmp_path)
```

```
698         print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size /
    1024 / 1024:.2f} MB")
699
700         return FileResponse(
701             tmp_path,
702             media_type="application/octet-stream",
703             filename="signal_data.parquet",
704             headers={"X-Signal-Meta": json.dumps(meta)}
705         )
706
707     except Exception as e:
708         print(f"[ERROR] Parquet export failed: {e}")
709         raise
710
711
712 async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
713     """
714     Экспортирует данные в JSON (медленнее и больше, но совместимее)
715     """
716     from fastapi.responses import JSONResponse
717
718     try:
719         # Формируем JSON с каждым сигналом отдельно
720         data_dict = {}
721         for signal_name, df in signals_data.items():
722             df_copy = df.copy()
723             df_copy["datetime"] = df_copy["datetime"].astype(str)
724             data_dict[signal_name] = df_copy.to_dict(orient="records")
725
726         response_data = {
727             **meta,
728             "data": data_dict
729         }
730
731         return JSONResponse(response_data)
732
733     except Exception as e:
734         print(f"[ERROR] JSON export failed: {e}")
735         raise
736
737 # Простое файловое хранилище для сессий визуализации
738 VIS_SESSIONS_DIR = os.path.join(tempfile.gettempdir(), "viz_sessions")
739 os.makedirs(VIS_SESSIONS_DIR, exist_ok=True)
740
741 def _save_viz_session(data: Dict) -> str:
742     token = uuid.uuid4().hex
743     path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
744     with open(path, "wb") as f:
745         pickle.dump(data, f)
746     return token
747
748 def _load_viz_session(token: str) -> Dict:
749     path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
750     if not os.path.exists(path):
751         return None
752     with open(path, "rb") as f:
753         return pickle.load(f)
754
755 @app.post("/api/visualize/session")
756 async def create_visualize_session(payload: Dict):
757     signals = payload.get("signals", [])
758     code = payload.get("code", "")
759     if not isinstance(signals, list):
760         raise HTTPException(status_code=400, detail="signals must be a list")
761     token = _save_viz_session({"signals": signals, "code": code})
```

```
762      return {"token": token}
763
764  @app.get("/api/visualize/session/{token}")
765  def get_visualize_session(token: str):
766      data = _load_viz_session(token)
767      if not data:
768          raise HTTPException(status_code=404, detail="session not found")
769      return data
770
771  @app.post("/api/resolve-signals")
772  async def api_resolve_signals(request: Request):
773      """
774      Разворачивает зависимости сигналов (матрёшку).
775
776      Request body:
777      {
778          "signals": ["SIGNAL1", "SIGNAL2", ...]
779      }
780
781      Response:
782      {
783          "base_signals": ["BASE1", "BASE2", ...],
784          "synthetic_signals": {
785              "SYN1": {"formula": "...", "dependencies": [...]},
786              ...
787          },
788          "computation_order": ["SYN1", "SYN2", ...]
789      }
790      """
791      try:
792          data = await request.json()
793          signal_names = data.get("signals", [])
794
795          print(f"[INFO] Resolving dependencies for signals: {signal_names}")
796
797          # Разворачиваем зависимости
798          base_signals, synthetic_signals = resolve_signal_dependencies(signal_names)
799
800          # Топологическая сортировка для правильного порядка вычисления
801          computation_order = topological_sort_signals(synthetic_signals)
802
803          print(f"[INFO] Base signals: {base_signals}")
804          print(f"[INFO] Synthetic signals: {list(synthetic_signals.keys())}")
805          print(f"[INFO] Computation order: {computation_order}")
806
807          return {
808              "base_signals": list(base_signals),
809              "synthetic_signals": synthetic_signals,
810              "computation_order": computation_order
811          }
812
813      except ValueError as ve:
814          # Циклическая зависимость
815          raise HTTPException(status_code=400, detail=str(ve))
816      except Exception as e:
817          print(f"[ERROR] resolve-signals failed: {e}")
818          raise HTTPException(status_code=500, detail=str(e))
819
820
821  # Раздаём фронтенд
822  WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
823  app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")
824
825  # visualizer_app.py — замени/обнови
826
```

```
827  import pandas as pd
828  import requests
829  import streamlit as st
830  import plotly.express as px
831  import numpy as np
832  import plotly.graph_objects as go
833  from typing import List  # добавь в начало файла если нет
834
835  from code_signal import compute_code_signal, sanitize_numeric_column
836
837  st.set_page_config(page_title="Signal Visualizer", layout="wide")
838  st.title("📊 Визуализация сигналов")
839
840  query_params = st.query_params
841  session_token = query_params.get("session", None)
842  api_url = query_params.get("api_url", "http://localhost:8000")
843
844  signal_codes = query_params.get("signals", [])
845  if isinstance(signal_codes, str):
846      signal_codes = [signal_codes]
847
848  CODE = ""
849  if session_token:
850      try:
851          resp = requests.get(f"{api_url}/api/visualize/session/{session_token}")
852          resp.raise_for_status()
853          payload = resp.json()
854          signal_codes = payload.get("signals", signal_codes)
855          CODE = payload.get("code", CODE)
856      except Exception as e:
857          st.error(f"Не удалось получить данные сессии: {e}")
858
859  if "signals_data" not in st.session_state:
860      st.session_state.signals_data = None
861  if "selected_signals" not in st.session_state:
862      st.session_state.selected_signals = set()
863  if "plot_areas" not in st.session_state:
864      st.session_state.plot_areas = []
865  if "derived_signals" not in st.session_state:
866      st.session_state.derived_signals = {}
867  if "code_signal_name" not in st.session_state:
868      st.session_state.code_signal_name = None
869  if "synthetic_computed" not in st.session_state:
870      st.session_state.synthetic_computed = {}  # уже вычисленные синтетические сигналы
871  if "signal_groups" not in st.session_state:
872      st.session_state.signal_groups = {"project": set(), "dependencies": set()}
873
874
875  def load_base_signals_data(signal_names: List[str]) -> pd.DataFrame | None:
876      """Загружает данные базовых сигналов из архива"""
877      if not signal_names:
878          return None
879
880      try:
881          response = requests.post(
882              f"{api_url}/api/signal-data",
883              json={"signal_names": signal_names, "format": "json"},
884          )
885          response.raise_for_status()
886          result = response.json()
887
888          found = result.get("found", [])
889          not_found = result.get("not_found", [])
890          data_dict = result.get("data", {})
891
```

```
892          if not_found:
893              st.warning(f"⚠️ Базовые сигналы не найдены в архиве: {',
     '.join(not_found)}")
894
895          if not data_dict:
896              return None
897
898          frames = []
899          for sig, records in data_dict.items():
900              if not records:
901                  continue
902              df = pd.DataFrame(records)
903              if "datetime" not in df or "value" not in df:
904                  continue
905              df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
906              df = df.dropna(subset=["datetime"])
907              df = df.set_index("datetime").sort_index()
908              df = df.rename(columns={"value": sig})
909              frames.append(df[[sig]])
910
911          if not frames:
912              return None
913
914          return pd.concat(frames, axis=1).sort_index()
915
916      except Exception as exc:
917          st.error(f"❌ Ошибка загрузки базовых сигналов: {exc}")
918          return None
919
920
921  # visualizer_app.py — замени функцию resolve_and_load_all_signals
922
923  def resolve_and_load_all_signals(input_signals: List[str]) -> tuple[pd.DataFrame |
     None, List[str], List[str]]:
924      """
925      Разворачивает зависимости и загружает все сигналы (базовые + синтетические).
926
927      Returns:
928          df_all: DataFrame со всеми сигналами
929          found: список найденных сигналов
930          not_found: список ненайденных сигналов
931      """
932      if not input_signals:
933          return None, [], []
934
935      try:
936          # 1. Разворачиваем зависимости через API
937          with st.spinner("🔍 Разворачиваем зависимости сигналов..."):
938              resolve_resp = requests.post(
939                  f"{api_url}/api/resolve-signals",
940                  json={"signals": input_signals}
941              )
942              resolve_resp.raise_for_status()
943              resolve_data = resolve_resp.json()
944
945          base_signals = resolve_data.get("base_signals", [])
946          synthetic_signals = resolve_data.get("synthetic_signals", {})
947          computation_order = resolve_data.get("computation_order", [])
948
949          # === СОХРАНЯЕМ ГРУППИРОВКУ СИГНАЛОВ ===
950          # Сигналы из текущего проекта (исходные входные)
951          project_signals = set(input_signals)
952
953          # Сигналы из зависимостей (все остальные)
954          dependency_signals = set()
```

```
955         for syn_name, syn_data in synthetic_signals.items():
956             if syn_name not in project_signals:
957                 dependency_signals.add(syn_name)
958             for dep in syn_data.get("dependencies", []):
959                 if dep not in project_signals:
960                     dependency_signals.add(dep)
961 
962         # Также добавляем базовые сигналы, которые не из проекта
963         for bs in base_signals:
964             if bs not in project_signals:
965                 dependency_signals.add(bs)
966 
967         # Сохраняем в session_state для использования в сайдбаре
968         st.session_state.signal_groups = {
969             "project": project_signals,       # входные сигналы текущего проекта
970             "dependencies": dependency_signals # сигналы из развёрнутых зависимостей
971         }
972 
973         st.info(f"📊 Сигналов проекта: {len(project_signals)} | Из зависимостей:
    {len(dependency_signals)}")
974 
975         if synthetic_signals:
976             with st.expander("🔗 Граф зависимостей синтетических сигналов"):
977                 for syn_name in computation_order:
978                     deps = synthetic_signals[syn_name].get("dependencies", [])
979                     marker = "📌" if syn_name in project_signals else "🔗"
980                     st.text(f"  {marker} {syn_name} ← {deps}")
981 
982         # 2. Загружаем базовые сигналы
983         df_all = None
984         found_signals = []
985         not_found_signals = []
986 
987         if base_signals:
988             with st.spinner(f"📥 Загружаем {len(base_signals)} базовых сигналов..."):
989                 df_all = load_base_signals_data(base_signals)
990                 if df_all is not None:
991                     found_signals = list(df_all.columns)
992                     not_found_signals = [s for s in base_signals if s not in
    df_all.columns]
993 
994         if df_all is None:
995             df_all = pd.DataFrame()
996 
997         # 3. Вычисляем синтетические сигналы в правильном порядке
998         if computation_order:
999             with st.spinner(f"⚙️ Вычисляем {len(computation_order)} синтетических
    сигналов..."):
1000                 progress_bar = st.progress(0)
1001 
1002                 for idx, syn_name in enumerate(computation_order):
1003                     syn_data = synthetic_signals[syn_name]
1004                     formula = syn_data.get("formula", "")
1005 
1006                     if not formula:
1007                         st.warning(f"⚠️ Синтетический сигнал '{syn_name}' не имеет
    формулы")
1008                         continue
1009 
1010                     if df_all.empty:
1011                         st.warning(f"⚠️ Нет данных для вычисления '{syn_name}'")
1012                         continue
1013 
1014                     try:
1015                         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)}")


    # --- остальной код без изменений, начиная с get_all_signals_df ---

    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()

```

```
1080
1081  def compute_stats_numeric(df: pd.DataFrame) -> pd.DataFrame:
1082      if df is None or df.empty:
1083          return pd.DataFrame()
1084
1085      numeric = df.apply(sanitize_numeric_column)
1086      valid_cols = [col for col in numeric.columns if numeric[col].count() > 0]
1087      if not valid_cols:
1088          return pd.DataFrame()
1089
1090      numeric = numeric[valid_cols]
1091      stats = pd.DataFrame(index=numeric.columns)
1092      stats["count"] = numeric.count()
1093      stats["min"] = numeric.min()
1094      stats["max"] = numeric.max()
1095      stats["mean"] = numeric.mean()
1096      stats["std"] = numeric.std()
1097      stats["median"] = numeric.median()
1098
1099      starts, ends = [], []
1100      for col in numeric.columns:
1101          series = numeric[col].dropna()
1102          starts.append(series.index.min() if not series.empty else pd.NaT)
1103          ends.append(series.index.max() if not series.empty else pd.NaT)
1104
1105      stats["start"] = starts
1106      stats["end"] = ends
1107      return stats
1108
1109
1110  def make_unique_name(base_name: str) -> str:
1111      existing = set()
1112      if st.session_state.signals_data is not None:
1113          existing |= set(st.session_state.signals_data.columns)
1114      existing |= set(st.session_state.derived_signals.keys())
1115      if base_name not in existing:
1116          return base_name
1117      idx = 2
1118      while f"{base_name}_{idx}" in existing:
1119          idx += 1
1120      return f"{base_name}_{idx}"
1121
1122
1123  if signal_codes and st.session_state.signals_data is None:
1124      with st.spinner("Загружаем данные сигналов..."):
1125          df_all, found_codes, not_found_codes =
      resolve_and_load_all_signals(signal_codes)
1126          st.success(f"✅ Загружено сигналов: {len(found_codes)}")
1127          if not_found_codes:
1128              st.warning(f"⚠️ Не найдены: {', '.join(not_found_codes)}")
1129
1130  # --- синтетический сигнал из CODE (считаем один раз, потом не пересчитываем) ---
1131  code_signal_name = st.session_state.code_signal_name
1132  df_for_code = get_all_signals_df(exclude={code_signal_name} if code_signal_name else
      None)
1133
1134  # Ключ "какой CODE мы уже считали" (можно оставить просто CODE; session_token добавил
      на всякий)
1135  code_key = (session_token, CODE)
1136
1137  already_have_series = (
1138      st.session_state.code_signal_name is not None
1139      and st.session_state.code_signal_name in st.session_state.derived_signals
1140  )
1141
```

```
1142  if CODE and df_for_code is not None:
1143      need_recalc = (st.session_state.get("code_key") != code_key) or (not
      already_have_series)
1144
1145      if need_recalc:
1146          try:
1147              synthetic_series = compute_code_signal(
1148                  CODE,
1149                  df_for_code,
1150                  warn_callback=lambda msg: st.warning(msg, icon="⚠️"),
1151              )
1152              target_name = code_signal_name or make_unique_name("CODE_RESULT")
1153              synthetic_series.name = target_name
1154
1155              st.session_state.derived_signals[target_name] = pd.DataFrame({target_name:
      synthetic_series})
1156              st.session_state.code_signal_name = target_name
1157              st.session_state.selected_signals.add(target_name)
1158
1159              st.session_state.code_key = code_key
1160              st.success(f"Синтетический сигнал обновлён: {target_name}")
1161          except Exception as exc:
1162              st.warning(f"Не удалось вычислить CODE: {exc}")
1163
1164  elif not CODE:
1165      # если CODE исчез — удаляем синтетический сигнал и сбрасываем ключ
1166      if code_signal_name:
1167          st.session_state.derived_signals.pop(code_signal_name, None)
1168          st.session_state.selected_signals.discard(code_signal_name)
1169          st.session_state.code_signal_name = None
1170      st.session_state.code_key = None
1171
1172  # --- итоговый DataFrame со всеми сигналами ---
1173  df_all_signals = get_all_signals_df()
1174
1175  with st.sidebar:
1176      st.header("Выбор сигналов")
1177
1178      if df_all_signals is not None:
1179          available_signals = df_all_signals.columns.tolist()
1180
1181          # Получаем группы сигналов
1182          signal_groups = st.session_state.get("signal_groups", {
1183              "project": set(available_signals),
1184              "dependencies": set()
1185          })
1186
1187          project_signals = [s for s in available_signals if s in
      signal_groups.get("project", set())]
1188          dependency_signals = [s for s in available_signals if s in
      signal_groups.get("dependencies", set())]
1189
1190          # === СИГНАЛЫ ПРОЕКТА ===
1191          if project_signals:
1192              st.subheader("📌 Сигналы проекта")
1193              for signal in project_signals:
1194                  # Помечаем синтетические сигналы
1195                  is_synthetic = signal in st.session_state.get("synthetic_computed",
      {})
1196                  label = f"⚙️ {signal}" if is_synthetic else signal
1197
1198                  checked = st.checkbox(
1199                      label,
1200                      value=(signal in st.session_state.selected_signals),
1201                      key=f"proj_{signal}"
```

```
1202                 )
1203                 if checked:
1204                     st.session_state.selected_signals.add(signal)
1205                 else:
1206                     st.session_state.selected_signals.discard(signal)
1207 
1208         # === СИГНАЛЫ ИЗ ЗАВИСИМОСТЕЙ ===
1209         if dependency_signals:
1210             st.divider()
1211             with st.expander(f"🔗 Из зависимостей ({len(dependency_signals)})",
     expanded=False):
1212                 for signal in dependency_signals:
1213                     is_synthetic = signal in
     st.session_state.get("synthetic_computed", {})
1214                     label = f"⚙️ {signal}" if is_synthetic else signal
1215 
1216                     checked = st.checkbox(
1217                         label,
1218                         value=(signal in st.session_state.selected_signals),
1219                         key=f"dep_{signal}"
1220                     )
1221                     if checked:
1222                         st.session_state.selected_signals.add(signal)
1223                     else:
1224                         st.session_state.selected_signals.discard(signal)
1225 
1226         # === Быстрые действия ===
1227         st.divider()
1228         col1, col2 = st.columns(2)
1229         with col1:
1230             if st.button("✅ Все проекта"):
1231                 st.session_state.selected_signals.update(project_signals)
1232                 st.rerun()
1233         with col2:
1234             if st.button("❌ Снять все"):
1235                 st.session_state.selected_signals.clear()
1236                 st.rerun()
1237 
1238         st.divider()
1239         st.subheader("Создать обрезанный сигнал")
1240 
1241         base_df = st.session_state.signals_data
1242         if base_df is not None and not base_df.empty:
1243             base_choice = st.selectbox("Исходный сигнал", base_df.columns)
1244             series = base_df[base_choice].dropna()
1245             if not series.empty:
1246                 col1, col2 = st.columns(2)
1247                 with col1:
1248                     start_date = st.date_input(
1249                         "Начало",
1250                         value=series.index.min().date(),
1251                     )
1252                 with col2:
1253                     end_date = st.date_input(
1254                         "Конец",
1255                         value=series.index.max().date(),
1256                     )
1257 
1258                 start_ts = pd.Timestamp(start_date)
1259                 end_ts = pd.Timestamp(end_date) + pd.Timedelta(days=1) - pd.Timedelta(
1260                     microseconds=1
1261                 )
1262 
1263                 default_name = f"{base_choice}__{start_ts.date()}_{end_ts.date()}"
1264                 new_name = st.text_input("Имя нового сигнала", value=default_name)
```

```
1265
1266                 col3, col4 = st.columns(2)
1267                 if col3.button("Создать"):
1268                     name_unique = make_unique_name(new_name.strip())
1269                     cut_series = series[(series.index >= start_ts) & (series.index <=
     end_ts)]
1270                     if cut_series.empty:
1271                         st.warning("В выбранном диапазоне нет точек.")
1272                     else:
1273                         st.session_state.derived_signals[name_unique] = pd.DataFrame(
1274                             {name_unique: cut_series}
1275                         )
1276                         st.success(f"Создан обрезанный сигнал: {name_unique}")
1277                         st.rerun()
1278                 if col4.button("Очистить все обрезанные"):
1279                     st.session_state.derived_signals = {
1280                         k: v
1281                         for k, v in st.session_state.derived_signals.items()
1282                         if k == st.session_state.code_signal_name
1283                     }
1284                     st.session_state.selected_signals = {
1285                         sig
1286                         for sig in st.session_state.selected_signals
1287                         if (st.session_state.signals_data is not None and sig in
     st.session_state.signals_data.columns)
1288                         or sig == st.session_state.code_signal_name
1289                     }
1290                     st.rerun()
1291
1292         if st.session_state.derived_signals:
1293             st.subheader("Удалить обрезанный/синтетический сигнал")
1294             derived_names = [name for name in st.session_state.derived_signals.keys()]
1295             delete_candidate = st.selectbox("Выберите", ["—"] + derived_names)
1296             if st.button("Удалить выбранный") and delete_candidate != "—":
1297                 st.session_state.derived_signals.pop(delete_candidate, None)
1298                 st.session_state.selected_signals.discard(delete_candidate)
1299                 if delete_candidate == st.session_state.code_signal_name:
1300                     st.session_state.code_signal_name = None
1301                 st.rerun()
1302
1303         st.divider()
1304         st.subheader("Области построения")
1305         col_a, col_b = st.columns(2)
1306         if col_a.button("➕ Добавить график"):
1307             new_id = max([area.get("id", 0) for area in st.session_state.plot_areas] +
     [0]) + 1
1308             st.session_state.plot_areas.append({"id": new_id, "signals": []})
1309             st.rerun()
1310         if col_b.button("❌ Очистить все"):
1311             st.session_state.plot_areas = []
1312             st.session_state.selected_signals = set()
1313             st.rerun()
1314     else:
1315         st.info("📥 Данные сигналов еще не загружены.")
1316
1317 if df_all_signals is not None and st.session_state.selected_signals:
1318     if not st.session_state.plot_areas:
1319         st.session_state.plot_areas.append(
1320             {"id": 1, "signals": list(st.session_state.selected_signals)}
1321         )
1322
1323     for i, plot_area in enumerate(st.session_state.plot_areas):
1324         with st.container():
1325             col1, col2 = st.columns([3, 1])
1326             with col1:
```

```
1327                 st.subheader(f"График #{plot_area['id']}")
1328             with col2:
1329                 if st.button("Удалить", key=f"remove_area_{i}"):
1330                     st.session_state.plot_areas.pop(i)
1331                     st.rerun()
1332 
1333             selected = st.multiselect(
1334                 "Выберите сигнал(ы):",
1335                 list(st.session_state.selected_signals),
1336                 default=plot_area.get("signals", []),
1337                 key=f"signals_sel_{i}",
1338             )
1339             st.session_state.plot_areas[i]["signals"] = selected
1340 
1341             if selected:
1342                 df_plot = df_all_signals[selected].copy()
1343 
1344                 # Для графика приводим к числам (поддержка запятых)
1345                 df_plot_num = df_plot.apply(sanitize_numeric_column)
1346 
1347                 valid_index = df_plot_num.dropna(how="all").index
1348                 if len(valid_index) == 0:
1349                     st.warning("Нет числовых данных для выбранных сигналов.")
1350                 else:
1351                     ts_idx = st.slider(
1352                         "Вертикальная линия (время)",
1353                         min_value=0,
1354                         max_value=len(valid_index) - 1,
1355                         value=len(valid_index) - 1,
1356                         key=f"vline_{i}",
1357                     )
1358                     ts = valid_index[ts_idx]
1359 
1360                     # график с вертикальной линией
1361                     fig = px.line(
1362                         df_plot_num,
1363                         x=df_plot_num.index,
1364                         y=selected,
1365                         title=f"График #{plot_area['id']}",
1366                         render_mode="webgl"
1367                     )
1368                     fig.add_vline(x=ts, line_width=2, line_dash="dash",
     line_color="red")
1369                     fig.update_layout(
1370                         uirevision=f"plot_area_{plot_area['id']}",
1371                         height=650,
1372                         legend_title_text="Сигналы",
1373                         xaxis_title="Время",
1374                         yaxis_title="Значение",
1375                         margin=dict(l=20, r=20, t=40, b=20),
1376                     )
1377                     st.plotly_chart(fig, use_container_width=True)
1378 
1379                     # значения на линии
1380                     nearest =
     df_plot_num.reindex(df_plot_num.index.union([ts])).sort_index()
1381                     nearest = nearest.ffill().loc[ts]
1382 
1383                     # статистика + колонка значений на линии
1384                     st.markdown("**📊 Статистика (по всему сигналу):**")
1385                     stats_df = compute_stats_numeric(df_plot)
1386                     if stats_df.empty:
1387                         st.info("Нет числовых данных для расчёта статистики.")
1388                     else:
1389                         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_at_line": "{:.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 (вкл. обрезанные/синтет.)",
    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 (оригинал)"):
            st.code(CODE, language="text")


    #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."""
```

```
1454
1455  def sanitize_numeric_column(series: pd.Series) -> pd.Series:
1456      if series.dtype.kind in ("i", "u", "f"):
1457          return series
1458      text = series.astype(str).str.replace(",", ".", regex=False)
1459      return pd.to_numeric(text, errors="coerce")
1460
1461
1462  def evaluate_code_expression(code_str: str, df_all: pd.DataFrame) -> Tuple[pd.Series,
      List[str]]:
1463      if df_all is None or df_all.empty:
1464          raise CodeEvaluationError("Нет данных для расчёта синтетического сигнала.")
1465      if not code_str or not code_str.strip():
1466          raise CodeEvaluationError("Строка CODE пуста.")
1467
1468      index = df_all.index
1469      numeric_df = df_all.apply(sanitize_numeric_column)
1470      series_map = {col: numeric_df[col] for col in numeric_df.columns}
1471      warnings: List[str] = []
1472
1473      # ---------- обработка «неправильных» имён сигналов ----------
1474      safe_name_map: Dict[str, str] = {}
1475      used_safe_names = set()
1476
1477      def _make_safe_name(original: str, idx: int) -> str:
1478          base = re.sub(r"\W", "_", original)
1479          if not base or not re.match(r"[A-Za-z_]", base):
1480              base = f"SIG_{idx}"
1481          while base in used_safe_names:
1482              base += "_"
1483          used_safe_names.add(base)
1484          return base
1485
1486      sorted_signals = sorted(series_map.keys(), key=len, reverse=True)
1487      for idx, sig_name in enumerate(sorted_signals):
1488          safe = _make_safe_name(sig_name, idx)
1489          safe_name_map[sig_name] = safe
1490
1491      def _replace_signal_names(expr: str) -> str:
1492          result = []
1493          i = 0
1494          in_string = False
1495          string_char = ""
1496
1497          while i < len(expr):
1498              ch = expr[i]
1499              if in_string:
1500                  result.append(ch)
1501                  if ch == string_char and expr[i - 1] != "\\":
1502                      in_string = False
1503                  i += 1
1504                  continue
1505
1506              if ch in ("'", '"'):
1507                  in_string = True
1508                  string_char = ch
1509                  result.append(ch)
1510                  i += 1
1511                  continue
1512
1513              matched = None
1514              for name in sorted_signals:
1515                  if expr.startswith(name, i):
1516                      matched = name
1517                      break
```

```
1518              if matched:
1519                  result.append(safe_name_map[matched])
1520                  i += len(matched)
1521              else:
1522                  result.append(ch)
1523                  i += 1
1524
1525          return "".join(result)
1526
1527      # ---------- вспомогательные функции ----------
1528      def _ensure_series(value) -> pd.Series:
1529          if isinstance(value, pd.Series):
1530              return value.reindex(index)
1531          if isinstance(value, pd.DataFrame):
1532              if value.shape[1] == 1:
1533                  return value.iloc[:, 0].reindex(index)
1534              raise CodeEvaluationError("Невозможно привести DataFrame с несколькими
      колонками к Series.")
1535          if isinstance(value, (list, tuple, np.ndarray)):
1536              arr = np.asarray(value, dtype=float)
1537              if arr.size == 1:
1538                  arr = np.full(len(index), arr.item())
1539              elif arr.shape[0] != len(index):
1540                  return pd.Series(np.nan, index=index)
1541              return pd.Series(arr, index=index)
1542          if value is None or np.isscalar(value):
1543              return pd.Series(value, index=index)
1544          try:
1545              return pd.Series(value, index=index)
1546          except Exception as exc:
1547              raise CodeEvaluationError(f"Невозможно преобразовать значение '{value}' к
      Series.") from exc
1548
1549      def _aggregate_nanfunc(func, args, empty_value=np.nan):
1550          if not args:
1551              return pd.Series(empty_value, index=index)
1552          stacked = np.vstack([_ensure_series(arg).values for arg in args])
1553          return pd.Series(func(stacked, axis=0), index=index)
1554
1555      def GETPOINT(*_):
1556          if "GETPOINT" not in warnings:
1557              warnings.append("GETPOINT пока не поддержан — возвращается NaN.")
1558          return pd.Series(np.nan, index=index)
1559
1560      def PREV(param_name):
1561          if param_name not in series_map:
1562              return pd.Series(np.nan, index=index)
1563          return series_map[param_name].shift(1)
1564
1565      def _history_series(param_name):
1566          if param_name not in series_map:
1567              return None
1568          return series_map[param_name]
1569
1570      def _history_window(period):
1571          try:
1572              minutes = int(period)
1573          except (TypeError, ValueError):
1574              return None
1575          if minutes <= 0:
1576              return None
1577          return f"{minutes}min"
1578
1579      def _history_apply(param_name, period, fn):
1580          s = _history_series(param_name)
```

```
1581          window = _history_window(period)
1582          if s is None or window is None:
1583              return pd.Series(np.nan, index=index)
1584          return fn(s.rolling(window))
1585
1586      HISTORYAVG = lambda n, p: _history_apply(n, p, lambda r: r.mean())
1587      HISTORYCOUNT = lambda n, p: _history_apply(n, p, lambda r: r.count())
1588      HISTORYSUM = lambda n, p: _history_apply(n, p, lambda r: r.sum())
1589      HISTORYMAX = lambda n, p: _history_apply(n, p, lambda r: r.max())
1590      HISTORYMIN = lambda n, p: _history_apply(n, p, lambda r: r.min())
1591      HISTORYDIFF = lambda n, p: _history_apply(n, p, lambda r: r.max() - r.min())
1592
1593      def HISTORYGRADIENT(param_name, period):
1594          s = _history_series(param_name)
1595          window = _history_window(period)
1596          if s is None or window is None:
1597              return pd.Series(np.nan, index=index)
1598
1599          def slope(window_series: pd.Series):
1600              valid = window_series.dropna()
1601              if len(valid) < 2:
1602                  return np.nan
1603              x = valid.index.view(np.int64).astype(float) / 1e9
1604              y = valid.values.astype(float)
1605              x_mean = x.mean()
1606              y_mean = y.mean()
1607              denom = np.sum((x - x_mean) ** 2)
1608              if denom == 0:
1609                  return np.nan
1610              return np.sum((x - x_mean) * (y - y_mean)) / denom
1611
1612          return s.rolling(window).apply(slope, raw=False)
1613
1614      def ROUND(a, b=0):
1615          a_values = _ensure_series(a).values
1616          b_values = _ensure_series(b).values
1617          decimals = [
1618              0 if np.isnan(dec) else int(round(dec))
1619              for dec in b_values
1620          ]
1621          rounded = np.array([
1622              np.round(val, dec) if not np.isnan(val) else np.nan
1623              for val, dec in zip(a_values, decimals)
1624          ])
1625          return pd.Series(rounded, index=index)
1626
1627      # ---------- окружение eval ----------
1628      env = {
1629          "np": np,
1630          "ABS": lambda a: pd.Series(np.abs(_ensure_series(a).values), index=index),
1631          "POW": lambda a, b: pd.Series(np.power(_ensure_series(a).values,
      _ensure_series(b).values), index=index),
1632          "MIN": lambda *args: _aggregate_nanfunc(np.nanmin, args),
1633          "MAX": lambda *args: _aggregate_nanfunc(np.nanmax, args),
1634          "AVG": lambda *args: _aggregate_nanfunc(np.nanmean, args, empty_value=0.0),
1635          "MED": lambda *args: _aggregate_nanfunc(np.nanmedian, args),
1636          "ROUND": ROUND,
1637          "WHEN": lambda cond, t_val, f_val: pd.Series(
1638              np.where(_ensure_series(cond).astype(bool).values,
1639                       _ensure_series(t_val).values,
1640                       _ensure_series(f_val).values),
1641              index=index,
1642          ),
1643          "LOG": lambda x: pd.Series(np.log(_ensure_series(x).values), index=index),
1644          # Логарифм по основанию 10 (если нужен)
```

```
        "LOG10": lambda x: pd.Series(np.log10(_ensure_series(x).values), index=index),
        "PREV": PREV,
        "HISTORYAVG": HISTORYAVG,
        "HISTORYCOUNT": HISTORYCOUNT,
        "HISTORYSUM": HISTORYSUM,
        "HISTORYMAX": HISTORYMAX,
        "HISTORYMIN": HISTORYMIN,
        "HISTORYDIFF": HISTORYDIFF,
        "HISTORYGRADIENT": HISTORYGRADIENT,
        "GETPOINT": GETPOINT,
    }

    for original_name, safe_name in safe_name_map.items():
        env[safe_name] = series_map[original_name]

    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

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>
```

```
1710             <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</
     button>
1711             <button class="menu-btn" id="btn-visualize">📈 Визуализировать</button>
1712             <div class="menu-separator"></div>
1713             <div class="zoom-controls">
1714                 <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
1715                 <span id="zoom-level">100%</span>
1716                 <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
1717                 <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
1718                 <button class="menu-btn" id="btn-zoom-reset">1:1</button>
1719             </div>
1720             <input type="file" id="file-input" accept=".json">
1721         </div>
1722 
1723         <div id="main">
1724             <div id="palette">
1725                 <h3>📦 Элементы</h3>
1726                 <div class="palette-section">
1727                     <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>
1728 
1729                     <div class="palette-item" data-type="group">
1730                         <svg viewBox="0 0 60 40">
1731                         <rect x="6" y="8" width="48" height="24" rx="4"
1732                                 fill="none" stroke="#6b7280" stroke-width="2" stroke-
     dasharray="4,2"/>
1733                         <text x="14" y="25" fill="#6b7280" font-size="10" font-
     weight="bold">GROUP</text>
1734                         </svg>
1735                         <div class="palette-item-name">Группа</div>
1736                     </div>
1737                     </div>
1738 
1739                 <div class="palette-section">
1740                     <div class="palette-section-title">ВХОДЫ</div>
1741 
1742                     <div class="palette-item" data-type="input-signal">
1743                         <svg viewBox="0 0 60 40">
1744                             <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460"
     stroke="#4a90d9" stroke-width="2"/>
1745                             <text x="12" y="24" fill="#eee" font-size="10">IN</text>
1746                         </svg>
1747                         <div class="palette-item-name">Входной сигнал</div>
1748                     </div>
1749                 </div>
1750                 <div class="palette-section">
1751                     <div class="palette-section-title">ВЫХОДЫ</div>
1752 
1753                     <div class="palette-item" data-type="output">
1754                         <svg viewBox="0 0 60 40">
1755                             <rect x="5" y="5" width="50" height="30" rx="6"
     fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
1756                             <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</
     text>
1757                         </svg>
1758                         <div class="palette-item-name">Выход</div>
1759                     </div>
1760                 </div>
1761 
1762                 <div class="palette-section">
1763                     <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>
1764 
1765                     <div class="palette-item" data-type="and">
1766                         <svg viewBox="0 0 60 40">
1767                             <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"/>
                            <text x="8" y="25" fill="#f59e0b" font-size="10" font-
    weight="bold">✓/✗</text>
                        </svg>
                        <div class="palette-item-name">Сепаратор</div>
                    </div>
                </div>

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

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

```
                        <div class="palette-item-name">Константа</div>
                    </div>

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

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

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

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

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

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

    <!-- Модальное окно свойств проекта -->
    <div id="project-modal-overlay" class="modal-overlay-class">
        <div id="project-modal" class="modal-class">
            <h3>Свойства проекта</h3>
            <div id="project-modal-content"></div>
            <div class="modal-buttons">
```

```
                <button class="modal-btn cancel" id="project-modal-cancel">Отмена</button>
                <button class="modal-btn save" id="project-modal-save">Сохранить</button>
            </div>
        </div>
    </div>

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

            <div class="project-list__toolbar">
            <input id="project-search" type="text" placeholder="Фильтр по имени или описанию…" />
            <button id="project-refresh" class="btn btn-secondary">Обновить</button>
            </div>

            <div class="project-list__table-container">
            <table class="project-list__table">
                <thead>
                <tr>
                    <th>Файл</th>
                    <th>Tagname</th>
                    <th>Description</th>
                    <th>Тип</th>
                </tr>
                </thead>
                <tbody id="project-list-body">
                <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
```

```
                </tbody>
            </table>
            </div>

            <div class="modal__actions">
                <button id="project-cancel" class="btn btn-secondary">Отмена</button>
                <button id="project-load" class="btn btn-primary" disabled>Загрузить</
button>
            </div>
        </div>
    </div>

</body>
</html>

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

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

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

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

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

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

openSignalVisualizer() {
```

```
2009      try {
2010          // 1) Собираем входные сигналы
2011          const signals = Object.values(AppState.elements)
2012              .filter(e => e && e.type === 'input-signal')
2013              .map(e => e.props?.name || e.id);
2014          const uniqSignals = [...new Set(signals)];
2015
2016          if (uniqSignals.length === 0) {
2017              alert('Нет входных сигналов в схеме.');
2018              return;
2019          }
2020
2021          // 2) Генерируем код
2022          let codeStr = '';
2023          if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
      {
2024              codeStr = CodeGen.generate() || '';
2025          }
2026
2027          // 3) Определяем URL-ы динамически
2028          const currentHost = window.location.hostname; // IP или домен сервера
2029          const apiPort = window.location.port || 8000;
2030          const visualizerPort = Settings.config?.visualizerPort || 8501;
2031
2032          const apiUrl = `http://${currentHost}:${apiPort}`;
2033          const visualizerBase = `http://${currentHost}:${visualizerPort}`;
2034
2035          console.log('API URL:', apiUrl);
2036          console.log('Visualizer URL:', visualizerBase);
2037
2038          // 4) Создаём сессию на backend
2039          fetch('/api/visualize/session', {
2040              method: 'POST',
2041              headers: { 'Content-Type': 'application/json' },
2042              body: JSON.stringify({ signals: uniqSignals, code: codeStr })
2043          })
2044          .then(r => {
2045              if (!r.ok) throw new Error('Failed to create visualize session');
2046              return r.json();
2047          })
2048          .then(data => {
2049              const token = data.token;
2050              const params = new URLSearchParams();
2051              params.set('session', token);
2052              params.set('api_url', apiUrl);
2053
2054              const visualizerUrl = `${visualizerBase}/?${params.toString()}`;
2055              console.log('Opening visualizer:', visualizerUrl);
2056              window.open(visualizerUrl, '_blank');
2057          })
2058          .catch(err => {
2059              console.error(err);
2060              alert('Не удалось открыть визуализатор: ' + err.message);
2061          });
2062
2063      } catch (e) {
2064          console.error(e);
2065          alert('Ошибка при подготовке визуализации: ' + e.message);
2066      }
2067  },
2068
2069      /**
2070       * Отмена состояния drag из палитры (helper)
2071       */
2072      cancelPaletteDrag() {
```

```
2073          if (AppState.dragPreview) {
2074              try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
2075              AppState.dragPreview = null;
2076          }
2077          AppState.isDraggingFromPalette = false;
2078          AppState.dragType = null;
2079      },
2080
2081      /**
2082       * Настройка счётчика выходов в меню
2083       */
2084      setupOutputCounter() {
2085          // Не создавать повторно, если уже есть
2086          if (document.getElementById('btn-outputs')) return;
2087
2088          const menu = document.getElementById('menu');
2089
2090          // Создаём кнопку с счётчиком выходов
2091          const outputBtn = document.createElement('button');
2092          outputBtn.className = 'menu-btn output-btn';
2093          outputBtn.id = 'btn-outputs';
2094          outputBtn.innerHTML = `
2095              📤 Выходы
2096              <span id="output-counter" class="output-counter">0</span>
2097          `;
2098
2099          // Вставляем после кнопки свойств проекта
2100          const projectBtn = document.getElementById('btn-project-settings');
2101          if (projectBtn) {
2102              projectBtn.after(outputBtn);
2103          } else {
2104              menu.appendChild(outputBtn);
2105          }
2106
2107          outputBtn.addEventListener('click', () => {
2108              Modal.showProjectPropertiesModal();
2109          });
2110      },
2111
2112      /**
2113       * Настройка drag & drop из палитры
2114       */
2115      setupPaletteDragDrop() {
2116          document.querySelectorAll('.palette-item').forEach(item => {
2117              item.addEventListener('mousedown', (e) => {
2118                  // Только левая кнопка мыши должна запускать drag из палитры
2119                  if (e.button !== 0) return;
2120                  e.preventDefault();
2121
2122                  AppState.isDraggingFromPalette = true;
2123                  AppState.dragType = item.dataset.type;
2124
2125                  AppState.dragPreview = document.createElement('div');
2126                  AppState.dragPreview.className = 'drag-preview';
2127                  AppState.dragPreview.textContent =
      ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
2128                  AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2129                  AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2130                  document.body.appendChild(AppState.dragPreview);
2131              });
2132          });
2133      },
2134
2135      /**
2136       * Глобальные обработчики мыши
```

```
2137       */
2138  /**
2139   * Глобальные обработчики мыши
2140   */
2141  setupGlobalMouseHandlers() {
2142      document.addEventListener('mousemove', (e) => {
2143          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
2144              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
2145              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
2146          }
2147          if (AppState.resizing) {
2148              Elements.handleResize(e);
2149              return;
2150          }
2151          if (AppState.draggingElement) {
2152              Elements.handleDrag(e);
2153          }
2154          if (AppState.tempLine && AppState.connectingFrom) {
2155              Connections.drawTempConnection(e);
2156          }
2157      });
2158
2159      document.addEventListener('mouseup', (e) => {
2160          if (AppState.resizing) {
2161              AppState.resizing = null;
2162              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
2163          }
2164
2165          if (AppState.isDraggingFromPalette) {
2166              try {
2167                  if (AppState.dragPreview) {
2168                      AppState.dragPreview.remove();
2169                      AppState.dragPreview = null;
2170                  }
2171
2172                  const container = document.getElementById('workspace-container');
2173                  const rect = container.getBoundingClientRect();
2174
2175                  if (e.clientX >= rect.left && e.clientX <= rect.right &&
2176                      e.clientY >= rect.top && e.clientY <= rect.bottom) {
2177
2178                      const canvasPos = screenToCanvas(e.clientX, e.clientY);
2179                      const config = ELEMENT_TYPES[AppState.dragType];
2180                      if (config) {
2181                          const defaultWidth = config.minWidth || 120;
2182                          const defaultHeight = config.minHeight || 60;
2183
2184                          // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
      обработать
2185                          const newElement = Elements.addElement(
2186                              AppState.dragType,
2187                              canvasPos.x - defaultWidth / 2,
2188                              canvasPos.y - defaultHeight / 2
2189                          );
2190
2191                          if (newElement && typeof Outputs !== 'undefined') {
2192                              Outputs.updateOutputStatus();
2193                          }
2194                      } else {
2195                          console.error('Неизвестный тип элемента при drop:',
      AppState.dragType);
2196                      }
2197                  }
2198              } finally {
2199                  App.cancelPaletteDrag();
```

```
2200              }
2201          }
2202
2203          if (AppState.draggingElement) {
2204              AppState.draggingElement = null;
2205          }
2206
2207          Connections.clearConnectionState();
2208      });
2209
2210      document.addEventListener('keydown', (e) => {
2211          if (e.key === 'Delete' && AppState.selectedElement) {
2212              Elements.deleteElement(AppState.selectedElement);
2213              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
2214          }
2215          if (e.key === 'Escape') {
2216              Elements.deselectAll();
2217              Connections.clearConnectionState();
2218              if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
2219          }
2220      });
2221  },
2222
2223      /**
2224       * Настройка контекстного меню
2225       */
2226      setupContextMenu() {
2227          document.addEventListener('click', (e) => {
2228              const menu = document.getElementById('context-menu');
2229              if (!menu.contains(e.target)) {
2230                  menu.style.display = 'none';
2231              }
2232          });
2233
2234          document.getElementById('ctx-properties').addEventListener('click', () => {
2235              const elemId = document.getElementById('context-menu').dataset.elementId;
2236              document.getElementById('context-menu').style.display = 'none';
2237              const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
2238              if (config?.hasProperties) {
2239                  Modal.showPropertiesModal(elemId);
2240              }
2241          });
2242
2243          document.getElementById('ctx-delete').addEventListener('click', () => {
2244              document.getElementById('context-menu').style.display = 'none';
2245
2246              // Используем новую функцию для удаления всех выделенных
2247              Elements.deleteSelectedElements();
2248
2249              if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
2250                  Outputs.updateOutputStatus();
2251              }
2252          });
2253          document.getElementById('ctx-copy').addEventListener('click', () => {
2254              document.getElementById('context-menu').style.display = 'none';
2255              Elements.copySelectedElements();
2256          });
2257      },
2258
2259      /**
2260       * Клик по рабочей области
2261       */
2262      // app.js
2263      // app.js
2264      setupWorkspaceClick() {
```

```
2265          const container = document.getElementById('workspace-container');
2266
2267          container.addEventListener('click', (e) => {
2268              // Если мы только что закончили тянуть РАМКУ (реальное выделение), не
      сбрасываем
2269              if (AppState.marqueeJustEnded) return;
2270
2271              // Если кликнули ЛЕВОЙ кнопкой мыши НЕ по элементу и НЕ по порту
2272              if (e.button === 0 && !e.target.closest('.element') && !
      e.target.closest('.port')) {
2273                  Elements.deselectAll();
2274              }
2275          });
2276      },
2277      /**
2278       * --- Выделение рамкой и множественное перемещение ---
2279       */
2280      // app.js
2281      setupMultiSelection() {
2282          const container = document.getElementById('workspace-container');
2283          const rectEl = document.getElementById('selection-rect');
2284
2285          container.addEventListener('mousedown', (e) => {
2286              // РАМКА: только ЛЕВАЯ кнопка (0) и клик НЕ по элементу
2287              if (e.button !== 0 || e.target.closest('.element') ||
      e.target.closest('#minimap')) return;
2288
2289              const pos = screenToCanvas(e.clientX, e.clientY);
2290              AppState.multiSelecting = true;
2291              AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
      pos.y, w: 0, h: 0 };
2292
2293              rectEl.style.left = e.clientX + 'px';
2294              rectEl.style.top = e.clientY + 'px';
2295              rectEl.style.width = '0px';
2296              rectEl.style.height = '0px';
2297              rectEl.style.display = 'block';
2298          });
2299
2300          document.addEventListener('mousemove', (e) => {
2301              if (!AppState.multiSelecting) return;
2302
2303              const pos = screenToCanvas(e.clientX, e.clientY);
2304              const sx = AppState.selectionRect.startX;
2305              const sy = AppState.selectionRect.startY;
2306
2307              const x = Math.min(sx, pos.x);
2308              const y = Math.min(sy, pos.y);
2309              const w = Math.abs(pos.x - sx);
2310              const h = Math.abs(pos.y - sy);
2311
2312              // Обновляем визуальную рамку
2313              rectEl.style.left = (x * AppState.viewport.zoom + AppState.viewport.panX)
      + 'px';
2314              rectEl.style.top = (y * AppState.viewport.zoom + AppState.viewport.panY) +
      'px';
2315              rectEl.style.width = (w * AppState.viewport.zoom) + 'px';
2316              rectEl.style.height = (h * AppState.viewport.zoom) + 'px';
2317
2318              // Ищем элементы внутри
2319              const selected = [];
2320              for (const [id, elData] of Object.entries(AppState.elements)) {
2321                  if (!elData || elData.type === 'output-frame') continue;
2322                  if (elData.x >= x && elData.x + elData.width <= x + w &&
2323                      elData.y >= y && elData.y + elData.height <= y + h) {
```

```
                    selected.push(id);
                }
            }

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

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

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

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

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

// js/codegen_graph.js


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

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

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

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

```
2388          }
2389
2390          // 2. НОВОЕ: если это separator — учитываем контекст его входа
2391          if (elem.type === 'separator' && graph.inputs.length > 0) {
2392              const inputGraph = graph.inputs[0].fromGraph;
2393              const inputContext = this.collectAllCond(inputGraph);
2394              if (inputContext) {
2395                  c = c ? Optimizer.And(c, inputContext) : inputContext;
2396              }
2397          }
2398
2399          return c;
2400      },
2401      buildDependencyGraph(elementId) {
2402          const graph = {
2403              nodeId: elementId,
2404              elem: AppState.elements[elementId],
2405              inputs: [],
2406              condInput: null,
2407          };
2408
2409          if (!graph.elem) return null;
2410
2411          const inConns = AppState.connections
2412          .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
2413          .sort((a, b) => {
2414              const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
2415              const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
2416              return ai - bi;
2417          });
2418
2419          inConns.forEach(conn => {
2420          graph.inputs.push({
2421              conn,
2422              fromGraph: this.buildDependencyGraph(conn.fromElement)
2423          });
2424          });
2425
2426          const condConn = AppState.connections.find(c =>
2427              c.toElement === elementId && c.toPort === 'cond-0'
2428          );
2429          if (condConn) {
2430              graph.condInput = {
2431                  conn: condConn,
2432                  fromGraph: this.buildDependencyGraph(condConn.fromElement)
2433              };
2434          }
2435
2436          return graph;
2437      },
2438
2439      /**
2440       * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
2441       */
2442      evalLogic(graph) {
2443          if (!graph) return Optimizer.TrueCond;
2444          const elem = graph.elem;
2445
2446          switch (elem.type) {
2447              case 'if': {
2448                  const left = graph.inputs[0]?.fromGraph;
2449                  const right = graph.inputs[1]?.fromGraph;
2450
2451                  const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
2452                  const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);
```

```
2453
2454                const op = elem.props.operator || '=';
2455                return this.buildIfLogic(leftVal, op, rightVal);
2456            }
2457
2458            case 'and': {
2459                let result = null;
2460                for (const inp of graph.inputs) {
2461                    const inLogic = this.evalLogic(inp.fromGraph);
2462                    result = result ? Optimizer.And(result, inLogic) : inLogic;
2463                }
2464                return result || Optimizer.TrueCond;
2465            }
2466
2467            case 'or': {
2468                let result = null;
2469                for (const inp of graph.inputs) {
2470                    const inLogic = this.evalLogic(inp.fromGraph);
2471                    result = result ? Optimizer.Or(result, inLogic) : inLogic;
2472                }
2473                return result || Optimizer.FalseCond;
2474            }
2475
2476            case 'not': {
2477                const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
2478                return Optimizer.Not(inLogic);
2479            }
2480
2481            case 'separator': {
2482                return this.evalLogic(graph.inputs[0]?.fromGraph);
2483            }
2484
2485            default:
2486                return Optimizer.TrueCond;
2487        }
2488    },
2489
2490    /**
2491     * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
2492     */
2493    evalValue(graph) {
2494        if (!graph) return Optimizer.Const(0);
2495        const elem = graph.elem;
2496
2497        switch (elem.type) {
2498            case 'input-signal':
2499                return Optimizer.Var(elem.props.name || graph.nodeId);
2500
2501            case 'const':
2502                return Optimizer.Const(Number(elem.props.value) || 0);
2503
2504            case 'formula': {
2505                const expr = this.buildFormulaExpr(elem);
2506                return Optimizer.Var(expr);
2507            }
2508
2509            case 'separator':
2510                return this.evalValue(graph.inputs[0]?.fromGraph);
2511
2512            default:
2513                return Optimizer.Const(0);
2514        }
2515    },
2516
2517    // js/codegen_graph.js
```

```
2518
2519      /**
2520       * Рекурсивно собрать полный контекст условий для элемента
2521       * через всю цепочку cond-портов вверх
2522       */
2523    // В codegen_graph.js, в evalFullContext добавь:
2524
2525      evalFullContext(graph) {
2526          if (!graph) return null;
2527
2528          let context = null;
2529          const elem = graph.elem;
2530
2531          console.log(`evalFullContext для ${elem.id} (${elem.type})`);
2532
2533          // 1. Если сам элемент имеет cond-порт — собираем его условие
2534          if (graph.condInput) {
2535              const condConn = graph.condInput.conn;
2536              console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
      {condConn.fromPort}`);
2537
2538              const condLogic = this.evalConditionFromPort(
2539                  graph.condInput.fromGraph,
2540                  condConn.fromPort
2541              );
2542              console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
2543              context = condLogic;
2544
2545              // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
2546              const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
2547              if (upstreamContext) {
2548                  console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
2549                  context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
2550              }
2551          } else {
2552              console.log(`  → нет cond-0`);
2553          }
2554
2555          console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
2556          return context;
2557      },
2558
2559      /**
2560       * Получить УСЛОВИЕ для cond-порта элемента
2561       * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
2562       */
2563      evalConditionFromPort(graph, fromPort) {
2564          if (!graph) return null;
2565          const elem = graph.elem;
2566
2567          // Если это сепаратор — вычисляем его вход и применяем ветвление
2568          if (elem.type === 'separator') {
2569              const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
2570
2571              if (fromPort === 'out-0') {
2572                  return inputLogic;
2573              } else if (fromPort === 'out-1') {
2574                  return Optimizer.Not(inputLogic);
2575              }
2576          }
2577
2578          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
```

```
2579          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
2580              return this.evalLogic(graph);
2581          }
2582
2583          return null;
2584      },
2585
2586      /**
2587       * Главная функция: получить {cond, expr} для элемента
2588       */
2589      evalGraphValue(graph) {
2590
2591          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
2592
2593          const elem = graph.elem;
2594          //let cond = null;
2595
2596          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
2597          let cond = this.collectAllCond(graph);
2598
2599          let expr = null;
2600
2601          switch (elem.type) {
2602              case 'input-signal':
2603                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
2604                  break;
2605
2606              case 'const':
2607                  expr = Optimizer.Const(Number(elem.props.value) || 0);
2608                  break;
2609
2610              case 'formula': {
2611                  // Для формулы также собираем условия от всех входных элементов
2612                  const inputConds = graph.inputs.map(inp => {
2613                      const inResult = this.evalGraphValue(inp.fromGraph);
2614                      return inResult.cond;
2615                  }).filter(c => c);
2616
2617                  // Объединяем cond-порт с условиями от входов
2618                  for (const inCond of inputConds) {
2619                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
2620                  }
2621
2622                  expr = Optimizer.Var(this.buildFormulaExpr(elem));
2623                  break;
2624              }
2625
2626              case 'separator':
2627                  // Сепаратор — просто пробрасываем значение дальше
2628                  return this.evalGraphValue(graph.inputs[0]?.fromGraph);
2629
2630              // Логические элементы не должны здесь быть
2631              case 'and':
2632              case 'or':
2633              case 'not':
2634              case 'if':
2635              default:
2636                  expr = Optimizer.Const(0);
2637          }
2638
2639          return { cond, expr };
2640      },
2641
2642      buildIfLogic(leftVal, op, rightVal) {
```

```
2643          const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
2644          const rightName = rightVal.type === 'var' ? rightVal.name :
      String(rightVal.n);
2645
2646          const leftZero = leftVal.type === 'const' && leftVal.n === 0;
2647          const rightZero = rightVal.type === 'const' && rightVal.n === 0;
2648
2649          switch (op) {
2650              case '=':
2651                  if (rightZero) return Optimizer.Eq0(leftName);
2652                  if (leftZero) return Optimizer.Eq0(rightName);
2653                  return Optimizer.Cmp(leftName, '=', rightName);
2654              case '!=':
2655                  if (rightZero) return Optimizer.Ne0(leftName);
2656                  if (leftZero) return Optimizer.Ne0(rightName);
2657                  return Optimizer.Cmp(leftName, '!=', rightName);
2658              case '>':
2659              case '<':
2660              case '>=':
2661              case '<=':
2662                  return Optimizer.Cmp(leftName, op, rightName);
2663              default:
2664                  return Optimizer.TrueCond;
2665          }
2666      },
2667
2668
2669      buildFormulaExpr(elem) {
2670          let result = elem.props.expression || '0';
2671
2672          // 1) Сначала раскрываем шаблоны (h и др.)
2673          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
2674              ? Settings.getTemplatesMap()
2675              : null;
2676          result = expandFormulaTemplates(result, map);
2677
2678          // 2) Потом раскрываем ссылки на формулы
2679          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
2680          for (const ref of formulaRefs) {
2681              const refElem = AppState.elements[ref];
2682              if (refElem && refElem.type === 'formula') {
2683                  const refExpr = this.buildFormulaExpr(refElem);
2684                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
2685              }
2686          }
2687
2688          return result;
2689      }
2690  };
2691
2692  window.CodeGenGraph = CodeGenGraph;
2693
2694  // js/codegen_optimizer.js
2695
2696  let _depth = 0;
2697  const MAX_DEPTH = 200;
2698
2699  // === Конструкторы ===
2700  function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
2701  function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
2702  function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
2703  function And(a, b) {
2704      if (!a) return b;
2705      if (!b) return a;
2706      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;
```

```
2772      }
2773  }
2774
2775  // Преобразует cmp-условие в интервал по одной переменной
2776  // Возвращает { varName, min, minInc, max, maxInc } или null
2777  function cmpToInterval(c) {
2778      if (!c || c.type !== 'cmp') return null;
2779
2780      const lNum = parseNumberLiteral(c.l);
2781      const rNum = parseNumberLiteral(c.r);
2782
2783      let varName, op, val;
2784
2785      if (lNum == null && rNum != null) {
2786          // var OP const
2787          varName = c.l;
2788          op = c.op;
2789          val = rNum;
2790      } else if (lNum != null && rNum == null) {
2791          // const OP var  ->  var (OP') const
2792          varName = c.r;
2793          op = reverseOp(c.op);
2794          if (!op) return null;
2795          val = lNum;
2796      } else {
2797          // Либо обе стороны числа, либо обе не числа — не трогаем
2798          return null;
2799      }
2800
2801      // Интересуют только упорядочивающие операторы
2802      switch (op) {
2803          case '<':
2804          case '<=':
2805          case '>':
2806          case '>=':
2807          case '=':
2808              break;
2809          default:
2810              return null;
2811      }
2812
2813      let min = Number.NEGATIVE_INFINITY;
2814      let max = Number.POSITIVE_INFINITY;
2815      let minInc = false;
2816      let maxInc = false;
2817
2818      switch (op) {
2819          case '<':
2820              max = val; maxInc = false; break;
2821          case '<=':
2822              max = val; maxInc = true; break;
2823          case '>':
2824              min = val; minInc = false; break;
2825          case '>=':
2826              min = val; minInc = true; break;
2827          case '=':
2828              min = val; minInc = true;
2829              max = val; maxInc = true;
2830              break;
2831      }
2832
2833      return { varName, min, minInc, max, maxInc };
2834  }
2835
2836  function intervalSubset(a, b) {
```

```
2837      if (!a || !b) return false;
2838
2839      // Нижняя граница: a.min >= b.min
2840      const amin = a.min, bmin = b.min;
2841      if (amin === Number.NEGATIVE_INFINITY) {
2842          if (bmin !== Number.NEGATIVE_INFINITY) return false;
2843          // оба -∞ — ок
2844      } else if (bmin === Number.NEGATIVE_INFINITY) {
2845          // b начинается “раньше” — ок
2846      } else if (amin > bmin) {
2847          // a стартует правее b — ок
2848      } else if (amin < bmin) {
2849          // a захватывает меньшее значение — не подмножество
2850          return false;
2851      } else {
2852          // amin === bmin
2853          if (a.minInc && !b.minInc) {
2854              // a включает границу, а b — нет → в a есть точка, не входящая в b
2855              return false;
2856          }
2857      }
2858
2859      // Верхняя граница: a.max <= b.max
2860      const amax = a.max, bmax = b.max;
2861      if (amax === Number.POSITIVE_INFINITY) {
2862          if (bmax !== Number.POSITIVE_INFINITY) return false;
2863      } else if (bmax === Number.POSITIVE_INFINITY) {
2864          // b идёт дальше — ок
2865      } else if (amax < bmax) {
2866          // a заканчивается раньше — ок
2867      } else if (amax > bmax) {
2868          return false;
2869      } else {
2870          // amax === bmax
2871          if (a.maxInc && !b.maxInc) {
2872              return false;
2873          }
2874      }
2875
2876      return true;
2877  }
2878
2879  // Удаляет избыточные cmp-условия в массиве атомов
2880  // mode: 'and' | 'or'
2881  function removeRedundantCmpAtoms(atoms, mode) {
2882      if (!atoms || atoms.length < 2) return atoms;
2883
2884      const keep = new Array(atoms.length).fill(true);
2885
2886      for (let i = 0; i < atoms.length; i++) {
2887          if (!keep[i]) continue;
2888          const a = atoms[i];
2889          if (!a || a.type !== 'cmp') continue;
2890
2891          for (let j = 0; j < atoms.length; j++) {
2892              if (i === j || !keep[j]) continue;
2893              const b = atoms[j];
2894              if (!b || b.type !== 'cmp') continue;
2895
2896              const rel = cmpImplicationRelation(a, b);
2897              if (!rel) continue;
2898
2899              if (rel === 'a_in_b') {
2900                  if (mode === 'or') {
2901                      // A ⊆ B → A OR B = B  → A лишнее
```

```
2902                      keep[i] = false;
2903                      break;
2904                  } else if (mode === 'and') {
2905                      // A ⊆ B → A AND B = A → B лишнее
2906                      keep[j] = false;
2907                  }
2908              } else if (rel === 'b_in_a') {
2909                  if (mode === 'or') {
2910                      // B ⊆ A → A OR B = A → B лишнее
2911                      keep[j] = false;
2912                  } else if (mode === 'and') {
2913                      // B ⊆ A → A AND B = B → A лишнее
2914                      keep[i] = false;
2915                      break;
2916                  }
2917              }
2918          }
2919      }
2920
2921      return atoms.filter((_, idx) => keep[idx]);
2922  }
2923
2924  // Отношение между двумя cmp-условиями через интервалы
2925  // 'a_in_b'  — A ⊆ B
2926  // 'b_in_a'  — B ⊆ A
2927  // 'equal'   — одинаковые интервалы (редко используем)
2928  // null      — не можем определить
2929  function cmpImplicationRelation(c1, c2) {
2930      const i1 = cmpToInterval(c1);
2931      const i2 = cmpToInterval(c2);
2932      if (!i1 || !i2) return null;
2933      if (i1.varName !== i2.varName) return null;
2934
2935      const aInB = intervalSubset(i1, i2);
2936      const bInA = intervalSubset(i2, i1);
2937
2938      if (aInB && bInA) return 'equal';
2939      if (aInB) return 'a_in_b';
2940      if (bInA) return 'b_in_a';
2941      return null;
2942  }
2943
2944  // Разворот оператора при перестановке аргументов (левый/правый)
2945  function reverseOp(op) {
2946      switch (op) {
2947          case '<':  return '>';
2948          case '>':  return '<';
2949          case '<=': return '>=';
2950          case '>=': return '<=';
2951          case '=':
2952          case '!=':
2953              return op;
2954          default:
2955              return null;
2956      }
2957  }
2958
2959  // Аккуратный парсер числового литерала.
2960  // Возвращает число или null, если строка не чисто числовая.
2961  function parseNumberLiteral(s) {
2962      if (typeof s !== 'string') return null;
2963      const trimmed = s.trim().replace(',', '.');
2964
2965      // Только простые вещи: -123, 45, 3.14
2966      if (!/^-?\d+(\.\d+)?$/.test(trimmed)) return null;
```

```
2967
2968      const n = Number(trimmed);
2969      return Number.isFinite(n) ? n : null;
2970  }
2971
2972
2973  function negateAtomKey(key) {
2974      if (!key) return null;
2975      if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
2976      if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
2977      if (key.startsWith('cmp:')) {
2978          const parts = key.slice(4).split(':');
2979          if (parts.length === 3) {
2980              const negOp = negateOp(parts[1]);
2981              if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
2982          }
2983      }
2984      return null;
2985  }
2986
2987  function isNegation(a, b) {
2988      if (!a || !b) return false;
2989      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
2990      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
2991      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
2992          return a.op === negateOp(b.op);
2993      }
2994      if (a.type === 'not' && condEq(a.x, b)) return true;
2995      if (b.type === 'not' && condEq(b.x, a)) return true;
2996      return false;
2997  }
2998
2999  function isAtomCond(t) {
3000      return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
3001  }
3002
3003  function pruneOrByContext(orTerm, contextAtoms) {
3004      const branches = flattenOr(orTerm);
3005      const kept = [];
3006
3007      for (const br of branches) {
3008          let contradicts = false;
3009
3010          for (const ctx of contextAtoms) {
3011              if (isNegation(br, ctx)) {
3012                  contradicts = true;
3013                  break;
3014              }
3015          }
3016
3017          if (!contradicts) kept.push(br);
3018      }
3019
3020      if (kept.length === 0) return FalseCond;
3021      if (kept.length === 1) return kept[0];
3022      return buildOr(kept);
3023  }
3024
3025  function condNegationEq(a, b) {
3026      if (!a || !b) return false;
3027
3028      // Простая проверка: a == NOT(b)
3029      if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;
3030
3031      // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
```

```
3032      // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
3033      if (a.type === 'or' && b.type === 'and') {
3034          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
3035                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3036      }
3037      // Симметрично
3038      if (a.type === 'and' && b.type === 'or') {
3039          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
3040                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
3041      }
3042
3043      // Проверка атомов: (X = 0) vs (X != 0)
3044      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
3045      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
3046
3047      // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
3048      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
3049          return a.op === negateOp(b.op);
3050      }
3051
3052      return false;
3053  }
3054
3055
3056
3057  function condEq(a, b) {
3058      if (a === b) return true;
3059      if (!a || !b) return false;
3060      if (a.type !== b.type) return false;
3061
3062      switch (a.type) {
3063          case 'eq0':
3064          case 'ne0':
3065              return a.v === b.v;
3066          case 'cmp':
3067              return a.l === b.l && a.op === b.op && a.r === b.r;
3068          case 'true':
3069          case 'false':
3070              return true;
3071          case 'not':
3072              return condEq(a.x, b.x);
3073          case 'and':
3074          case 'or':
3075              return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
3076                     (condEq(a.a, b.b) && condEq(a.b, b.a));
3077          default:
3078              return false;
3079      }
3080  }
3081
3082  function flattenAnd(c) {
3083      if (!c) return [];
3084      if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
3085      return [c];
3086  }
3087
3088  function flattenOr(c) {
3089      if (!c) return [];
3090      if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
3091      return [c];
3092  }
3093
3094  function buildAnd(terms) {
3095      if (terms.length === 0) return TrueCond;
3096      let result = terms[0];
```

```
3097      for (let i = 1; i < terms.length; i++) {
3098          result = And(result, terms[i]);
3099      }
3100      return result;
3101  }
3102
3103  function buildOr(terms) {
3104      if (terms.length === 0) return FalseCond;
3105      let result = terms[0];
3106      for (let i = 1; i < terms.length; i++) {
3107          result = Or(result, terms[i]);
3108      }
3109      return result;
3110  }
3111
3112  // Поглощение для AND: X AND (X OR Y) = X
3113  function applyAndAbsorption(terms) {
3114      if (!terms || terms.length < 2) return terms;
3115
3116      const keep = new Array(terms.length).fill(true);
3117
3118      for (let i = 0; i < terms.length; i++) {
3119          if (!keep[i]) continue;
3120          const ti = terms[i];
3121          if (!ti || ti.type !== 'or') continue;
3122
3123          const orParts = flattenOr(ti);
3124          let drop = false;
3125
3126          outer:
3127          for (const part of orParts) {
3128              for (let j = 0; j < terms.length; j++) {
3129                  if (j === i || !keep[j]) continue;
3130                  if (condEq(part, terms[j])) {
3131                      drop = true;
3132                      break outer;
3133                  }
3134              }
3135          }
3136
3137          if (drop) {
3138              keep[i] = false;
3139          }
3140      }
3141
3142      return terms.filter((_, idx) => keep[idx]);
3143  }
3144
3145  // Поглощение для OR: X OR (X AND Y) = X
3146  function applyOrAbsorption(terms) {
3147      if (!terms || terms.length < 2) return terms;
3148
3149      const keep = new Array(terms.length).fill(true);
3150
3151      for (let i = 0; i < terms.length; i++) {
3152          if (!keep[i]) continue;
3153          const ti = terms[i];
3154          if (!ti || ti.type !== 'and') continue;
3155
3156          const andParts = flattenAnd(ti);
3157          let drop = false;
3158
3159          outer:
3160          for (const part of andParts) {
3161              for (let j = 0; j < terms.length; j++) {
```

```
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

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

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

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

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

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

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

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

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

    if (!a) return b;
    if (!b) return a;
```

```
3227      if (a.type === 'false' || b.type === 'false') return FalseCond;
3228      if (a.type === 'true') return b;
3229      if (b.type === 'true') return a;
3230
3231      const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
3232
3233      // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
3234      const eq0Vars = new Map(); // var -> term
3235      const ne0Vars = new Map(); // var -> term
3236      const cmpTerms = [];
3237      const otherTerms = [];
3238
3239      for (const t of allTerms) {
3240          if (t.type === 'true') continue;
3241          if (t.type === 'false') return FalseCond;
3242
3243          if (t.type === 'eq0') {
3244              // Проверка на противоречие сразу
3245              if (ne0Vars.has(t.v)) {
3246                  console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
3247                  return FalseCond;
3248              }
3249              eq0Vars.set(t.v, t);
3250          } else if (t.type === 'ne0') {
3251              // Проверка на противоречие сразу
3252              if (eq0Vars.has(t.v)) {
3253                  console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
3254                  return FalseCond;
3255              }
3256              ne0Vars.set(t.v, t);
3257          } else if (t.type === 'cmp') {
3258              cmpTerms.push(t);
3259          } else if (t.type === 'or') {
3260              // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
3261              const orTerms = flattenOr(t);
3262              const validBranches = [];
3263
3264              for (const branch of orTerms) {
3265                  let branchValid = true;
3266
3267                  if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
3268                      console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
      {branch.v} = 0`);
3269                      branchValid = false;
3270                  } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
3271                      console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
      {branch.v} != 0`);
3272                      branchValid = false;
3273                  }
3274
3275                  if (branchValid) {
3276                      validBranches.push(branch);
3277                  }
3278              }
3279
3280              if (validBranches.length === 0) {
3281                  console.log(`Все ветки OR противоречат контексту → FALSE`);
3282                  return FalseCond;
3283              } else if (validBranches.length === 1) {
3284                  // Если осталась только одна ветка OR, добавляем её напрямую
3285                  const singleBranch = validBranches[0];
3286                  if (singleBranch.type === 'eq0') {
3287                      if (ne0Vars.has(singleBranch.v)) return FalseCond;
3288                      eq0Vars.set(singleBranch.v, singleBranch);
3289                  } else if (singleBranch.type === 'ne0') {
```

```
3290                      if (eq0Vars.has(singleBranch.v)) return FalseCond;
3291                      ne0Vars.set(singleBranch.v, singleBranch);
3292                  } else {
3293                      otherTerms.push(singleBranch);
3294                  }
3295              } else {
3296                  // Перестраиваем OR только с валидными ветками
3297                  otherTerms.push(buildOr(validBranches));
3298              }
3299          } else {
3300              otherTerms.push(t);
3301          }
3302      }
3303
3304      // Собираем уникальные атомы
3305      const atomMap = new Map();
3306
3307      for (const [v, term] of eq0Vars) {
3308          const key = atomKey(term);
3309          if (key) atomMap.set(key, term);
3310      }
3311
3312      for (const [v, term] of ne0Vars) {
3313          const key = atomKey(term);
3314          if (key) atomMap.set(key, term);
3315      }
3316
3317      for (const term of cmpTerms) {
3318          const key = atomKey(term);
3319          if (key) {
3320              const negKey = negateAtomKey(key);
3321              if (negKey && atomMap.has(negKey)) {
3322                  return FalseCond;
3323              }
3324              if (!atomMap.has(key)) {
3325                  atomMap.set(key, term);
3326              }
3327          }
3328      }
3329
3330      let uniqueAtoms = Array.from(atomMap.values());
3331      uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
3332
3333      let result = [...uniqueAtoms, ...otherTerms];
3334
3335      // Поглощение: X AND (X OR Y) = X
3336      // === НОВОЕ: выбрасываем из OR ветки, противоречащие контексту AND ===
3337      const contextAtoms = result.filter(t => isAtomCond(t));
3338      result = result.map(t => {
3339          if (t.type !== 'or') return t;
3340          return pruneOrByContext(t, contextAtoms);
3341      }).filter(t => t.type !== 'true'); // на всякий случай
3342
3343      result = applyAndAbsorption(result);
3344
3345      if (result.length === 0) return TrueCond;
3346      if (result.length === 1) return result[0];
3347
3348      return buildAnd(result);
3349  }
3350
3351          case 'or': {
3352              const a = simplifyCondCore(c.a);
3353              const b = simplifyCondCore(c.b);
3354
```

```
3355              if (!a) return b;
3356              if (!b) return a;
3357              if (a.type === 'true' || b.type === 'true') return TrueCond;
3358              if (a.type === 'false') return b;
3359              if (b.type === 'false') return a;
3360
3361              const allTerms = [...flattenOr(a), ...flattenOr(b)];
3362              const atomMap = new Map();
3363              const otherTerms = [];
3364
3365              for (const t of allTerms) {
3366                  if (t.type === 'true') return TrueCond;
3367                  if (t.type === 'false') continue;
3368
3369                  const key = atomKey(t);
3370                  if (key) {
3371                      const negKey = negateAtomKey(key);
3372                      if (negKey && atomMap.has(negKey)) {
3373                          return TrueCond;
3374                      }
3375                      if (!atomMap.has(key)) {
3376                          atomMap.set(key, t);
3377                      }
3378                  } else {
3379                      otherTerms.push(t);
3380                  }
3381              }
3382
3383              let uniqueAtoms = Array.from(atomMap.values());
3384              uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
3385
3386              let result = [...uniqueAtoms, ...otherTerms];
3387
3388              // Поглощение: X OR (X AND Y) = X
3389              result = applyOrAbsorption(result);
3390
3391              if (result.length === 0) return FalseCond;
3392              if (result.length === 1) return result[0];
3393
3394              return buildOr(result);
3395          }
3396
3397          default:
3398              return c;
3399      }
3400  }
3401
3402  // === Сравнение выражений ===
3403  function exprEq(a, b) {
3404      if (a === b) return true;
3405      if (!a && !b) return true;
3406      if (!a || !b) return false;
3407      if (a.type !== b.type) return false;
3408
3409      switch (a.type) {
3410          case 'const': return a.n === b.n;
3411          case 'var': return a.name === b.name;
3412          case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
3413          case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
3414          default: return false;
3415      }
3416  }
3417
3418  // === Упрощение выражений ===
3419  function simplifyExpr(expr) {
```

```
3420      _depth++;
3421      if (_depth > MAX_DEPTH) {
3422          _depth--;
3423          return expr;
3424      }
3425
3426      try {
3427          return simplifyExprCore(expr);
3428      } finally {
3429          _depth--;
3430      }
3431  }
3432
3433  function simplifyExprCore(expr) {
3434      if (!expr || expr.kind !== 'expr') return expr;
3435
3436      switch (expr.type) {
3437          case 'const':
3438          case 'var':
3439              return expr;
3440
3441          case 'op': {
3442              const l = simplifyExprCore(expr.l);
3443              const r = simplifyExprCore(expr.r);
3444
3445              if (expr.op === '+') {
3446                  if (r?.type === 'const' && r.n === 0) return l;
3447                  if (l?.type === 'const' && l.n === 0) return r;
3448              }
3449              if (expr.op === '*') {
3450                  if (l?.type === 'const' && l.n === 0) return Const(0);
3451                  if (r?.type === 'const' && r.n === 0) return Const(0);
3452                  if (l?.type === 'const' && l.n === 1) return r;
3453                  if (r?.type === 'const' && r.n === 1) return l;
3454              }
3455              return Op(expr.op, l, r);
3456          }
3457
3458          case 'when': {
3459              const c = simplifyCond(expr.c);
3460              const t = simplifyExprCore(expr.t);
3461              const e = simplifyExprCore(expr.e);
3462
3463              if (c?.type === 'true') return t;
3464              if (c?.type === 'false') return e;
3465              if (exprEq(t, e)) return t;
3466              // ✅ НОВОЕ: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
3467              if (e && e.type === 'when') {
3468                  const c2 = simplifyCond(e.c);
3469                  const t2 = simplifyExprCore(e.t);
3470                  const e2 = simplifyExprCore(e.e);
3471
3472                  if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
3473                      return When(c, t, t2);
3474                  }
3475              }
3476              // Узкое правило: WHEN(A∧B, t1, WHEN(A∧¬B, t2, WHEN(¬A, t3, e3))) -> ...
      t3
3477              if (e && e.type === 'when') {
3478                  const c2 = e.c, t2 = e.t, e2 = e.e;
3479
3480                  if (e2 && e2.type === 'when') {
3481                      const c3 = e2.c, t3 = e2.t;
3482
3483                      const shared = findSharedAndComplement(c, c2);
```

```
3484                      if (shared && condNegationEq(c3, shared.shared)) {
3485                          return When(c, t, When(c2, t2, t3));
3486                      }
3487                  }
3488              }
3489
3490              return When(c, t, e);
3491          }
3492
3493          default:
3494              return expr;
3495      }
3496  }
3497
3498  // === Печать ===
3499  function printCond(c) {
3500      if (!c) return 'TRUE';
3501
3502      switch (c.type) {
3503          case 'eq0': return `(${c.v} = 0)`;
3504          case 'ne0': return `(${c.v} != 0)`;
3505          case 'cmp': return `(${c.l} ${c.op} ${c.r})`;
3506          case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
3507          case 'or':  return `(${printCond(c.a)} OR ${printCond(c.b)})`;
3508          case 'not': return `NOT(${printCond(c.x)})`;
3509          case 'true': return 'TRUE';
3510          case 'false': return 'FALSE';
3511          default: return '?';
3512      }
3513  }
3514
3515  function printExpr(e) {
3516      if (!e) return '0';
3517
3518      switch (e.type) {
3519          case 'const': return String(e.n);
3520          case 'var': return e.name;
3521          case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
3522          case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, $
      {printExpr(e.e)})`;
3523          default: return '?';
3524      }
3525  }
3526
3527  window.Optimizer = {
3528      Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
3529      Const, Var, Op, When,
3530      simplifyCond, simplifyExpr,
3531      printCond, printExpr,
3532      condEq, exprEq
3533  };
3534
3535  // js/codegen.js
3536
3537  const CodeGen = {
3538      _cache: {},
3539      _branchCache: {},
3540      _resolveCache: {},
3541      _visiting: new Set(),
3542
3543      reset() {
3544          this._cache = {};
3545          this._branchCache = {};
3546          this._resolveCache = {};
3547          this._visiting = new Set();
```

```
3548      },
3549
3550      toExpr(valueStr) {
3551          const s = String(valueStr).trim();
3552          if (s === '0') return Optimizer.Const(0);
3553          const num = parseFloat(s);
3554          if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
3555          return Optimizer.Var(s);
3556      },
3557
3558      exprToName(exprAst) {
3559          if (!exprAst) return '0';
3560          if (exprAst.type === 'var') return exprAst.name;
3561          if (exprAst.type === 'const') return String(exprAst.n);
3562          return Optimizer.printExpr(exprAst);
3563      },
3564
3565      mergeCond(a, b) {
3566          if (!a && !b) return null;
3567          if (!a) return b;
3568          if (!b) return a;
3569          if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
3570          return Optimizer.And(a, b);
3571      },
3572
3573      getConn(toId, toPort) {
3574          return AppState.connections.find(c => c.toElement === toId && c.toPort ===
      toPort);
3575      },
3576
3577      getConns(toId, prefix) {
3578          return AppState.connections.filter(c => c.toElement === toId &&
      c.toPort.startsWith(prefix));
3579      },
3580
3581      buildFormulaExpr(elem) {
3582          let result = elem.props.expression || '0';
3583
3584          // 1) Сначала раскрываем шаблоны (h и др.)
3585          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
3586              ? Settings.getTemplatesMap()
3587              : null;
3588          result = expandFormulaTemplates(result, map);
3589
3590          // 2) Потом раскрываем ссылки на формулы
3591          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
3592          for (const ref of formulaRefs) {
3593              const refElem = AppState.elements[ref];
3594              if (refElem && refElem.type === 'formula') {
3595                  const refExpr = this.buildFormulaExpr(refElem);
3596                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
3597              }
3598          }
3599
3600          return result;
3601      },
3602
3603      // === Получить ЧИСТУЮ логику элемента ===
3604      getPureLogic(id) {
3605          const cacheKey = `logic:${id}`;
3606          if (cacheKey in this._cache) {
3607              return this._cache[cacheKey];
3608          }
3609
3610          const elem = AppState.elements[id];
```

```
3611          if (!elem) return null;
3612
3613          let logic = null;
3614
3615          switch (elem.type) {
3616              case 'if': {
3617                  const leftConn = this.getConn(id, 'in-0');
3618                  const rightConn = this.getConn(id, 'in-1');
3619
3620                  const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
      Optimizer.Const(0);
3621                  const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
      Optimizer.Const(0);
3622
3623                  const op = (elem.props.operator || '=').trim();
3624                  const leftName = this.exprToName(leftVal);
3625                  const rightName = this.exprToName(rightVal);
3626
3627                  const leftZero = leftVal.type === 'const' && leftVal.n === 0;
3628                  const rightZero = rightVal.type === 'const' && rightVal.n === 0;
3629
3630                  switch (op) {
3631                      case '=':
3632                          if (rightZero) {
3633                              logic = Optimizer.Eq0(leftName);
3634                          } else if (leftZero) {
3635                              logic = Optimizer.Eq0(rightName);
3636                          } else {
3637                              logic = Optimizer.Cmp(leftName, '=', rightName);
3638                          }
3639                          break;
3640                      case '!=':
3641                          if (rightZero) {
3642                              logic = Optimizer.Ne0(leftName);
3643                          } else if (leftZero) {
3644                              logic = Optimizer.Ne0(rightName);
3645                          } else {
3646                              logic = Optimizer.Cmp(leftName, '!=', rightName);
3647                          }
3648                          break;
3649                      case '>':
3650                      case '<':
3651                      case '>=':
3652                      case '<=':
3653                          logic = Optimizer.Cmp(leftName, op, rightName);
3654                          break;
3655                      default:
3656                          logic = Optimizer.TrueCond;
3657                  }
3658                  break;
3659              }
3660
3661              case 'and':
3662              case 'or': {
3663                  const isAnd = elem.type === 'and';
3664                  const count = elem.props.inputCount || 2;
3665                  let result = null;
3666
3667                  for (let i = 0; i < count; i++) {
3668                      const conn = this.getConn(id, `in-${i}`);
3669                      if (!conn) continue;
3670
3671                      const val = this.getPureLogic(conn.fromElement);
3672                      if (!val) continue;
3673
```

```
3674                     if (result === null) {
3675                         result = val;
3676                     } else {
3677                         result = isAnd ? Optimizer.And(result, val) :
      Optimizer.Or(result, val);
3678                     }
3679                 }
3680                 logic = result || Optimizer.FalseCond;
3681                 break;
3682             }
3683 
3684             case 'not': {
3685                 const conn = this.getConn(id, 'in-0');
3686                 const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
3687                 logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
3688                 break;
3689             }
3690 
3691             case 'separator': {
3692                 const conn = this.getConn(id, 'in-0');
3693                 logic = conn ? this.getPureLogic(conn.fromElement) :
      Optimizer.FalseCond;
3694                 break;
3695             }
3696 
3697             default:
3698                 logic = null;
3699         }
3700 
3701         // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
3702         if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
      elem.type === 'not') {
3703             const ctx = this.getConditionFromPort(id);
3704             if (ctx) {
3705                 if (logic) {
3706                     logic = Optimizer.And(ctx, logic);
3707                 } else {
3708                     logic = ctx;
3709                 }
3710             }
3711         }
3712 
3713         this._cache[cacheKey] = logic;
3714         return logic;
3715     },
3716 
3717     // === Получить значение ===
3718     getValue(id) {
3719         const elem = AppState.elements[id];
3720         if (!elem) return Optimizer.Const(0);
3721 
3722         switch (elem.type) {
3723             case 'input-signal':
3724                 // Имя сигнала или id как Var(...)
3725                 return this.toExpr(elem.props.name || id);
3726 
3727             case 'const':
3728                 return Optimizer.Const(Number(elem.props.value) || 0);
3729 
3730             case 'formula': {
3731                 // Используем текст формулы как выражение
3732                 const exprStr = this.buildFormulaExpr(elem) || '0';
3733                 return this.toExpr(exprStr);
3734             }
3735 
```

```
3736                default:
3737                    // На всякий случай — даём символическое имя, а не 0
3738                    if (elem.props && typeof elem.props.name === 'string') {
3739                        return this.toExpr(elem.props.name);
3740                    }
3741                    return this.toExpr(id);
3742            }
3743        },
3744
3745        // === Получить ПОЛНОЕ условие для ветки сепаратора ===
3746        getBranchCondition(sepId, fromPort) {
3747            const cacheKey = `${sepId}:${fromPort}`;
3748            if (cacheKey in this._branchCache) {
3749                return this._branchCache[cacheKey];
3750            }
3751
3752            const sep = AppState.elements[sepId];
3753            if (!sep || sep.type !== 'separator') return null;
3754
3755            const inputLogic = this.getPureLogic(sepId);
3756            const sepContext = this.getConditionFromPort(sepId);
3757
3758            let branchLogic;
3759            if (fromPort === 'out-1') {
3760                branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
3761            } else {
3762                branchLogic = inputLogic || Optimizer.TrueCond;
3763            }
3764
3765            let result;
3766            if (sepContext) {
3767                result = Optimizer.And(sepContext, branchLogic);
3768            } else {
3769                result = branchLogic;
3770            }
3771
3772            this._branchCache[cacheKey] = result;
3773            return result;
3774        },
3775
3776        // === Получить условие от cond-порта ===
3777        getConditionFromPort(id) {
3778            const conn = this.getConn(id, 'cond-0');
3779            if (!conn) return null;
3780
3781            const sourceElem = AppState.elements[conn.fromElement];
3782            if (!sourceElem) return null;
3783
3784            if (sourceElem.type === 'separator') {
3785                return this.getBranchCondition(conn.fromElement, conn.fromPort);
3786            }
3787
3788            return this.getPureLogic(conn.fromElement);
3789        },
3790
3791        // === Основная функция разрешения ===
3792        resolve(id) {
3793            if (id in this._resolveCache) {
3794                return this._resolveCache[id];
3795            }
3796
3797            if (this._visiting.has(id)) {
3798                return null;
3799            }
3800            this._visiting.add(id);
```

```
3801
3802          const elem = AppState.elements[id];
3803          if (!elem) {
3804              this._visiting.delete(id);
3805              return null;
3806          }
3807
3808          let result = null;
3809
3810          try {
3811              switch (elem.type) {
3812                  case 'input-signal':
3813                      result = {
3814                          isValue: true,
3815                          cond: null,
3816                          expr: this.toExpr(elem.props.name || id)
3817                      };
3818                      break;
3819
3820                  case 'const': {
3821                      const cond = this.getConditionFromPort(id);
3822                      result = {
3823                          isValue: true,
3824                          cond: cond,
3825                          expr: Optimizer.Const(Number(elem.props.value) || 0)
3826                      };
3827                      break;
3828                  }
3829
3830                  case 'formula': {
3831                      let cond = this.getConditionFromPort(id);
3832
3833                      const inConns = this.getConns(id, 'in-');
3834                      for (const conn of inConns) {
3835                          const inputNode = this.resolve(conn.fromElement);
3836                          if (inputNode && inputNode.cond) {
3837                              cond = this.mergeCond(cond, inputNode.cond);
3838                          }
3839                      }
3840
3841                      const fullExpr = this.buildFormulaExpr(elem);
3842                      result = {
3843                          isValue: true,
3844                          cond: cond,
3845                          expr: Optimizer.Var(fullExpr)
3846                      };
3847                      break;
3848                  }
3849
3850                  default:
3851                      result = null;
3852              }
3853          } finally {
3854              this._visiting.delete(id);
3855          }
3856
3857          this._resolveCache[id] = result;
3858          return result;
3859      },
3860
3861      generate() {
3862          console.log('=== Генерация кода (граф) ===');
3863          this.reset();
3864
3865          try {
```

```
3866             const outputs = Object.values(AppState.elements).filter(e => e.type ===
     'output');
3867
3868             if (outputs.length === 0) {
3869                 return '/* Нет выходов */';
3870             }
3871
3872             const allVariants = [];
3873
3874             for (const out of outputs) {
3875                 const conns = this.getConns(out.id, 'in-');
3876
3877                 for (const conn of conns) {
3878                     console.log(`\n=== Обработка выхода ${out.id}, вход от $
     {conn.fromElement} ===`);
3879                     const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
3880                     const result = CodeGenGraph.evalGraphValue(graph);
3881                     console.log(`Результат: cond=${Optimizer.printCond(result.cond)},
     expr=${Optimizer.printExpr(result.expr)}`);
3882
3883                     if (!result || !result.expr) continue;
3884
3885                     const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
     null;
3886                     const isZero = result.expr.type === 'const' && result.expr.n ===
     0;
3887
3888                     if (isZero && !cond) continue;
3889
3890                     allVariants.push({
3891                         cond,
3892                         expr: result.expr,
3893                         isZero
3894                     });
3895                 }
3896             }
3897
3898             console.log('Варианты:', allVariants.map(v => ({
3899                 cond: Optimizer.printCond(v.cond),
3900                 expr: Optimizer.printExpr(v.expr)
3901             })));
3902
3903             if (allVariants.length === 0) return '0';
3904
3905             const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
3906             if (valueVariants.length === 0) return '0';
3907
3908             let result = Optimizer.Const(0);
3909
3910             for (let i = valueVariants.length - 1; i >= 0; i--) {
3911                 const v = valueVariants[i];
3912                 if (v.cond) {
3913                     result = Optimizer.When(v.cond, v.expr, result);
3914                 } else {
3915                     result = v.expr;
3916                 }
3917             }
3918
3919             const simplified = Optimizer.simplifyExpr(result);
3920             return Optimizer.printExpr(simplified);
3921
3922         } catch (err) {
3923             console.error('Ошибка:', err);
3924             return `/* Ошибка: ${err.message} */`;
3925         }
```

```
3926      }
3927  };
3928
3929  window.CodeGen = CodeGen;
3930
3931  /**
3932   * Конфигурация приложения
3933   * config.js
3934   */
3935
3936  // Типы сигналов
3937  const SIGNAL_TYPE = {
3938      NUMERIC: 'numeric',   // Числовой сигнал
3939      LOGIC: 'logic',       // Логический (может быть TRUE или FALSE)
3940      TRUE: 'true',         // Явно ИСТИНА
3941      FALSE: 'false',       // Явно ЛОЖЬ
3942      ANY: 'any'            // Любой тип
3943  };
3944
3945  // Типы проекта
3946  const PROJECT_TYPE = {
3947      PARAMETER: 'parameter',
3948      RULE: 'rule'
3949  };
3950
3951  // Конфигурация элементов
3952  const ELEMENT_TYPES = {
3953      'input-signal': {
3954          name: 'Вход',
3955          inputs: 0,
3956          outputs: 1,
3957          outputLabels: ['out'],
3958          outputTypes: [SIGNAL_TYPE.NUMERIC],
3959          color: '#4a90d9',
3960          hasProperties: true,
3961          defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
3962          resizable: true,
3963          minWidth: 150,
3964          minHeight: 50
3965      },
3966      'and': {
3967          name: 'И',
3968          inputs: 2,  // По умолчанию 2, но может быть изменено
3969          outputs: 1,
3970          inputLabels: ['A', 'B'],
3971          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3972          outputLabels: ['результат'],
3973          outputTypes: [SIGNAL_TYPE.LOGIC],
3974          color: '#a855f7',
3975          hasProperties: true,  // ← Теперь есть свойства (для изменения количества
      входов)
3976          resizable: true,
3977          minWidth: 120,
3978          minHeight: 80,
3979          hasConditionPort: true,
3980          conditionPortType: SIGNAL_TYPE.LOGIC,
3981          defaultProps: {
3982              inputCount: 2  // ← Новое свойство
3983          }
3984      },
3985      'or': {
3986          name: 'ИЛИ',
3987          inputs: 2,  // По умолчанию 2
3988          outputs: 1,
3989          inputLabels: ['A', 'B'],
```

```
3990          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3991          outputLabels: ['результат'],
3992          outputTypes: [SIGNAL_TYPE.LOGIC],
3993          color: '#a855f7',
3994          hasProperties: true,  // ← Теперь есть свойства
3995          resizable: true,
3996          minWidth: 120,
3997          minHeight: 80,
3998          hasConditionPort: true,
3999          conditionPortType: SIGNAL_TYPE.LOGIC,
4000          defaultProps: {
4001              inputCount: 2  // ← Новое свойство
4002          }
4003      },
4004      'not': {
4005          name: 'НЕ',
4006          inputs: 1,
4007          outputs: 1,
4008          inputLabels: ['A'],
4009          inputTypes: [SIGNAL_TYPE.LOGIC],
4010          outputLabels: ['¬A'],
4011          outputTypes: [SIGNAL_TYPE.LOGIC],
4012          color: '#a855f7',
4013          hasProperties: true,
4014          resizable: true,
4015          minWidth: 100,
4016          minHeight: 60,
4017          hasConditionPort: true,
4018          conditionPortType: SIGNAL_TYPE.LOGIC
4019      },
4020      'if': {
4021          name: 'ЕСЛИ',
4022          inputs: 2,
4023          outputs: 1,  // ← Только один выход!
4024          inputLabels: ['A', 'B'],
4025          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
4026          outputLabels: ['результат'],  // ← Просто результат
4027          outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
4028          color: '#e94560',
4029          hasProperties: true,
4030          defaultProps: { operator: '=' },
4031          resizable: true,
4032          minWidth: 120,
4033          minHeight: 80,
4034          hasConditionPort: true,
4035          conditionPortType: SIGNAL_TYPE.LOGIC
4036      },
4037      'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
4038          name: 'Сепаратор',
4039          inputs: 1,
4040          outputs: 2,
4041          inputLabels: ['сигнал'],
4042          inputTypes: [SIGNAL_TYPE.LOGIC],
4043          outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
4044          outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
4045          color: '#f59e0b',
4046          hasProperties: true,
4047          resizable: true,
4048          minWidth: 120,
4049          minHeight: 80,
4050          hasConditionPort: true,
4051          conditionPortType: SIGNAL_TYPE.LOGIC
4052      },
4053      'const': {
4054          name: 'Константа',
```

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

const VIEWPORT_CONFIG = {
    minZoom: 0.1,
    maxZoom: 3,
    zoomStep: 0.1,
    panSpeed: 1,
    canvasWidth: 5000,
```

```
4120      canvasHeight: 5000
4121  };
4122
4123  const MINIMAP_CONFIG = {
4124      width: 200,
4125      height: 150,
4126      padding: 10
4127  };
4128
4129  /**
4130   * Модуль работы с соединениями
4131   * connections.js
4132   */
4133
4134  const Connections = {
4135      /**
4136       * Настройка обработчиков порта
4137       */
4138      setupPortHandlers(port) {
4139          port.addEventListener('mousedown', (e) => {
4140              e.stopPropagation();
4141
4142              if (port.classList.contains('output')) {
4143                  const elemId = port.dataset.element;
4144                  const portName = port.dataset.port;
4145                  const signalType = getOutputPortType(elemId, portName);
4146
4147                  AppState.connectingFrom = {
4148                      element: elemId,
4149                      port: portName
4150                  };
4151                  AppState.connectingFromType = signalType;
4152
4153                  this.highlightCompatiblePorts(signalType);
4154
4155                  const svg = document.getElementById('connections-svg');
4156                  const startPos = this._getPortCanvasCenter(port);
4157
4158                  AppState.tempLine = document.createElementNS('http://www.w3.org/2000/
      svg', 'path');
4159                  AppState.tempLine.setAttribute('class', 'temp-connection');
4160                  AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L $
      {startPos.x} ${startPos.y}`);
4161                  svg.appendChild(AppState.tempLine);
4162              }
4163          });
4164
4165          port.addEventListener('mouseup', (e) => {
4166              e.stopPropagation();
4167              e.preventDefault();
4168
4169              if (AppState.connectingFrom && port.classList.contains('input')) {
4170                  const toElement = port.dataset.element;
4171                  const toPortName = port.dataset.port;
4172                  const inputType = getInputPortType(toElement, toPortName);
4173
4174                  if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
4175                      this.clearConnectionState();
4176                      return;
4177                  }
4178
4179                  if (AppState.connectingFrom.element !== toElement) {
4180                      const targetElem = AppState.elements[toElement];
4181                      const allowMultipleInputs = targetElem?.type === 'output';
4182
```

```
                    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');
            }
        });
    },

    /**
     * Подсветка совместимых портов
     */
    highlightCompatiblePorts(signalType) {
        document.querySelectorAll('.port.input').forEach(port => {
            const inputType = getInputPortType(port.dataset.element,
port.dataset.port);

            if (areTypesCompatible(signalType, inputType)) {
                port.classList.add('compatible-highlight');
            } else {
                port.classList.add('incompatible');
            }
        });
    },

    /**
     * Очистка состояния соединения
     */
    clearConnectionState() {
        if (AppState.tempLine) {
```

```
4247              AppState.tempLine.remove();
4248              AppState.tempLine = null;
4249          }
4250          AppState.connectingFrom = null;
4251          AppState.connectingFromType = null;
4252
4253          document.querySelectorAll('.port').forEach(port => {
4254              port.classList.remove('compatible-highlight', 'incompatible');
4255          });
4256      },
4257
4258      /**
4259       * Отрисовка временной линии соединения
4260       */
4261      drawTempConnection(e) {
4262      if (!AppState.tempLine || !AppState.connectingFrom) return;
4263
4264      const fromElem = document.getElementById(AppState.connectingFrom.element);
4265      if (!fromElem) return;
4266
4267      const fromPort = fromElem.querySelector(`[data-port="$
     {AppState.connectingFrom.port}"]`);
4268      if (!fromPort) return;
4269
4270      const startPos = this._getPortCanvasCenter(fromPort);
4271      const endPos = screenToCanvas(e.clientX, e.clientY);
4272
4273      const horizontalDist = Math.abs(endPos.x - startPos.x);
4274      const controlDist = Math.max(horizontalDist * 0.4, 50);
4275
4276      // Тянем всегда от выхода (вектор 1, 0)
4277      const cx1 = startPos.x + controlDist;
4278      const cy1 = startPos.y;
4279
4280      // Вторая точка контроля для плавности за курсором
4281      const cx2 = endPos.x - controlDist;
4282      const cy2 = endPos.y;
4283
4284      AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} $
     {cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
4285      AppState.tempLine.setAttribute('fill', 'none');
4286  },
4287
4288      /**
4289       * Отрисовка всех соединений
4290       */
4291      drawConnections() {
4292      const svg = document.getElementById('connections-svg');
4293
4294      // 1. Очистка старых линий
4295      svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
4296
4297      // 2. Сброс визуального состояния портов
4298      document.querySelectorAll('.port.connected').forEach(port => {
4299          port.classList.remove('connected');
4300      });
4301
4302      // 3. Перебор всех соединений из AppState
4303      AppState.connections.forEach(conn => {
4304          const fromElem = document.getElementById(conn.fromElement);
4305          const toElem = document.getElementById(conn.toElement);
4306
4307          if (!fromElem || !toElem) return;
4308
4309          const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
```

```
4310          const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
4311
4312          if (!fromPort || !toPort) return;
4313
4314          fromPort.classList.add('connected');
4315          toPort.classList.add('connected');
4316
4317          const startPos = this._getPortCanvasCenter(fromPort);
4318          const endPos = this._getPortCanvasCenter(toPort);
4319
4320          if (!startPos || !endPos) return;
4321
4322          // Расстояние для изгиба кривой
4323          const horizontalDist = Math.abs(endPos.x - startPos.x);
4324          const verticalDist = Math.abs(endPos.y - startPos.y);
4325          const controlDist = Math.max(horizontalDist * 0.4, 50);
4326
4327          // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
4328          let d;
4329          let cx1 = startPos.x;
4330          let cy1 = startPos.y;
4331          let cx2 = endPos.x;
4332          let cy2 = endPos.y;
4333
4334          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
4335          cx1 = startPos.x + controlDist;
4336          cy1 = startPos.y;
4337
4338          // ВХОД (Target):
4339          if (conn.toPort === 'cond-0') {
4340              // Технический порт: Касательная (0, 1) в декартовой (вверх)
4341              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
4342              cx2 = endPos.x;
4343              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
4344          } else {
4345              // Обычный вход: Касательная (-1, 0) -> Слева направо
4346              cx2 = endPos.x - controlDist;
4347              cy2 = endPos.y;
4348          }
4349
4350          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
     ${endPos.y}`;
4351
4352          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
4353          path.setAttribute('d', d);
4354          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
4355
4356          // --- ЛОГИКА ЦВЕТА (Классы) ---
4357          let cssClass = 'connection';
4358          const type = conn.signalType;
4359
4360          // Приоритет новым типам сигналов
4361          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
4362          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
4363          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
4364          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
4365          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
4366
4367          path.setAttribute('class', cssClass);
4368
4369          // Обработчики событий
4370          path.style.pointerEvents = 'stroke';
4371          path.style.cursor = 'pointer';
4372          path.addEventListener('click', () => this.handleConnectionClick(conn));
4373
```

```
4374          svg.appendChild(path);
4375      });
4376
4377      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
4378          Outputs.updateOutputStatus();
4379      }
4380      Viewport.updateMinimap();
4381  },
4382      /**
4383       * Обработка клика по соединению (удаление)
4384       */
4385      handleConnectionClick(conn) {
4386          if (confirm('Удалить соединение?')) {
4387              AppState.connections = AppState.connections.filter(c =>
4388                  !(c.fromElement === conn.fromElement &&
4389                    c.fromPort === conn.fromPort &&
4390                    c.toElement === conn.toElement &&
4391                    c.toPort === conn.toPort)
4392              );
4393
4394              this.drawConnections();
4395          }
4396      },
4397
4398      /**
4399       * Получение центра порта в координатах Canvas
4400       */
4401      _getPortCanvasCenter(portEl) {
4402          if (!portEl) return null;
4403
4404          const rect = portEl.getBoundingClientRect();
4405          return screenToCanvas(
4406              rect.left + rect.width / 2,
4407              rect.top + rect.height / 2
4408          );
4409      }
4410  };
4411
4412  /**
4413   * Модуль работы с элементами схемы
4414   * elements.js
4415   */
4416
4417  const Elements = {
4418      /**
4419       * Генерация HTML для элемента
4420       */
4421          createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
4422              const config = ELEMENT_TYPES[elemType];
4423              if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
4424
4425              const safe = (value, fallback = '') => (value === null || value ===
      undefined) ? fallback : String(value);
4426              const w = width ?? config.minWidth ?? 120;
4427              const h = height ?? config.minHeight ?? 60;
4428
4429              const getPortClass = (signalType, direction) => {
4430                  const base = direction === 'output' ? 'port output' : 'port input';
4431                  if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
4432                  if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
4433                  return `${base} any-port`;
4434              };
4435
4436              // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
      innerHTML
```

```
4437            // Она тут остается, но ее результат не встраивается в HTML-строку
    напрямую, кроме формулы
4438            const buildConditionPortHTML = () => {
4439                return `
4440                    <div class="condition-port-wrapper">
4441                        <div class="condition-port-label">условие</div>
4442                        <div class="port input condition-port"
4443                            data-port="cond-0"
4444                            data-element="${elemId}"
4445                            data-signal-type="${SIGNAL_TYPE.LOGIC}"
4446                            title="Техническое условие">
4447                        </div>
4448                    </div>`;
4449            };
4450
4451
4452            const buildInputPorts = (count, types = [], labels = []) => {
4453                let html = '';
4454                for (let i = 0; i < count; i++) {
4455                    const type = types[i] ?? types[types.length - 1] ??
    SIGNAL_TYPE.ANY;
4456                    html += `<div class="${getPortClass(type, 'input')}" data-
    port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
    || `Вход ${i+1}`}"></div>`;
4457                }
4458                return html;
4459            };
4460
4461            const buildOutputPorts = (count, types = [], labels = []) => {
4462                let html = '';
4463                for (let i = 0; i < count; i++) {
4464                    const type = types[i] ?? types[types.length - 1] ??
    SIGNAL_TYPE.ANY;
4465                    html += `<div class="${getPortClass(type, 'output')}" data-
    port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
    || `Выход ${i+1}`}"></div>`;
4466                }
4467                return html;
4468            };
4469
4470            const resizeHandles = config.resizable ? `<div class="resize-handle
    handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
    direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
    '';
4471            // hasCondClass будет добавляться в addElement
4472            // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
    '';
4473
4474            let innerHTML = '';
4475
4476            if (elemType === 'input-signal') {
4477                const name = safe(props.name, 'Сигнал');
4478                const type = props.signalType || SIGNAL_TYPE.NUMBER;
4479                const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
4480                innerHTML = `
4481                    <div class="element-header" style="background:$
    {config.color};">Источник</div>
4482                    <div class="element-body">
4483                        <div class="element-symbol">
4484                            <span class="input-signal-icon">${symbol}</span>
4485                            <span class="input-signal-name">${name}</span>
4486                        </div>
4487                        <div class="ports-right">
4488                            ${buildOutputPorts(1, [type], ['Выход'])}
4489                        </div>
```

```
                    </div>`;
            }
            else if (elemType === 'const') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Константа</div>
                    <div class="element-body">
                        <div class="element-symbol">${props.value ?? 0}</div>
                        <div class="ports-right">
                            ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'separator') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Сепаратор</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(1, config.inputTypes, config.inputLabels)}</div>
                        <div class="element-symbol">✓/✗</div>
                        <div class="ports-right">
                            <div class="port output logic-port true-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}" title="ИСТИНА"></div>
                            <div class="port output logic-port false-port" data-port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}" title="ЛОЖЬ"></div>
                        </div>
                    </div>`;
            }
            else if (elemType === 'and' || elemType === 'or') {
                const gateSymbol = elemType === 'and' ? '∧' : '∨';
                const inputCount = props.inputCount || config.defaultProps?.inputCount || 2;

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

                innerHTML = `
                    <div class="element-header" style="background:${config.color};">${config.name}</div>
                    <div class="element-body">
                        <div class="ports-left">
                            ${inputsHTML}
                        </div>
                        <div class="element-symbol">${gateSymbol}</div>
                        <div class="ports-right">
                            <div class="port output logic-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></div>
                        </div>
                    </div>`;
            }
            else if (elemType === 'if') {
                const op = safe(props.operator, '=');
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Условие</div>
                    <div class="element-body">
```

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

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

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

```
4596
4597             else { // Для любых других (fallback)
4598                 innerHTML = `
4599                     <div class="element-header" style="background:${config.color};">${config.name}</div>
4600                     <div class="element-body">
4601                         <div class="ports-left">${buildInputPorts(config.inputs || 0, config.inputTypes, config.inputLabels)}</div>
4602                         <div class="element-symbol">${config.name}</div>
4603                         <div class="ports-right">
4604                             ${buildOutputPorts(config.outputs || 0, config.outputTypes, config.outputLabels)}
4605                         </div>
4606                     </div>`;
4607             }
4608             const commentHtml = `<div class="element-comment">${safe(props.comment, '')}</div>`;
4609
4610
4611             const html = `
4612                 <div class="element ${elemType}" id="${elemId}"
4613                     style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;" data-type="${elemType}">
4614                     ${innerHTML}
4615                     ${commentHtml}
4616                     ${resizeHandles}
4617                 </div>`;
4618
4619             return { html, width: w, height: h };
4620         },
4621
4622     /**
4623      * Добавление элемента
4624      */
4625         addElement(elemType, x, y, props = {}, elemId = null, customWidth = null, customHeight = null) {
4626             const config = ELEMENT_TYPES[elemType];
4627             if (!config) {
4628                 console.error(`Неизвестный тип элемента: ${elemType}`);
4629                 return null;
4630             }
4631
4632             if (!elemId) {
4633                 elemId = `${elemType}_${++AppState.elementCounter}`;
4634             }
4635
4636             let width = customWidth;
4637             let height = customHeight;
4638
4639             if (width === null || width === undefined) {
4640                 width = config.minWidth || 140;
4641             }
4642             if (height === null || height === undefined) {
4643                 height = config.minHeight || 70;
4644             }
4645
4646             try {
4647                 const result = this.createElementHTML(elemType, elemId, x, y, props, width, height);
4648                 if (!result || !result.html) {
4649                     console.error('createElementHTML вернул пустой результат');
4650                     return null;
4651                 }
4652
4653                 const workspace = document.getElementById('workspace');
```

```
4654                 const wrapper = document.createElement('div');
4655                 wrapper.innerHTML = result.html.trim();
4656                 const element = wrapper.firstElementChild;
4657                 if (!element) {
4658                     console.error('Не удалось создать DOM элемент из HTML');
4659                     return null;
4660                 }
4661 
4662                 // Добавляем класс для отступа
4663                 if (config.hasConditionPort) {
4664                     element.classList.add('has-condition-port');
4665                 }
4666 
4667                 workspace.appendChild(element);
4668 
4669                 AppState.elements[elemId] = {
4670                     id: elemId,
4671                     type: elemType,
4672                     x,
4673                     y,
4674                     width: result.width || width,
4675                     height: result.height || height,
4676                     props: { ...(config.defaultProps || {}), ...(props || {}) }
4677                 };
4678 
4679                 // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ ЕГО В HTML)
4680                 if (config.hasConditionPort && elemType !== 'formula') {
4681                     const condPortWrapper = document.createElement('div');
4682                     condPortWrapper.innerHTML = `
4683                         <div class="condition-port-wrapper">
4684                             <div class="condition-port-label">условие</div>
4685                             <div class="port input condition-port"
4686                                 data-port="cond-0"
4687                                 data-element="${elemId}"
4688                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"
4689                                 title="Техническое условие">
4690                             </div>
4691                         </div>`;
4692                     element.prepend(condPortWrapper.firstElementChild); // Вставляем в самое начало элемента
4693                 }
4694 
4695 
4696                 this.setupElementHandlers(elemId); // Передаем ID элемента
4697 
4698                 // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
4699                 // element.querySelectorAll('.port').forEach(port => {
4700                 //     Connections.setupPortHandlers(port);
4701                 // });
4702 
4703                 Connections.drawConnections(); // Перерисовываем соединения, чтобы учесть новые порты
4704                 Viewport.updateMinimap();
4705                 return elemId;
4706             } catch (err) {
4707                 console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
4708                 return null;
4709             }
4710         },
4711 
4712     /**
4713      * Обновление входов логического элемента (AND, OR)
4714      */
4715     updateLogicGateInputs(elemId, inputCount) {
```

```
4716          const elem = document.getElementById(elemId);
4717          if (!elem) return;
4718
4719          const portsLeft = elem.querySelector('.ports-left');
4720          if (!portsLeft) return;
4721
4722          // Удаляем соединения к портам, которые больше не существуют
4723          AppState.connections = AppState.connections.filter(c => {
4724              if (c.toElement === elemId && c.toPort.startsWith('in-')) {
4725                  const portNum = parseInt(c.toPort.split('-')[1], 10);
4726                  return portNum < inputCount;
4727              }
4728              return true;
4729          });
4730
4731          // Генерируем новые входы
4732          let inputsHTML = '';
4733          for (let i = 0; i < inputCount; i++) {
4734              inputsHTML += `
4735                  <div class="port input logic-port"
4736                      data-port="in-${i}"
4737                      data-element="${elemId}"
4738                      data-signal-type="${SIGNAL_TYPE.LOGIC}"
4739                      title="Вход ${i+1}">
4740                  </div>
4741              `;
4742          }
4743          portsLeft.innerHTML = inputsHTML;
4744
4745          // Переподключаем обработчики
4746          portsLeft.querySelectorAll('.port').forEach(port =>
4747              Connections.setupPortHandlers(port)
4748          );
4749
4750          Connections.drawConnections();
4751      },
4752
4753      /**
4754       * Удаление элемента
4755       */
4756      deleteElement(elemId) {
4757          AppState.connections = AppState.connections.filter(c =>
4758              c.fromElement !== elemId && c.toElement !== elemId
4759          );
4760
4761          const elem = document.getElementById(elemId);
4762          if (elem) elem.remove();
4763
4764          delete AppState.elements[elemId];
4765
4766          if (AppState.selectedElement === elemId) {
4767              AppState.selectedElement = null;
4768          }
4769
4770          Connections.drawConnections();
4771          Viewport.updateMinimap();
4772      },
4773
4774      /**
4775       * Выделение элемента
4776       */
4777  // elements.js
4778      selectElement(elemId) {
4779          // Снимаем старое выделение со всех
4780          this.deselectAll();
```

```
4781
4782          AppState.selectedElement = elemId;
4783          AppState.selectedElements = [elemId];
4784
4785          const elem = document.getElementById(elemId);
4786          if (elem) elem.classList.add('selected');
4787
4788          const elemData = AppState.elements[elemId];
4789          if (elemData) {
4790              document.getElementById('selection-info').textContent =
4791                  `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
4792          }
4793      },
4794
4795      deselectAll() {
4796          // Снимаем класс со всех элементов на странице
4797          document.querySelectorAll('.element.selected').forEach(el =>
      el.classList.remove('selected'));
4798
4799          AppState.selectedElement = null;
4800          AppState.selectedElements = [];
4801          if (document.getElementById('selection-info')) {
4802              document.getElementById('selection-info').textContent = '';
4803          }
4804      },
4805      /**
4806       * Настройка обработчиков элемента
4807       */
4808      setupElementHandlers(elemId) {
4809          try {
4810              const elem = document.getElementById(elemId);
4811              if (!elem) return;
4812
4813              // elements.js -> setupElementHandlers
4814              elem.addEventListener('mousedown', (e) => {
4815                  if (e.target.classList.contains('port')) return;
4816                  if (e.target.classList.contains('resize-handle')) return;
4817
4818                  e.preventDefault();
4819                  e.stopPropagation();
4820
4821                  // ПРАВКА ТУТ:
4822                  // Если элемент НЕ в группе — выделяем только его.
4823                  // Если элемент УЖЕ в группе — не трогаем группу, чтобы можно было
      тянуть всех.
4824                  if (!AppState.selectedElements.includes(elemId)) {
4825                      this.selectElement(elemId);
4826                  }
4827
4828                  AppState.draggingElement = elemId;
4829                  const canvasPos = screenToCanvas(e.clientX, e.clientY);
4830                  const elemData = AppState.elements[elemId];
4831                  AppState.dragOffset.x = canvasPos.x - elemData.x;
4832                  AppState.dragOffset.y = canvasPos.y - elemData.y;
4833              });
4834
4835              elem.addEventListener('dblclick', (e) => {
4836                  if (e.target.classList.contains('port')) return;
4837                  const config = ELEMENT_TYPES[AppState.elements[elemId].type];
4838                  if (config?.hasProperties) {
4839                      Modal.showPropertiesModal(elemId);
4840                  }
4841              });
4842
4843              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;
```

```
4908
4909          const offsetX = 50;
4910          const offsetY = 50;
4911
4912          const idMap = {};
4913          const newIds = [];
4914
4915          originals.forEach(el => {
4916              // Копируем свойства элемента (глубокое копирование props)
4917              const newProps = JSON.parse(JSON.stringify(el.props || {}));
4918
4919              // Используем существующую функцию addElement
4920              // Она сама сгенерирует ID и создаст DOM
4921              const createdId = this.addElement(
4922                  el.type,                    // тип элемента
4923                  el.x + offsetX,             // новая позиция X
4924                  el.y + offsetY,             // новая позиция Y
4925                  newProps,                   // скопированные свойства
4926                  null,                       // ID = null, чтобы addElement
      сгенерировал сам
4927                  el.width,                   // ширина
4928                  el.height                   // высота
4929              );
4930
4931              if (createdId) {
4932                  idMap[el.id] = createdId;
4933                  newIds.push(createdId);
4934              }
4935          });
4936
4937          // Копируем связи ТОЛЬКО между скопированными элементами
4938          const newConnections = [];
4939          AppState.connections.forEach(conn => {
4940              if (idMap[conn.fromElement] && idMap[conn.toElement]) {
4941                  newConnections.push({
4942                      fromElement: idMap[conn.fromElement],
4943                      fromPort: conn.fromPort,
4944                      toElement: idMap[conn.toElement],
4945                      toPort: conn.toPort,
4946                      signalType: conn.signalType || 'Boolean'
4947                  });
4948              }
4949          });
4950
4951          AppState.connections.push(...newConnections);
4952          Connections.drawConnections();
4953
4954          // Выделяем новые элементы
4955          this.deselectAll();
4956          AppState.selectedElements = newIds;
4957          AppState.selectedElement = newIds[newIds.length - 1];
4958
4959          newIds.forEach(id => {
4960              const el = document.getElementById(id);
4961              if (el) el.classList.add('selected');
4962          });
4963
4964          document.getElementById('selection-info').textContent =
4965              `Скопировано: ${newIds.length} элемент(ов)`;
4966
4967          Viewport.updateMinimap();
4968          console.log(`Скопировано ${newIds.length} элементов`);
4969      },
4970
4971          // elements.js — добавь в объект Elements
```

```
4972      deleteSelectedElements() {
4973          const ids = (AppState.selectedElements && AppState.selectedElements.length >
      0)
4974              ? [...AppState.selectedElements]
4975              : (AppState.selectedElement ? [AppState.selectedElement] : []);
4976
4977          if (ids.length === 0) {
4978              console.log('Нечего удалять');
4979              return;
4980          }
4981
4982          // Удаляем каждый элемент
4983          ids.forEach(id => {
4984              this.deleteElement(id);
4985          });
4986
4987          // Сбрасываем выделение
4988          AppState.selectedElement = null;
4989          AppState.selectedElements = [];
4990          document.getElementById('selection-info').textContent = '';
4991
4992          console.log(`Удалено ${ids.length} элементов`);
4993      },
4994
4995      /**
4996       * Обработка resize
4997       */
4998      handleResize(e) {
4999          if (!AppState.resizing) return;
5000
5001          const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
      startTop } = AppState.resizing;
5002          const elem = document.getElementById(elemId);
5003          const elemData = AppState.elements[elemId];
5004          const config = ELEMENT_TYPES[elemData.type];
5005
5006          const dx = (e.clientX - startX) / AppState.viewport.zoom;
5007          const dy = (e.clientY - startY) / AppState.viewport.zoom;
5008
5009          let newWidth = startWidth;
5010          let newHeight = startHeight;
5011          let newLeft = startLeft;
5012          let newTop = startTop;
5013
5014          if (handle.includes('e')) {
5015              newWidth = Math.max(config.minWidth, startWidth + dx);
5016          }
5017          if (handle.includes('w')) {
5018              newWidth = Math.max(config.minWidth, startWidth - dx);
5019              newLeft = startLeft + (startWidth - newWidth);
5020          }
5021          if (handle.includes('s')) {
5022              newHeight = Math.max(config.minHeight, startHeight + dy);
5023          }
5024          if (handle.includes('n')) {
5025              newHeight = Math.max(config.minHeight, startHeight - dy);
5026              newTop = startTop + (startHeight - newHeight);
5027          }
5028
5029          elem.style.width = `${newWidth}px`;
5030          elem.style.height = `${newHeight}px`;
5031          elem.style.left = `${newLeft}px`;
5032          elem.style.top = `${newTop}px`;
5033
5034          elemData.width = newWidth;
```

```
5035          elemData.height = newHeight;
5036          elemData.x = newLeft;
5037          elemData.y = newTop;
5038
5039          Connections.drawConnections();
5040      },
5041
5042      /**
5043       * Обработка перетаскивания элемента
5044       */
5045      handleDrag(e) {
5046          if (!AppState.draggingElement) return;
5047
5048          const canvasPos = screenToCanvas(e.clientX, e.clientY);
5049          const elemId = AppState.draggingElement;
5050          const elemData = AppState.elements[elemId];
5051          if (!elemData) return;
5052
5053          const newX = canvasPos.x - AppState.dragOffset.x;
5054          const newY = canvasPos.y - AppState.dragOffset.y;
5055          const dx = newX - elemData.x;
5056          const dy = newY - elemData.y;
5057
5058          // если выделено несколько
5059          const group = AppState.selectedElements && AppState.selectedElements.length >
      1
5060              ? AppState.selectedElements
5061              : [elemId];
5062
5063          for (const id of group) {
5064              const elData = AppState.elements[id];
5065              if (!elData) continue;
5066              elData.x += dx;
5067              elData.y += dy;
5068              const el = document.getElementById(id);
5069              if (el) {
5070              el.style.left = elData.x + 'px';
5071              el.style.top = elData.y + 'px';
5072              }
5073          }
5074
5075          Connections.drawConnections();
5076      },
5077
5078      /**
5079       * Обновление входов формулы
5080       */
5081      updateFormulaInputs(elemId, inputCount) {
5082          const elem = document.getElementById(elemId);
5083          if (!elem) return;
5084
5085          const portsLeft = elem.querySelector('.ports-left');
5086          if (!portsLeft) return;
5087
5088          AppState.connections = AppState.connections.filter(c => {
5089              if (c.toElement === elemId && c.toPort.startsWith('in-')) {
5090                  const portNum = parseInt(c.toPort.split('-')[1], 10);
5091                  return portNum < inputCount;
5092              }
5093              return true;
5094          });
5095
5096          let inputsHTML = '';
5097          for (let i = 0; i < inputCount; i++) {
5098              inputsHTML += `
```

```
5099                 <div class="port input any-port"
5100                      data-port="in-${i}"
5101                      data-element="${elemId}"
5102                      data-signal-type="${SIGNAL_TYPE.ANY}"
5103                      title="in${i} (Любой)">
5104                 </div>
5105             `;
5106         }
5107         portsLeft.innerHTML = inputsHTML;
5108 
5109         portsLeft.querySelectorAll('.port').forEach(port =>
5110             Connections.setupPortHandlers(port)
5111         );
5112 
5113         Connections.drawConnections();
5114     },
5115 
5116     /**
5117      * Рассчитать оптимальный размер элемента на основе количества портов
5118      */
5119     calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
5120         const elem = AppState.elements[elemId];
5121         if (!elem) return null;
5122 
5123         const config = ELEMENT_TYPES[elem.type];
5124         if (!config || !config.resizable) return null;
5125 
5126         // Базовая высота
5127         let baseHeight = config.minHeight || 60;
5128 
5129         // Каждый порт требует примерно 25-30px высоты
5130         const portSpacing = 28;
5131         const maxPorts = Math.max(inputCount, outputCount);
5132 
5133         // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
5134         const additionalHeight = (maxPorts - 1) * portSpacing;
5135         const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
5136 
5137         return newHeight;
5138     },
5139 
5140     /**
5141      * Обновление размера элемента при изменении портов
5142      */
5143     updateElementSize(elemId) {
5144         const elem = document.getElementById(elemId);
5145         const elemData = AppState.elements[elemId];
5146 
5147         if (!elem || !elemData) return;
5148 
5149         const config = ELEMENT_TYPES[elemData.type];
5150         if (!config || !config.resizable) return;
5151 
5152         let inputCount = 0;
5153         let outputCount = config.outputs || 1;
5154 
5155         // Определяем количество входов
5156         if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
      'formula') {
5157             inputCount = elemData.props.inputCount || config.inputs || 2;
5158         } else {
5159             inputCount = config.inputs || 0;
5160         }
5161 
5162         // Рассчитываем новую высоту
```

```
5163          const newHeight = this.calculateOptimalHeight(elemId, inputCount,
      outputCount);
5164
5165          if (newHeight && newHeight !== elemData.height) {
5166              elemData.height = newHeight;
5167              elem.style.height = `${newHeight}px`;
5168
5169              // Перерисовываем соединения, т.к. изменился размер элемента
5170              Connections.drawConnections();
5171              Viewport.updateMinimap();
5172          }
5173      }
5174
5175
5176  };
5177
5178  /**
5179   * Модуль модальных окон
5180   * modal.js
5181   */
5182
5183  const Modal = {
5184      /**
5185       * Инициализация модальных окон
5186       */
5187      init() {
5188          // Модальное окно свойств элемента
5189          document.getElementById('modal-save').addEventListener('click', () => {
5190              this.saveElementProperties();
5191          });
5192
5193          document.getElementById('modal-cancel').addEventListener('click', () => {
5194              this.hideModal('modal-overlay');
5195          });
5196
5197          document.getElementById('modal-overlay').addEventListener('click', (e) => {
5198              if (e.target.id === 'modal-overlay') {
5199                  this.hideModal('modal-overlay');
5200              }
5201          });
5202
5203          // Модальное окно свойств проекта
5204          document.getElementById('project-modal-save').addEventListener('click', () =>
      {
5205              this.saveProjectProperties();
5206          });
5207
5208          document.getElementById('project-modal-cancel').addEventListener('click', ()
      => {
5209              this.hideModal('project-modal-overlay');
5210          });
5211
5212          document.getElementById('project-modal-overlay').addEventListener('click', (e)
      => {
5213              if (e.target.id === 'project-modal-overlay') {
5214                  this.hideModal('project-modal-overlay');
5215              }
5216          });
5217      },
5218
5219      /**
5220       * Показать модальное окно
5221       */
5222      showModal(modalId) {
5223          document.getElementById(modalId).style.display = 'flex';
```

```
5224      },
5225
5226      /**
5227       * Скрыть модальное окно
5228       */
5229      hideModal(modalId) {
5230          document.getElementById(modalId).style.display = 'none';
5231          // Скрываем tooltip если он есть
5232          const tooltip = document.getElementById('template-tooltip');
5233          if (tooltip) {
5234              tooltip.classList.remove('visible');
5235          }
5236      },
5237
5238      /**
5239       * Показать свойства элемента
5240       */
5241      showPropertiesModal(elemId) {
5242          const elemData = AppState.elements[elemId];
5243          const elemType = elemData.type;
5244          const props = elemData.props;
5245          const config = ELEMENT_TYPES[elemType];
5246
5247          const modalOverlay = document.getElementById('modal-overlay');
5248          const modalTitle = document.getElementById('modal-title');
5249          const modalContent = document.getElementById('modal-content');
5250
5251          modalTitle.textContent = `Свойства: ${config.name}`;
5252
5253          let contentHTML = '';
5254
5255   if (elemType === 'input-signal') {
5256    const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
5257
5258    contentHTML = `
5259      <div class="modal-row">
5260        <label>Название сигнала:</label>
5261        <input type="text" id="prop-name" value="${props.name || ''}"
      placeholder="Например: 10LBA..." />
5262        <small style="color:#999;">
5263          Поиск по маске через * (например: *MAA*CP*)
5264        </small>
5265        <div id="signal-filter-results"
5266             style="max-height:160px; overflow-y:auto; background:#0f3460; border-
      radius:5px; margin-top:6px; display:none;">
5267        </div>
5268      </div>
5269
5270      <div class="modal-row">
5271        <label>Описание сигнала:</label>
5272        <textarea id="prop-description" readonly>${props.description || ''}</textarea>
5273      </div>
5274
5275      // modal.js в блоке input-signal
5276      <div class="modal-row">
5277          <label>Размерность:</label>
5278          <input type="text" id="prop-dimension" value="${props.dimension || ''}" />
5279      </div>
5280
5281      <div class="modal-row">
5282        <label>Тип сигнала:</label>
5283        <select id="prop-signal-type">
5284          <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
      'selected' : ''}>Числовой</option>
5285          <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
```

```
      'selected' : ''}>Логический</option>
5286        </select>
5287      </div>
5288    `;
5289
5290    // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
      DOM.
5291    // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
5292    // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
5293  } else if (elemType === 'if') {
5294              contentHTML = `
5295                  <div class="modal-row">
5296                      <label>Оператор сравнения:</label>
5297                      <select id="prop-operator">
5298                          <option value="=" ${props.operator === '=' ? 'selected' : ''}
      >=  (равно)</option>
5299                          <option value=">" ${props.operator === '>' ? 'selected' : ''}
      >>  (больше)</option>
5300                          <option value="<" ${props.operator === '<' ? 'selected' : ''}
      ><  (меньше)</option>
5301                          <option value=">=" ${props.operator === '>=' ? 'selected' :
      ''}>= (больше или равно)</option>
5302                          <option value="<=" ${props.operator === '<=' ? 'selected' :
      ''}>= (меньше или равно)</option>
5303                          <option value="!=" ${props.operator === '!=' ? 'selected' :
      ''}>!= (не равно)</option>
5304                      </select>
5305                  </div>
5306              `;
5307          } else if (elemType === 'and' || elemType === 'or') {
5308              contentHTML = `
5309                  <div class="modal-row">
5310                      <label>Количество входов:</label>
5311                      <input type="number" id="prop-input-count" value="$
      {props.inputCount || 2}" min="2" max="10">
5312                  </div>
5313                  <div class="modal-row">
5314                      <p style="color: #aaa; font-size: 12px;">
5315                          Измените количество входных портов для этого логического
      элемента.
5316                          Лишние соединения будут автоматически удалены.
5317                      </p>
5318                  </div>
5319              `;
5320          } else if (elemType === 'const') {
5321              contentHTML = `
5322                  <div class="modal-row">
5323                      <label>Значение:</label>
5324                      <input type="number" id="prop-value" value="${props.value ?? 0}"
      step="any">
5325                  </div>
5326              `;
5327          }
5328          else if (elemType === 'group') {
5329              contentHTML = `
5330              <div class="modal-row">
5331                  <label>Название группы:</label>
5332                  <input type="text" id="prop-title" value="${props.title || 'Группа'}">
5333              </div>`;
5334          }
5335
5336          else if (elemType === 'formula') {
5337              let signalsHTML = '';
5338              AppState.connections.forEach(conn => {
5339                  if (conn.toElement === elemId) {
```

```
                    const fromElem = AppState.elements[conn.fromElement];
                    if (fromElem) {
                        const signalName = fromElem.props?.name || fromElem.id;
                        signalsHTML += `<div class="signal-item" data-signal="${signalName}">${signalName} (${conn.toPort})</div>`;
                    }
                }
            });

            // ... (где-то выше код сбора signalsHTML) ...

            contentHTML = `
                <div class="modal-row">
                    <label>Количество входов:</label>
                    <input type="number" id="prop-input-count" value="${props.inputCount || 2}" min="1" max="10">
                </div>

                <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->
                <div style="display: flex; gap: 15px; margin-bottom: 15px; height: 140px;">
                    <!-- Левая колонка: Сигналы -->
                    <div style="flex: 1; display: flex; flex-direction: column;">
                        <label style="margin-bottom: 5px; display:block;">Входные сигналы:</label>
                        <div class="signal-list" id="signal-list" style="flex: 1; overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px solid #4a90d9;">
                            ${signalsHTML || '<div style="color:#888;padding:5px;">Нет сигналов</div>'}
                        </div>
                    </div>

                    <!-- Правая колонка: Шаблоны -->
                    <div style="flex: 1; display: flex; flex-direction: column;">
                        <label style="margin-bottom: 5px; display:block;">Шаблоны:</label>
                        <div class="signal-list" id="template-list" style="flex: 1; overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px solid #4a90d9;">
                            <div style="color:#888;padding:5px;">Загрузка…</div>
                        </div>
                    </div>
                </div>

                <!-- Нижний блок: Поле формулы (во всю ширину) -->
                <div class="modal-row">
                    <label>Выражение формулы:</label>
                    <textarea id="prop-expression"
                              style="width: 100%; min-height: 80px; font-family: monospace; font-size: 14px; line-height: 1.4;"
                              spellcheck="false">${props.expression || ''}</textarea>
                    <small style="color:#999; display:block; margin-top:4px;">
                        Двойной клик по сигналу или шаблону вставит его в позицию курсора (или заменит выделенный текст).
                    </small>
                </div>
            `;
        }
        if (!contentHTML) {
            contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных свойств.</div>`;
            }
        contentHTML += `
            <div class="modal-row">
```

```
5392                 <label>Комментарий:</label>
5393                 <textarea id="prop-comment" placeholder="Комментарий к элементу...">${props.comment || ''}</textarea>
5394             </div>
5395             `;
5396 
5397 
5398         modalContent.innerHTML = contentHTML;
5399         // modal.js — внутри showPropertiesModal, блок if (elemType === 'formula')
5400         if (elemType === 'formula') {
5401             const listEl = document.getElementById('template-list');
5402 
5403             // Создаём tooltip элемент (один на всю страницу)
5404             let tooltip = document.getElementById('template-tooltip');
5405             if (!tooltip) {
5406                 tooltip = document.createElement('div');
5407                 tooltip.id = 'template-tooltip';
5408                 tooltip.className = 'template-tooltip';
5409                 document.body.appendChild(tooltip);
5410             }
5411 
5412             (async () => {
5413                 try {
5414                     const data = await Settings.fetchFormulaTemplates();
5415                     const items = data.templates || [];
5416 
5417                     if (!items.length) {
5418                         listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет шаблонов</div>';
5419                         return;
5420                     }
5421 
5422                     listEl.innerHTML = items.map(t => {
5423                         const sig = `${t.name}(${(t.args || []).join(', ')})`;
5424                         // Сохраняем description в data-атрибут
5425                         const desc = (t.description || '').replace(/"/g, '&quot;');
5426                         return `<div class="signal-item template-item"
5427                                     data-insert="${sig}"
5428                                     data-name="${t.name}"
5429                                     data-description="${desc}">${sig}</div>`;
5430                     }).join('');
5431 
5432                     // Обработчики для каждого шаблона
5433                     listEl.querySelectorAll('.template-item').forEach(div => {
5434                         // Двойной клик — вставка
5435                         div.addEventListener('dblclick', () => {
5436                             const insert = div.dataset.insert;
5437                             const textarea = document.getElementById('prop-expression');
5438                             insertAtCursor(textarea, insert);
5439                         });
5440 
5441                         // Наведение — показать tooltip
5442                         div.addEventListener('mouseenter', (e) => {
5443                             const description = div.dataset.description;
5444                             const name = div.dataset.name;
5445 
5446                             if (!description) return;
5447 
5448                             tooltip.innerHTML = `
5449                                 <div class="template-tooltip-title">${name}</div>
5450                                 <div>${description}</div>
5451                             `;
5452 
5453                             // Позиционируем tooltip
```

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

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

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



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

            let timer = null;

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

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

                results.style.display = 'block';

                results.querySelectorAll('.signal-result-item').forEach((div, i) => {
                div.addEventListener('click', () => {
                    const chosen = items[i];
                    input.value = chosen.Tagname;
                    descField.value = chosen.Description || '';
                    const dimField = document.getElementById('prop-dimension');
                    if (dimField) dimField.value = chosen.EngineeringUnit || chosen.Dimension || '';
                    results.style.display = 'none';
                });
                });
            };

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

```
5514
5515                // Показываем список только если пользователь реально использует маску
5516                if (!mask.includes('*')) {
5517                results.style.display = 'none';
5518                return;
5519                }
5520
5521                results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</
      div>';
5522                results.style.display = 'block';
5523
5524                try {
5525                // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
5526                const data = await Settings.fetchSignals(mask, 50);
5527                renderList(data.items || []);
5528                } catch (e) {
5529                results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка
      загрузки сигналов</div>';
5530                results.style.display = 'block';
5531                console.error(e);
5532                }
5533            };
5534
5535            input.addEventListener('input', () => {
5536                clearTimeout(timer);
5537                timer = setTimeout(search, 200); // debounce
5538            });
5539
5540            // опционально: закрывать список кликом вне
5541            document.addEventListener('mousedown', (e) => {
5542                if (!results.contains(e.target) && e.target !== input) {
5543                results.style.display = 'none';
5544                }
5545            }, { once: true });
5546        }
5547        modalOverlay.dataset.elementId = elemId;
5548        this.showModal('modal-overlay');
5549
5550        // Функция для умной вставки текста в позицию курсора
5551        const insertAtCursor = (field, text) => {
5552            if (!field) return;
5553
5554            // Получаем позиции выделения
5555            const startPos = field.selectionStart;
5556            const endPos = field.selectionEnd;
5557            const currentValue = field.value;
5558
5559            // Вставляем текст: (текст до) + (новый текст) + (текст после)
5560            field.value = currentValue.substring(0, startPos) +
5561                        text +
5562                        currentValue.substring(endPos, currentValue.length);
5563
5564            // Возвращаем фокус и ставим курсор сразу после вставленного текста
5565            field.focus();
5566            const newCursorPos = startPos + text.length;
5567            field.setSelectionRange(newCursorPos, newCursorPos);
5568        };
5569
5570        // Обработчик вставки сигналов для формулы
5571        if (elemType === 'formula') {
5572            document.querySelectorAll('.signal-item').forEach(item => {
5573            item.addEventListener('dblclick', () => {
5574                const signal = item.dataset.signal;
5575                const textarea = document.getElementById('prop-expression');
5576
```

```
5577                 // БЫЛО: textarea.value += signal;
5578                 // СТАЛО:
5579                 insertAtCursor(textarea, signal);
5580                 });
5581             });
5582         }
5583     },
5584
5585     /**
5586      * Сохранить свойства элемента
5587      */
5588 /**
5589  * Сохранить свойства элемента
5590  */
5591     saveElementProperties() {
5592         try {
5593             const modalOverlay = document.getElementById('modal-overlay');
5594             const elemId = modalOverlay.dataset.elementId;
5595             const elemData = AppState.elements[elemId];
5596             const elem = document.getElementById(elemId);
5597             if (!elemData) {
5598                 alert('⚠ Элемент не найден — возможно, он был удалён или
     переименован.');
5599                 console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
5600                 this.hideModal('modal-overlay');
5601                 return;
5602             }
5603
5604             const elemType = elemData.type;
5605
5606             if (elemType === 'input-signal') {
5607                 const name = document.getElementById('prop-name').value || 'Сигнал';
5608                 const description = document.getElementById('prop-description').value
     || '';
5609                 const signalType = document.getElementById('prop-signal-type').value;
5610                 const dimension = document.getElementById('prop-dimension').value ||
     '';
5611                 elemData.props.dimension = dimension;
5612
5613                 const oldSignalType = elemData.props.signalType;
5614                 elemData.props.name = name;
5615                 elemData.props.description = description;
5616                 elemData.props.signalType = signalType;
5617
5618                 if (oldSignalType !== signalType) {
5619                     AppState.connections = AppState.connections.filter(conn => {
5620                     if (conn.fromElement === elemId) {
5621                         const toPortIndex = parseInt(conn.toPort.split('-')[1]);
5622                         const inputType = getInputPortType(conn.toElement,
     toPortIndex);
5623                         return areTypesCompatible(signalType, inputType);
5624                     }
5625                     return true;
5626                     });
5627                 }
5628
5629                 const { html } = Elements.createElementHTML(
5630                     elemType, elemId, elemData.x, elemData.y, elemData.props,
     elemData.width, elemData.height
5631                 );
5632                 elem.outerHTML = html;
5633
5634                 Elements.setupElementHandlers(elemId);
5635                 Connections.drawConnections();
5636             } else if (elemType === 'if') {
```

```
5637                 const operator = document.getElementById('prop-operator').value;
5638                 elemData.props.operator = operator;
5639                 const symbol = elem.querySelector('.element-symbol');
5640                 if (symbol) symbol.textContent = operator;
5641 
5642             } else if (elemType === 'const') {
5643                 const value = parseFloat(document.getElementById('prop-value').value)
      || 0;
5644                 elemData.props.value = value;
5645                 const symbol = elem.querySelector('.element-symbol');
5646                 if (symbol) symbol.textContent = String(value);
5647 
5648             } else if (elemType === 'formula') {
5649                 const expression = document.getElementById('prop-expression').value;
5650                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
5651 
5652                 elemData.props.expression = expression;
5653                 elemData.props.inputCount = inputCount;
5654 
5655                 const symbol = elem.querySelector('.element-symbol');
5656                 if (symbol) {
5657                     symbol.textContent = expression.length > 12 ? `$
      {expression.slice(0, 12)}…` : (expression || 'f(x)');
5658                 }
5659 
5660                 Elements.updateFormulaInputs(elemId, inputCount);
5661                 Elements.updateElementSize(elemId);  // ← Добавляем это
5662             } else if (elemType === 'and' || elemType === 'or') {
5663                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
5664                 elemData.props.inputCount = inputCount;
5665 
5666                 Elements.updateLogicGateInputs(elemId, inputCount);
5667                 Elements.updateElementSize(elemId);  // ← Добавляем это
5668 
5669                 const symbol = elem.querySelector('.element-symbol');
5670                 if (symbol) {
5671                     symbol.textContent = elemType === 'and' ? '∧' : '∨';
5672                 }
5673 
5674         } else if (elemType === 'output') {
5675                 const label = document.getElementById('prop-label').value || 'Выход';
5676                 const outputGroup = document.getElementById('prop-output-group').value
      || '';
5677 
5678                 elemData.props.label = label;
5679                 elemData.props.outputGroup = outputGroup;
5680 
5681                 const symbol = elem.querySelector('.element-symbol');
5682                 if (symbol) symbol.textContent = label;
5683             }
5684             else if (elemType === 'group') {
5685                 const title = document.getElementById('prop-title').value || 'Группа';
5686                 elemData.props.title = title;
5687                 const titleEl = elem.querySelector('.group-title');
5688                 if (titleEl) titleEl.textContent = title;
5689                 }
5690             const commentEl = document.getElementById('prop-comment');
5691             if (commentEl) elemData.props.comment = commentEl.value || '';
5692 
5693             this.hideModal('modal-overlay');
5694 
5695         } catch (error) {
5696             console.error('❌ Ошибка при сохранении свойств:', error);
```

```
5697                alert('Ошибка сохранения: ' + error.message);
5698            }
5699        },
5700
5701        /**
5702         * Показать свойства проекта
5703         */
5704        showProjectPropertiesModal() {
5705            const content = document.getElementById('project-modal-content');
5706            const project = AppState.project;
5707
5708            // Генерируем HTML для списка выходов только если модуль загружен
5709            let outputsHtml = '';
5710            if (typeof Outputs !== 'undefined' && AppState.outputs) {
5711                const logicalOutputsHtml = AppState.outputs.logical.length > 0
5712                    ? AppState.outputs.logical.map(output => `
5713                        <div class="output-item"
5714                            data-element-id="${output.elementId}"
5715                            onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
5716                            onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
5717                            onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
5718                            <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
    '❌'}</span>
5719                            <span class="output-name">${output.elementName}</span>
5720                            <span class="output-port">→ ${output.portLabel}</span>
5721                        </div>
5722                    `).join('')
5723                    : '<div class="no-outputs">Нет логических выходов</div>';
5724
5725                const numericOutputsHtml = AppState.outputs.numeric.length > 0
5726                    ? AppState.outputs.numeric.map(output => `
5727                        <div class="output-item numeric"
5728                            data-element-id="${output.elementId}"
5729                            onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
5730                            onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
5731                            onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
5732                            <span class="output-icon">🔢</span>
5733                            <span class="output-name">${output.elementName}</span>
5734                            <span class="output-port">→ значение</span>
5735                        </div>
5736                    `).join('')
5737                    : '<div class="no-outputs">Нет числовых выходов</div>';
5738
5739                outputsHtml = `
5740                    <div class="modal-row">
5741                        <label>Выходные сигналы схемы:</label>
5742                        <div class="outputs-container">
5743                            <div class="outputs-section">
5744                                <div class="outputs-section-title">
5745                                    <span class="section-icon">🔀</span>
5746                                    Логические выходы (${AppState.outputs.logical.length})
5747                                </div>
5748                                <div class="outputs-list">
5749                                    ${logicalOutputsHtml}
5750                                </div>
5751                            </div>
5752                            <div class="outputs-section">
5753                                <div class="outputs-section-title">
5754                                    <span class="section-icon">📐</span>
```

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

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

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

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

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

```
рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
                </div>
            </div>

            ${outputsHtml}
        `;

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

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

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

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

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

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

        this.hideModal('project-modal-overlay');
    }
};


/**
 * Модуль управления проектом (сохранение, загрузка)
 * project.js
 */

// --- миграция id: '-' -> '_' с обновлением всех ссылок ---
function migrateIdsDashToUnderscore() {
  const map = {};
```

```
5867
5868    // 1) собрать map старых id → новых
5869    Object.values(AppState.elements).forEach(el => {
5870      if (typeof el.id === 'string' && el.id.includes('-')) {
5871        map[el.id] = el.id.replace(/-/g, '_');
5872      }
5873    });
5874
5875    if (!Object.keys(map).length) return;
5876
5877    // 2) DOM id + data-element
5878    Object.entries(map).forEach(([oldId, newId]) => {
5879      const dom = document.getElementById(oldId);
5880      if (dom) dom.id = newId;
5881
5882      if (dom) {
5883        dom.querySelectorAll('[data-element]').forEach(p => {
5884          if (p.dataset.element === oldId) p.dataset.element = newId;
5885        });
5886      }
5887    });
5888
5889    // 3) AppState.elements ключи
5890    Object.entries(map).forEach(([oldId, newId]) => {
5891      const el = AppState.elements[oldId];
5892      if (!el) return;
5893      el.id = newId;
5894      AppState.elements[newId] = el;
5895      delete AppState.elements[oldId];
5896    });
5897
5898    // 4) connections
5899    AppState.connections.forEach(c => {
5900      if (map[c.fromElement]) c.fromElement = map[c.fromElement];
5901      if (map[c.toElement]) c.toElement = map[c.toElement];
5902    });
5903
5904    // 5) формулы
5905    const escapeRegex = s => s.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
5906    Object.values(AppState.elements).forEach(el => {
5907      if (el.type === 'formula' && el.props?.expression) {
5908        let expr = el.props.expression;
5909        Object.entries(map).forEach(([oldId, newId]) => {
5910          const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
      z0-9_])`, 'g');
5911          expr = expr.replace(re, (m, p1) => `${p1}${newId}`);
5912        });
5913        el.props.expression = expr;
5914      }
5915    });
5916
5917    // 6) selected + modal
5918    if (map[AppState.selectedElement]) AppState.selectedElement =
      map[AppState.selectedElement];
5919    const modal = document.getElementById('modal-overlay');
5920    if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
      map[modal.dataset.elementId];
5921  }
5922
5923  const Project = {
5924      /**
5925       * Инициализация
5926       */
5927          /**
5928       * Инициализация
```

```
5929      */
5930  init() {
5931      document.getElementById('btn-new').addEventListener('click', () =>
      this.newProject());
5932      document.getElementById('btn-save').addEventListener('click', () =>
      this.saveProject());
5933      document.getElementById('btn-load').addEventListener('click', () =>
      this.openProjectListModal());
5934      document.getElementById('btn-project-settings').addEventListener('click', () => {
5935          Modal.showProjectPropertiesModal();
5936      });
5937
5938      // Работа с модалкой выбора проекта
5939      this.projectList = [];
5940      this.filteredProjectList = [];
5941      this.selectedProjectFilename = null;
5942
5943      document.getElementById('project-cancel').addEventListener('click', () =>
      this.closeProjectListModal());
5944      document.getElementById('project-refresh').addEventListener('click', () =>
      this.refreshProjectList());
5945
5946      document.getElementById('project-load').addEventListener('click', () => {
5947          if (this.selectedProjectFilename) {
5948              this.loadProjectFromList(this.selectedProjectFilename);
5949          }
5950      });
5951
5952      document.getElementById('project-search').addEventListener('input', (event) => {
5953          this.filterProjectList(event.target.value);
5954      });
5955  },
5956
5957      /**
5958       * Новый проект
5959       */
5960      newProject() {
5961          if (Object.keys(AppState.elements).length > 0) {
5962              if (!confirm('Создать новый проект? Несохранённые изменения будут
      потеряны.')) {
5963                  return;
5964              }
5965          }
5966
5967          document.getElementById('workspace').innerHTML = '';
5968          document.getElementById('connections-svg').innerHTML = '';
5969
5970          resetState();
5971          Viewport.updateTransform();
5972      },
5973
5974          /**
5975       * Запрос имени файла и загрузка с сервера
5976       */
5977      async loadProjectPrompt() {
5978          const filename = window.prompt(
5979              "Введите имя файла проекта для загрузки (с сервера). Пример:
      scheme_logic.json",
5980              AppState.project.code ? `${AppState.project.code}_$
      {AppState.project.type}.json` : "scheme_type.json"
5981          );
5982
5983          if (!filename) return; // Отмена
5984
5985          try {
```

```
5986              // Используем обертку из Settings.js для запроса к /api/project/load
5987              const data = await Settings.loadProject(filename);
5988
5989              // Если загрузка успешна, вызываем основную функцию обработки данных
5990              this._processLoadedData(data);
5991              alert(`Проект "${filename}" успешно загружен с сервера.`);
5992
5993          } catch (error) {
5994              console.error('Ошибка загрузки проекта:', error);
5995              alert(`Ошибка загрузки проекта: ${error.message}`);
5996          }
5997      },
5998
5999      /**
6000       * Сохранение проекта
6001       */
6002          async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
6003          // 1. Проверяем свойства проекта
6004          if (!AppState.project.code) {
6005              Modal.showProjectPropertiesModal();
6006              alert('Пожалуйста, укажите код проекта перед сохранением.');
6007              return;
6008          }
6009
6010          // Обновляем размеры рамок перед сохранением
6011          updateFrameChildren();
6012          // ✅ нормализуем, даже если проект был открыт до фикса
6013          migrateIdsDashToUnderscore();
6014
6015          // ✅ подчистим связи прямо перед сохранением
6016          const exists = (id) => !!AppState.elements[id];
6017          AppState.connections = (AppState.connections || [])
6018            .map(c => ({
6019              ...c,
6020              fromElement: exists(c.fromElement) ? c.fromElement :
      c.fromElement.replace(/-/g, '_'),
6021              toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g,
      '_')
6022            }))
6023            .filter(c => exists(c.fromElement) && exists(c.toElement))
6024            .filter((c, idx, arr) => {
6025              const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
6026              return arr.findIndex(x =>
6027                `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
6028              ) === idx;
6029            });
6030          // ✅ 1. Генерируем код заранее
6031          let generatedCode = '';
6032          if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
      {
6033              try {
6034                  generatedCode = CodeGen.generate() || '';
6035              } catch (err) {
6036                  console.error('Code generation failed:', err);
6037              }
6038          }
6039
6040          // 2. Сборка объекта проекта
6041          const project = {
6042              version: '1.0',
6043              project: AppState.project,
6044              elements: AppState.elements,
6045              connections: AppState.connections,
6046              counter: AppState.elementCounter,
6047              viewport: {
```

```
6048                 zoom: AppState.viewport.zoom,
6049                 panX: AppState.viewport.panX,
6050                 panY: AppState.viewport.panY
6051             },
6052             code: generatedCode
6053         };
6054 
6055         const filename = `${AppState.project.code || 'scheme'}_$
      {AppState.project.type}.json`;
6056 
6057         // 3. Сохранение на сервер
6058         try {
6059             await Settings.saveProject(filename, project);
6060             alert(`Проект успешно сохранен на сервере как: ${filename}`);
6061         } catch (error) {
6062             console.error('Ошибка сохранения проекта:', error);
6063             alert(`Ошибка сохранения проекта: ${error.message}`);
6064         }
6065     },
6066 
6067     async showProjectList() {
6068         try {
6069             const result = await Settings.listProjects(); // нужно реализовать в
      settings.js
6070             const list = result.projects || [];
6071 
6072             if (list.length === 0) {
6073                 alert('Проекты в папке не найдены.');
6074                 return;
6075             }
6076 
6077             const choice = window.prompt(
6078                 'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code ||
      p.filename} — ${p.description}`).join('\n') +
6079                 '\n\nВведите номер проекта для загрузки:',
6080                 '1'
6081             );
6082             const index = parseInt(choice, 10) - 1;
6083             if (isNaN(index) || !list[index]) return;
6084 
6085             await this.loadProjectByFilename(list[index].filename);
6086         } catch (error) {
6087             console.error(error);
6088             alert('Не удалось получить список проектов: ' + error.message);
6089         }
6090     },
6091 
6092     async loadProjectByFilename(filename) {
6093         try {
6094             const data = await Settings.loadProject(filename);
6095             this._processLoadedData(data);
6096             alert(`Проект "${filename}" загружен.`);
6097         } catch (error) {
6098             console.error(error);
6099             alert('Ошибка загрузки проекта: ' + error.message);
6100         }
6101     },
6102 
6103     openProjectListModal() {
6104   const modal = document.getElementById('modal-project-list');
6105   modal.classList.remove('hidden');
6106   document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
6107   this.refreshProjectList();
6108 },
```

```
6109
6110  closeProjectListModal() {
6111    const modal = document.getElementById('modal-project-list');
6112    modal.classList.add('hidden');
6113    document.body.classList.remove('modal-open');
6114  },
6115
6116  async refreshProjectList() {
6117    const tbody = document.getElementById('project-list-body');
6118    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
6119    try {
6120      const result = await Settings.listProjects();
6121      this.projectList = result.projects || [];
6122      this.filteredProjectList = [...this.projectList];
6123      this.renderProjectList();
6124    } catch (err) {
6125      console.error(err);
6126      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
6127    }
6128  },
6129
6130  renderProjectList() {
6131    const tbody = document.getElementById('project-list-body');
6132    const loadBtn = document.getElementById('project-load');
6133    loadBtn.disabled = true;
6134    this.selectedProjectFilename = null;
6135
6136    if (!this.filteredProjectList.length) {
6137      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
6138      return;
6139    }
6140
6141    tbody.innerHTML = '';
6142    this.filteredProjectList.forEach((item) => {
6143      const tr = document.createElement('tr');
6144      tr.innerHTML = `
6145        <td>${item.filename}</td>
6146        <td>${item.code || ''}</td>
6147        <td>${item.description || ''}</td>
6148        <td>${item.type || ''}</td>
6149      `;
6150      tr.addEventListener('click', () => {
6151        this.highlightRow(tr);
6152        this.selectedProjectFilename = item.filename;
6153        loadBtn.disabled = false;
6154      });
6155      tr.addEventListener('dblclick', () => {
6156        this.highlightRow(tr);
6157        this.selectedProjectFilename = item.filename;
6158        loadBtn.disabled = false;
6159        this.loadProjectFromList(item.filename);
6160      });
6161      tbody.appendChild(tr);
6162    });
6163  },
6164
6165  highlightRow(row) {
6166    const tbody = row.parentElement;
6167    [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
6168    row.classList.add('selected');
6169  },
6170
```

```
6171
6172  // Фильтр по поисковой строке
6173  filterProjectList(query) {
6174    const q = (query || '').trim().toLowerCase();
6175    if (!q) {
6176      this.filteredProjectList = [...this.projectList];
6177    } else {
6178      this.filteredProjectList = this.projectList.filter((item) => {
6179        return [
6180          item.filename,
6181          item.code,
6182          item.description,
6183          item.type
6184        ].some((field) => (field || '').toLowerCase().includes(q));
6185      });
6186    }
6187    this.renderProjectList();
6188  },
6189
6190  async loadProjectFromList(filename) {
6191    try {
6192      const data = await Settings.loadProject(filename);
6193      this._processLoadedData(data);
6194      this.closeProjectListModal();
6195      alert(`Проект "${filename}" успешно загружен.`);
6196    } catch (error) {
6197      console.error(error);
6198      alert('Ошибка загрузки проекта: ' + error.message);
6199    }
6200  },
6201
6202
6203
6204
6205
6206
6207      /**
6208       * Загрузка проекта
6209       */
6210   _processLoadedData(data) {
6211    try {
6212      document.getElementById('workspace').innerHTML = '';
6213      document.getElementById('connections-svg').innerHTML = '';
6214      resetState();
6215
6216      if (data.project) {
6217        AppState.project = { ...AppState.project, ...data.project };
6218      }
6219
6220      AppState.elementCounter = data.counter || 0;
6221
6222      if (data.viewport) {
6223        AppState.viewport.zoom = data.viewport.zoom || 1;
6224        AppState.viewport.panX = data.viewport.panX || 0;
6225        AppState.viewport.panY = data.viewport.panY || 0;
6226      }
6227
6228      const elements = data.elements || {};
6229      Object.values(elements)
6230        .filter(e => e.type === 'output-frame')
6231        .forEach(elemData => {
6232          Elements.addElement(
6233            elemData.type,
6234            elemData.x,
6235            elemData.y,
```

```
6236            elemData.props,
6237            elemData.id,
6238            elemData.width,
6239            elemData.height
6240          );
6241        });
6242
6243      Object.values(elements)
6244        .filter(e => e.type !== 'output-frame')
6245        .forEach(elemData => {
6246          Elements.addElement(
6247            elemData.type,
6248            elemData.x,
6249            elemData.y,
6250            elemData.props,
6251            elemData.id,
6252            elemData.width,
6253            elemData.height
6254          );
6255        });
6256
6257      AppState.connections = data.connections || [];
6258
6259      // ✅ ВСТАВЬ ЭТОТ БЛОК СРАЗУ ЗДЕСЬ (до вычисления счётчика)
6260      //Object.values(AppState.elements).forEach(e => {
6261      //  if (typeof e.id === 'string') {
6262      //    e.id = e.id.replace(/-/g, '_');
6263      //  }
6264      //  if (e.props?.name) {
6265      //    e.props.name = e.props.name.replace(/-/g, '_');
6266      //  }
6267      //});
6268      // ✅ конец добавленной секции
6269      // ✅ Миграция id: '-' -> '_'
6270      migrateIdsDashToUnderscore();
6271      // ✅ очистка соединений: удалить битые и дубликаты
6272      const exists = (id) => !!AppState.elements[id];
6273
6274      AppState.connections = (AppState.connections || [])
6275        // оставить только те, где оба конца реально существуют
6276        .filter(c => exists(c.fromElement) && exists(c.toElement))
6277        // убрать дубликаты
6278        .filter((c, idx, arr) => {
6279          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
6280          return arr.findIndex(x =>
6281            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
6282          ) === idx;
6283        });
6284
6285
6286      // корректно восстанавливаем счётчик
6287      const counterFromFile = Number(data.counter);
6288      AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
6289
6290      const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
6291          if (!el?.id) return max;
6292          const match = String(el.id).match(/_(\d+)$/);   // теперь хвост по
      подчёркиванию
6293          const num = match ? parseInt(match[1], 10) : NaN;
6294          return Number.isFinite(num) ? Math.max(max, num) : max;
6295      }, 0);
6296
6297      AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
6298
6299      Viewport.updateTransform();
```

```
6300      Connections.drawConnections();
6301      updateFrameChildren();
6302
6303    } catch (e) {
6304      alert('Ошибка обработки данных проекта: ' + e.message);
6305      console.error(e);
6306    }
6307  }
6308  };
6309
6310  // settings.js — ПОЛНАЯ ИСПРАВЛЕННАЯ ВЕРСИЯ
6311
6312  const Settings = {
6313    config: null,
6314    templates: null,
6315    apiUrl: '',  // ← Добавь это! Пустая строка = относительные пути
6316
6317    async init() {
6318      try {
6319        const r = await fetch('/api/settings');
6320        if (r.ok) this.config = await r.json();
6321      } catch (e) {
6322        console.warn('Settings load failed:', e);
6323      }
6324      try {
6325        const t = await this.fetchFormulaTemplates();
6326        this.templates = t.templates || [];
6327      } catch (e) {
6328        this.templates = [];
6329      }
6330    },
6331
6332    getTemplatesMap() {
6333      const map = {};
6334      (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
6335      return map;
6336    },
6337
6338    // ← ОДНА функция fetchSignals с cache-busting
6339    async fetchSignals(mask, limit = 50) {
6340      const timestamp = Date.now();
6341      const url = `${this.apiUrl}/api/signals?q=${encodeURIComponent(mask || '')}
      &limit=${limit}&_t=${timestamp}`;
6342      const r = await fetch(url);
6343      if (!r.ok) throw new Error('Failed to fetch signals');
6344      return await r.json();
6345    },
6346
6347    async saveProject(filename, projectData) {
6348      if (!filename.endsWith('.json')) {
6349        filename += '.json';
6350      }
6351      const r = await fetch(`${this.apiUrl}/api/project/save`, {
6352        method: 'POST',
6353        headers: { 'Content-Type': 'application/json' },
6354        body: JSON.stringify({
6355          filename: filename,
6356          content: projectData
6357        })
6358      });
6359      if (!r.ok) throw new Error('Failed to save project');
6360      return r.json();
6361    },
6362
6363    async listProjects() {
```

```
6364      const r = await fetch(`${this.apiUrl}/api/project/list`);
6365      if (!r.ok) throw new Error('Failed to list projects');
6366      return r.json();
6367    },
6368
6369    async fetchFormulaTemplates() {
6370      const r = await fetch(`${this.apiUrl}/api/formula-templates`);
6371      if (!r.ok) throw new Error('Failed to fetch formula templates');
6372      return await r.json();
6373    },
6374
6375    async loadProject(filename) {
6376      if (!filename.endsWith('.json')) {
6377        filename += '.json';
6378      }
6379      const r = await fetch(`${this.apiUrl}/api/project/load/$
      {encodeURIComponent(filename)}`);
6380      if (!r.ok) {
6381        if (r.status === 404) {
6382          throw new Error(`Project "${filename}" not found (404)`);
6383        }
6384        throw new Error('Failed to load project');
6385      }
6386      return r.json();
6387    }
6388  };
6389
6390  /**
6391   * Глобальное состояние приложения
6392   * state.js
6393   */
6394
6395  const AppState = {
6396      // Элементы схемы
6397      elements: {},
6398      connections: [],
6399      elementCounter: 0,
6400
6401      // Выделение
6402      selectedElement: null,
6403      selectedElements: [],    // ← ДОБАВЬ ЭТУ СТРОКУ (для группы)
6404
6405      // Перетаскивание
6406      draggingElement: null,
6407      dragOffset: { x: 0, y: 0 },
6408      isDraggingFromPalette: false,
6409      dragPreview: null,
6410      dragType: null,
6411
6412      // Соединения
6413      connectingFrom: null,
6414      connectingFromType: null,
6415      tempLine: null,
6416
6417      // Resize
6418      resizing: null,
6419
6420      // Viewport (масштабирование и перемещение)
6421      viewport: {
6422          zoom: 1,
6423          panX: 0,
6424          panY: 0,
6425          isPanning: false,
6426          lastMouseX: 0,
6427          lastMouseY: 0
```

```
6428      },
6429
6430      // Свойства проекта
6431      project: {
6432          code: '',
6433          type: PROJECT_TYPE.PARAMETER,
6434          // Для параметра
6435          dimension: '',
6436          // Для правила
6437          possibleCause: '',
6438          guidelines: ''
6439      },
6440
6441      // Выходные сигналы (автоматически определяются)
6442      outputs: {
6443          logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
6444          numeric: []     // Числовые выходы (формулы)
6445      }
6446  };
6447
6448  /**
6449   * Сброс состояния
6450   */
6451  function resetState() {
6452      AppState.elements = {};
6453      AppState.connections = [];
6454      AppState.elementCounter = 0;
6455      AppState.selectedElement = null;
6456      AppState.draggingElement = null;
6457      AppState.connectingFrom = null;
6458      AppState.tempLine = null;
6459      AppState.resizing = null;
6460
6461      AppState.viewport = {
6462          zoom: 1,
6463          panX: 0,
6464          panY: 0,
6465          isPanning: false,
6466          lastMouseX: 0,
6467          lastMouseY: 0
6468      };
6469
6470      AppState.project = {
6471          code: '',
6472          type: PROJECT_TYPE.PARAMETER,
6473          description: '',
6474          dimension: '',
6475          possibleCause: '',
6476          guidelines: ''
6477      };
6478
6479      AppState.outputs = {
6480          logical: [],
6481          numeric: []
6482      };
6483  }
6484
6485  /**
6486   * Вспомогательные функции
6487   * utils.js
6488   */
6489
6490  /**
6491   * Генерация уникального ID
6492   */
```

```
6493  function generateId() {
6494      AppState.elementCounter++;
6495      return `elem_${AppState.elementCounter}`;
6496  }
6497
6498  function getInputPortType(elementId, portIdentifier) {
6499      const element = AppState.elements[elementId];
6500      if (!element) return SIGNAL_TYPE.ANY;
6501
6502      const config = ELEMENT_TYPES[element.type];
6503      if (!config) return SIGNAL_TYPE.ANY;
6504
6505      let portIndex = portIdentifier;
6506
6507      // Обработка технического порта условия
6508      if (typeof portIdentifier === 'string') {
6509          if (portIdentifier === 'cond-0' && config.hasConditionPort) {
6510              return config.conditionPortType || SIGNAL_TYPE.LOGIC;
6511          }
6512
6513          if (portIdentifier.startsWith('in-')) {
6514              portIndex = parseInt(portIdentifier.split('-')[1], 10);
6515          }
6516      }
6517
6518      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
6519          portIndex = 0;
6520      }
6521
6522      // Динамические входы для AND/OR берут тип из конфига
6523      if ((element.type === 'and' || element.type === 'or')) {
6524          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
6525      }
6526
6527      if (element.type === 'formula') {
6528          return SIGNAL_TYPE.ANY;
6529      }
6530
6531      const types = config.inputTypes || [];
6532      if (types.length === 0) return SIGNAL_TYPE.ANY;
6533
6534      if (portIndex < types.length) {
6535          return types[portIndex] || SIGNAL_TYPE.ANY;
6536      }
6537
6538      return types[types.length - 1] || SIGNAL_TYPE.ANY;
6539  }
6540
6541  function getOutputPortType(elementId, portIdentifier) {
6542      const element = AppState.elements[elementId];
6543      if (!element) return SIGNAL_TYPE.ANY;
6544
6545      const config = ELEMENT_TYPES[element.type];
6546      if (!config) return SIGNAL_TYPE.ANY;
6547
6548      let portIndex = portIdentifier;
6549
6550      if (typeof portIdentifier === 'string') {
6551          if (portIdentifier.startsWith('out-')) {
6552              portIndex = parseInt(portIdentifier.split('-')[1], 10);
6553          }
6554      }
6555
6556      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
```

```
6557          portIndex = 0;
6558      }
6559
6560      const types = config.outputTypes || [];
6561      if (types.length === 0) return SIGNAL_TYPE.ANY;
6562
6563      if (portIndex < types.length) {
6564          return types[portIndex] || SIGNAL_TYPE.ANY;
6565      }
6566
6567      return types[types.length - 1] || SIGNAL_TYPE.ANY;
6568  }
6569  /**
6570   * Проверка совместимости типов сигналов
6571   *
6572   * Новая логика:
6573   * - ANY совместим со всем
6574   * - TRUE совместим с LOGIC, TRUE, ANY
6575   * - FALSE совместим с LOGIC, FALSE, ANY
6576   * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
6577   * - NUMERIC совместим с NUMERIC, ANY
6578   */
6579  function areTypesCompatible(outputType, inputType) {
6580      // Если один из типов ANY - совместимы
6581      if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
6582          return true;
6583      }
6584
6585      // Если типы одинаковые - совместимы
6586      if (outputType === inputType) {
6587          return true;
6588      }
6589
6590      // TRUE/FALSE совместимы с LOGIC
6591      if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
6592          inputType === SIGNAL_TYPE.LOGIC) {
6593          return true;
6594      }
6595
6596      // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
6597      if (outputType === SIGNAL_TYPE.LOGIC &&
6598          (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
6599          return true;
6600      }
6601
6602      return false;
6603  }
6604
6605  /**
6606   * Проверка, находится ли элемент внутри рамки
6607   */
6608  function isInsideFrame(elemId, frameId) {
6609      const elem = AppState.elements[elemId];
6610      const frame = AppState.elements[frameId];
6611
6612      if (!elem || !frame || frame.type !== 'output-frame') return false;
6613
6614      const elemCenterX = elem.x + elem.width / 2;
6615      const elemCenterY = elem.y + elem.height / 2;
6616
6617      return elemCenterX > frame.x &&
6618             elemCenterX < frame.x + frame.width &&
6619             elemCenterY > frame.y &&
6620             elemCenterY < frame.y + frame.height;
6621  }
```

```
6622
6623  /**
6624   * Обновить принадлежность элементов к рамкам
6625   */
6626  function updateFrameChildren() {
6627      // Сначала очистим children у рамок и parentFrame у всех элементов
6628      Object.values(AppState.elements).forEach(elem => {
6629          if (elem.type === 'output-frame') {
6630              elem.children = [];
6631          } else {
6632              // удаляем parentFrame по умолчанию (пересчитаем ниже)
6633              if (elem.parentFrame) delete elem.parentFrame;
6634          }
6635      });
6636
6637      // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
      попадает
6638      Object.values(AppState.elements).forEach(elem => {
6639          if (!elem || elem.type === 'output-frame') return;
6640
6641          Object.values(AppState.elements).forEach(frame => {
6642              if (!frame || frame.type !== 'output-frame') return;
6643
6644              if (isInsideFrame(elem.id, frame.id)) {
6645                  // добавляем в массив детей рамки
6646                  frame.children.push(elem.id);
6647                  // отмечаем у элемента родительскую рамку
6648                  if (AppState.elements[elem.id]) {
6649                      AppState.elements[elem.id].parentFrame = frame.id;
6650                  }
6651              }
6652          });
6653      });
6654  }
6655
6656  /**
6657   * Преобразование координат экрана в координаты холста
6658   */
6659  function screenToCanvas(screenX, screenY) {
6660      const container = document.getElementById('workspace-container');
6661      const rect = container.getBoundingClientRect();
6662
6663      const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
6664      const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;
6665
6666      return { x, y };
6667  }
6668
6669  /**
6670   * Преобразование координат холста в координаты экрана
6671   */
6672  function canvasToScreen(canvasX, canvasY) {
6673      const container = document.getElementById('workspace-container');
6674      const rect = container.getBoundingClientRect();
6675
6676      const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
6677      const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;
6678
6679      return { x, y };
6680  }
6681
6682  /**
6683   * Проверка, является ли порт выходным (не подключен к другим элементам)
6684   */
6685  function isOutputPort(elemId, portIndex) {
```

```
6686      const portKey = `out-${portIndex}`;
6687
6688      // Проверяем, есть ли соединения от этого порта
6689      const hasConnection = AppState.connections.some(conn =>
6690          conn.fromElement === elemId && conn.fromPort === portKey
6691      );
6692
6693      return !hasConnection;
6694  }
6695
6696  /**
6697   * Получить информацию о выходном порте
6698   */
6699  function getOutputPortInfo(elemId, portIndex) {
6700      const elem = AppState.elements[elemId];
6701      if (!elem) return null;
6702
6703      const config = ELEMENT_TYPES[elem.type];
6704      if (!config) return null;
6705
6706      return {
6707          elementId: elemId,
6708          elementType: elem.type,
6709          elementName: config.name,
6710          portIndex: portIndex,
6711          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
6712          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
6713          // Дополнительная информация для идентификации
6714          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
6715      };
6716  }
6717
6718  function splitArgsTopLevel(argStr) {
6719    const out = [];
6720    let cur = '';
6721    let depth = 0;
6722    for (let i = 0; i < argStr.length; i++) {
6723      const ch = argStr[i];
6724      if (ch === '(') depth++;
6725      if (ch === ')') depth--;
6726      if (ch === ',' && depth === 0) {
6727        out.push(cur.trim());
6728        cur = '';
6729      } else {
6730        cur += ch;
6731      }
6732    }
6733    if (cur.trim()) out.push(cur.trim());
6734    return out;
6735  }
6736
6737  function expandFormulaTemplates(expr, templatesMap) {
6738    if (!expr) return expr;
6739    if (!templatesMap) return expr;
6740
6741    // несколько проходов на случай вложенных шаблонов
6742    for (let pass = 0; pass < 10; pass++) {
6743      let changed = false;
6744
6745      expr = expr.replace(/([A-Za-z_]\w*)\s*\(([^()]|\([^()]*\))*\)/g, (match, name) =>
      {
6746        const tpl = templatesMap[name];
6747        if (!tpl) return match;
6748
```

```
6749        // вытащим аргументы вручную: name(....)
6750        const open = match.indexOf('(');
6751        const close = match.lastIndexOf(')');
6752        const inside = match.slice(open + 1, close);
6753
6754        const args = splitArgsTopLevel(inside);
6755        const formal = tpl.args || [];
6756        let body = String(tpl.body || '0');
6757
6758        // если количество аргументов не совпало — не трогаем (лучше так, чем сломать)
6759        if (args.length !== formal.length) return match;
6760
6761        formal.forEach((f, i) => {
6762          const re = new RegExp(`\\b${f}\\b`, 'g');
6763          body = body.replace(re, `(${args[i]})`);
6764        });
6765
6766        changed = true;
6767        return `(${body})`;
6768      });
6769
6770      if (!changed) break;
6771    }
6772
6773    return expr;
6774  }
6775
6776  /**
6777   * Модуль управления viewport (масштабирование и перемещение)
6778   * viewport.js
6779   */
6780
6781  const Viewport = {
6782      /**
6783       * Инициализация viewport
6784       */
6785      init() {
6786          this.setupZoomControls();
6787          this.setupPanning();
6788          this.setupMouseWheel();
6789          this.setupMinimap();
6790          this.setupCursorPosition();
6791          this.updateTransform();
6792          const container = document.getElementById('workspace-container');
6793          const rect = container.getBoundingClientRect();
6794          AppState.viewport.panX = 100; // немного отступить от левого края
6795          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
      AppState.viewport.zoom;
6796          this.updateTransform();
6797      },
6798
6799      /**
6800       * Настройка кнопок масштабирования
6801       */
6802      setupZoomControls() {
6803          document.getElementById('btn-zoom-in').addEventListener('click', () => {
6804              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
6805          });
6806
6807          document.getElementById('btn-zoom-out').addEventListener('click', () => {
6808              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
6809          });
6810
6811          document.getElementById('btn-zoom-reset').addEventListener('click', () => {
6812              this.setZoom(1);
```

```
6813              this.setPan(0, 0);
6814          });
6815
6816          document.getElementById('btn-zoom-fit').addEventListener('click', () => {
6817              this.fitToContent();
6818          });
6819      },
6820
6821      /**
6822       * Настройка перемещения (pan)
6823       */
6824      setupPanning() {
6825          const container = document.getElementById('workspace-container');
6826
6827          container.addEventListener('mousedown', (e) => {
6828              // Средняя кнопка мыши или пробел + левая кнопка
6829              if (e.button === 1 || (e.button === 0 && e.target === container)) {
6830                  e.preventDefault();
6831                  AppState.viewport.isPanning = true;
6832                  AppState.viewport.lastMouseX = e.clientX;
6833                  AppState.viewport.lastMouseY = e.clientY;
6834                  container.style.cursor = 'grabbing';
6835              }
6836          });
6837
6838          document.addEventListener('mousemove', (e) => {
6839              if (AppState.viewport.isPanning) {
6840                  const dx = e.clientX - AppState.viewport.lastMouseX;
6841                  const dy = e.clientY - AppState.viewport.lastMouseY;
6842
6843                  this.setPan(
6844                      AppState.viewport.panX + dx,
6845                      AppState.viewport.panY + dy
6846                  );
6847
6848                  AppState.viewport.lastMouseX = e.clientX;
6849                  AppState.viewport.lastMouseY = e.clientY;
6850              }
6851          });
6852
6853          document.addEventListener('mouseup', (e) => {
6854              if (AppState.viewport.isPanning) {
6855                  AppState.viewport.isPanning = false;
6856                  document.getElementById('workspace-container').style.cursor = '';
6857              }
6858          });
6859
6860          // Клавиша пробел для режима перемещения
6861          document.addEventListener('keydown', (e) => {
6862              if (e.code === 'Space' && !e.repeat) {
6863                  document.getElementById('workspace-container').style.cursor = 'grab';
6864              }
6865          });
6866
6867          document.addEventListener('keyup', (e) => {
6868              if (e.code === 'Space') {
6869                  document.getElementById('workspace-container').style.cursor = '';
6870              }
6871          });
6872      },
6873
6874      /**
6875       * Настройка масштабирования колесом мыши
6876       */
6877      setupMouseWheel() {
```

```
6878          const container = document.getElementById('workspace-container');
6879
6880          container.addEventListener('wheel', (e) => {
6881              e.preventDefault();
6882
6883              const rect = container.getBoundingClientRect();
6884              const mouseX = e.clientX - rect.left;
6885              const mouseY = e.clientY - rect.top;
6886
6887              // Позиция мыши на холсте до масштабирования
6888              const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6889              const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6890
6891              // Новый масштаб
6892              const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
      VIEWPORT_CONFIG.zoomStep;
6893              const newZoom = Math.max(
6894                  VIEWPORT_CONFIG.minZoom,
6895                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
6896              );
6897
6898              // Корректируем pan, чтобы точка под курсором осталась на месте
6899              const newPanX = mouseX - canvasPosBeforeX * newZoom;
6900              const newPanY = mouseY - canvasPosBeforeY * newZoom;
6901
6902              AppState.viewport.zoom = newZoom;
6903              AppState.viewport.panX = newPanX;
6904              AppState.viewport.panY = newPanY;
6905
6906              this.updateTransform();
6907          }, { passive: false });
6908      },
6909
6910      /**
6911       * Установить масштаб
6912       */
6913      setZoom(zoom) {
6914          const container = document.getElementById('workspace-container');
6915          const rect = container.getBoundingClientRect();
6916
6917          // Центр экрана
6918          const centerX = rect.width / 2;
6919          const centerY = rect.height / 2;
6920
6921          // Позиция центра на холсте
6922          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6923          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6924
6925          // Новый масштаб
6926          const newZoom = Math.max(
6927              VIEWPORT_CONFIG.minZoom,
6928              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
6929          );
6930
6931          // Корректируем pan
6932          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
6933          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
6934          AppState.viewport.zoom = newZoom;
6935
6936          this.updateTransform();
6937      },
```

```
6938
6939      /**
6940       * Установить смещение
6941       */
6942      setPan(x, y) {
6943          AppState.viewport.panX = x;
6944          AppState.viewport.panY = y;
6945          this.updateTransform();
6946      },
6947
6948      /**
6949       * Вписать содержимое в экран
6950       */
6951      fitToContent() {
6952          const elements = Object.values(AppState.elements);
6953          if (elements.length === 0) {
6954              this.setZoom(1);
6955              this.setPan(0, 0);
6956              return;
6957          }
6958
6959          // Находим границы содержимого
6960          let minX = Infinity, minY = Infinity;
6961          let maxX = -Infinity, maxY = -Infinity;
6962
6963          elements.forEach(elem => {
6964              minX = Math.min(minX, elem.x);
6965              minY = Math.min(minY, elem.y);
6966              maxX = Math.max(maxX, elem.x + elem.width);
6967              maxY = Math.max(maxY, elem.y + elem.height);
6968          });
6969
6970          const contentWidth = maxX - minX;
6971          const contentHeight = maxY - minY;
6972
6973          const container = document.getElementById('workspace-container');
6974          const rect = container.getBoundingClientRect();
6975
6976          const padding = 50;
6977          const availableWidth = rect.width - padding * 2;
6978          const availableHeight = rect.height - padding * 2;
6979
6980          const zoomX = availableWidth / contentWidth;
6981          const zoomY = availableHeight / contentHeight;
6982          const newZoom = Math.min(zoomX, zoomY, 1);
6983
6984          AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
6985          AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
      (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
6986          AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
      (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
6987
6988          this.updateTransform();
6989      },
6990
6991      /**
6992       * Обновить трансформацию
6993       */
6994      updateTransform() {
6995          const workspace = document.getElementById('workspace');
6996          const svg = document.getElementById('connections-svg');
6997
6998          const transform = `translate(${AppState.viewport.panX}px, $
      {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
6999
```

```
7000          workspace.style.transform = transform;
7001          svg.style.transform = transform;
7002
7003          // Обновляем отображение масштаба
7004          document.getElementById('zoom-level').textContent = `${Math.round(AppState.viewport.zoom * 100)}%`;
7005
7006          // Обновляем мини-карту
7007          this.updateMinimap();
7008      },
7009
7010      /**
7011       * Настройка мини-карты
7012       */
7013      setupMinimap() {
7014          const minimap = document.getElementById('minimap');
7015          const canvas = document.getElementById('minimap-canvas');
7016
7017          canvas.width = MINIMAP_CONFIG.width;
7018          canvas.height = MINIMAP_CONFIG.height;
7019
7020          // Клик по мини-карте для перемещения
7021          minimap.addEventListener('click', (e) => {
7022              const rect = minimap.getBoundingClientRect();
7023              const x = e.clientX - rect.left;
7024              const y = e.clientY - rect.top;
7025
7026              this.navigateToMinimapPosition(x, y);
7027          });
7028      },
7029
7030      /**
7031       * Обновить мини-карту
7032       */
7033      updateMinimap() {
7034          const canvas = document.getElementById('minimap-canvas');
7035          const ctx = canvas.getContext('2d');
7036          const viewportEl = document.getElementById('minimap-viewport');
7037
7038          // Очищаем
7039          ctx.fillStyle = '#0a0a1a';
7040          ctx.fillRect(0, 0, canvas.width, canvas.height);
7041
7042          // Масштаб мини-карты
7043          const scale = Math.min(
7044              canvas.width / VIEWPORT_CONFIG.canvasWidth,
7045              canvas.height / VIEWPORT_CONFIG.canvasHeight
7046          );
7047
7048          // Рисуем элементы
7049          Object.values(AppState.elements).forEach(elem => {
7050              const x = elem.x * scale;
7051              const y = elem.y * scale;
7052              const w = Math.max(elem.width * scale, 2);
7053              const h = Math.max(elem.height * scale, 2);
7054
7055              ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
7056              ctx.fillRect(x, y, w, h);
7057          });
7058
7059          // Рисуем viewport
7060          const container = document.getElementById('workspace-container');
7061          const rect = container.getBoundingClientRect();
7062
7063          const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
```

```
7064          const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
7065          const vpW = (rect.width / AppState.viewport.zoom) * scale;
7066          const vpH = (rect.height / AppState.viewport.zoom) * scale;
7067
7068          viewportEl.style.left = `${vpX}px`;
7069          viewportEl.style.top = `${vpY}px`;
7070          viewportEl.style.width = `${vpW}px`;
7071          viewportEl.style.height = `${vpH}px`;
7072      },
7073
7074      /**
7075       * Перейти к позиции на мини-карте
7076       */
7077      navigateToMinimapPosition(minimapX, minimapY) {
7078          const canvas = document.getElementById('minimap-canvas');
7079          const container = document.getElementById('workspace-container');
7080          const rect = container.getBoundingClientRect();
7081
7082          const scale = Math.min(
7083              canvas.width / VIEWPORT_CONFIG.canvasWidth,
7084              canvas.height / VIEWPORT_CONFIG.canvasHeight
7085          );
7086
7087          const canvasX = minimapX / scale;
7088          const canvasY = minimapY / scale;
7089
7090          // Центрируем viewport на этой точке
7091          AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
7092          AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
7093
7094          this.updateTransform();
7095      },
7096
7097      /**
7098       * Отслеживание позиции курсора
7099       */
7100      setupCursorPosition() {
7101          const container = document.getElementById('workspace-container');
7102
7103          container.addEventListener('mousemove', (e) => {
7104              const pos = screenToCanvas(e.clientX, e.clientY);
7105              document.getElementById('cursor-pos').textContent =
7106                  `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
7107          });
7108      }
7109  };
7110
7111  styles.css
7112
7113  * {
7114      margin: 0;
7115      padding: 0;
7116      box-sizing: border-box;
7117  }
7118
7119  body {
7120      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
7121      background: #1a1a2e;
7122      color: #eee;
7123      overflow: hidden;
7124  }
7125
7126  #app {
7127      display: flex;
7128      flex-direction: column;
```

```
7129      height: 100vh;
7130  }
7131
7132  /* ====== МЕНЮ ====== */
7133  #menu {
7134      background: #16213e;
7135      padding: 10px 20px;
7136      display: flex;
7137      gap: 10px;
7138      align-items: center;
7139      border-bottom: 2px solid #0f3460;
7140      z-index: 100;
7141      flex-wrap: wrap;
7142  }
7143
7144  .menu-btn {
7145      background: #0f3460;
7146      color: #eee;
7147      border: none;
7148      padding: 8px 16px;
7149      border-radius: 5px;
7150      cursor: pointer;
7151      transition: background 0.3s;
7152      font-size: 13px;
7153  }
7154
7155  .menu-btn:hover {
7156      background: #e94560;
7157  }
7158
7159  .menu-separator {
7160      width: 1px;
7161      height: 30px;
7162      background: #0f3460;
7163      margin: 0 10px;
7164  }
7165
7166  .zoom-controls {
7167      display: flex;
7168      align-items: center;
7169      gap: 8px;
7170      background: #0a0a1a;
7171      padding: 5px 10px;
7172      border-radius: 5px;
7173  }
7174
7175  .zoom-btn {
7176      width: 30px;
7177      height: 30px;
7178      padding: 0;
7179      font-size: 18px;
7180      font-weight: bold;
7181  }
7182
7183  #zoom-level {
7184      min-width: 50px;
7185      text-align: center;
7186      font-size: 12px;
7187      color: #aaa;
7188  }
7189
7190  /* ====== ОСНОВНАЯ ОБЛАСТЬ ====== */
7191  #main {
7192      display: flex;
7193      flex: 1;
```

```
7194      overflow: hidden;
7195  }
7196
7197  /* ====== ПАЛИТРА ====== */
7198  #palette {
7199      width: 200px;
7200      background: #16213e;
7201      padding: 15px;
7202      border-right: 2px solid #0f3460;
7203      overflow-y: auto;
7204      z-index: 10;
7205      flex-shrink: 0;
7206  }
7207
7208  #palette h3 {
7209      margin-bottom: 15px;
7210      color: #e94560;
7211      text-align: center;
7212      font-size: 14px;
7213  }
7214
7215  .palette-section {
7216      margin-bottom: 15px;
7217  }
7218
7219  .palette-section-title {
7220      font-size: 11px;
7221      color: #888;
7222      margin-bottom: 8px;
7223      padding-bottom: 3px;
7224      border-bottom: 1px solid #333;
7225  }
7226
7227  .palette-item {
7228      background: #0f3460;
7229      padding: 8px;
7230      margin-bottom: 6px;
7231      border-radius: 8px;
7232      cursor: grab;
7233      text-align: center;
7234      transition: all 0.3s;
7235      border: 2px solid transparent;
7236      user-select: none;
7237  }
7238
7239  .palette-item:hover {
7240      border-color: #e94560;
7241      transform: scale(1.02);
7242  }
7243
7244  .palette-item:active {
7245      cursor: grabbing;
7246  }
7247
7248  .palette-item svg {
7249      width: 50px;
7250      height: 32px;
7251      margin-bottom: 2px;
7252      pointer-events: none;
7253  }
7254
7255  .palette-item-name {
7256      font-size: 10px;
7257      color: #aaa;
7258      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;
    width: 5000px;
    height: 5000px;
}

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

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

/* ====== ЭЛЕМЕНТЫ ====== */
.element {
    position: absolute;
    background: #0f3460;
    border: 2px solid #4a90d9;
    border-radius: 8px;
    cursor: move;
    user-select: none;
```

```
7324      z-index: 10;
7325      display: flex;
7326      flex-direction: column;
7327  }
7328
7329  .element.selected {
7330      border-color: #e94560;
7331      box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
7332  }
7333
7334  .element-header {
7335      background: #4a90d9;
7336      padding: 5px 10px;
7337      border-radius: 5px 5px 0 0;
7338      font-size: 11px;
7339      font-weight: bold;
7340      text-align: center;
7341      white-space: nowrap;
7342      overflow: hidden;
7343      text-overflow: ellipsis;
7344  }
7345
7346  .element-body {
7347      padding: 10px;
7348      display: flex;
7349      justify-content: space-between;
7350      align-items: center;
7351      flex: 1;
7352      gap: 8px;
7353  }
7354
7355  .element-symbol {
7356      font-size: 16px;
7357      font-weight: bold;
7358      flex: 1;
7359      text-align: center;
7360      padding: 0 5px;
7361      word-break: break-all;
7362      color: #eee;
7363  }
7364
7365  /* ====== ПОРТЫ ====== */
7366  .ports-left, .ports-right {
7367      display: flex;
7368      flex-direction: column;
7369      justify-content: space-around;
7370      gap: 10px;
7371      height: 100%;
7372  }
7373
7374  .port {
7375      width: 14px;
7376      height: 14px;
7377      border-radius: 50%;
7378      border: 2px solid #fff;
7379      cursor: crosshair;
7380      transition: all 0.2s;
7381      position: relative;
7382      flex-shrink: 0;
7383  }
7384
7385  .port:hover { transform: scale(1.3); }
7386  .port.input { margin-left: -8px; }
7387  .port.output { margin-right: -8px; }
7388  .port.connected { background: #4ade80; }
```

```
7389
7390  /* Типы портов */
7391  .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
7392  .port.logic-port:hover { background: #c084fc; }
7393  .port.logic-port.connected { background: #7c3aed; }
7394
7395  .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
7396  .port.number-port:hover { background: #60a5fa; }
7397  .port.number-port.connected { background: #2563eb; }
7398
7399  .port.any-port { background: #6b7280; border-color: #d1d5db; }
7400  .port.any-port:hover { background: #9ca3af; }
7401  .port.any-port.connected { background: #4b5563; }
7402
7403  .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !
      important; }
7404  .port.output.no-port { background: #f87171 !important; border-color: #fecaca !
      important; }
7405
7406  .port.incompatible { opacity: 0.3; cursor: not-allowed; }
7407  .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
7408
7409  /* ====== RESIZE HANDLES ====== */
7410  .resize-handle {
7411      position: absolute;
7412      width: 12px;
7413      height: 12px;
7414      background: #e94560;
7415      border: 1px solid #fff;
7416      border-radius: 3px;
7417      z-index: 20;
7418      opacity: 0;
7419      transition: opacity 0.2s;
7420  }
7421  .element.selected .resize-handle { opacity: 0.8; }
7422  .resize-handle:hover { opacity: 1; }
7423  .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
7424  .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor:
      ew-resize; }
7425  .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%);
      cursor: ns-resize; }
7426
7427
7428  /* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
7429  .element.input-signal {
7430      background: transparent;
7431      border: none;
7432  }
7433
7434  .element.input-signal .element-header {
7435      display: none; /* У трапеции нет заголовка */
7436  }
7437
7438  .element.input-signal .element-body {
7439      padding: 0;
7440      background: #0f3460;
7441      border: 2px solid #4a90d9;
7442      clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
7443      display: flex;
7444      justify-content: space-between;
7445      align-items: center;
7446      padding-left: 15px;
7447      padding-right: 25px;
7448  }
7449
```

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

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

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

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

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

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

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


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

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

```
7515      border: 2px solid #f59e0b;
7516      background: #fff7ed;
7517      margin: 0; /* Сбрасываем лишние отступы */
7518  }
7519  .element.formula .condition-port:hover { background: #fde68a; }
7520
7521
7522  /* ====== СОЕДИНЕНИЯ ====== */
7523  .connection {
7524      fill: none !important;  /* ← добавляем !important */
7525      stroke: #4a90d9;
7526      stroke-width: 2.5;
7527  }
7528  .connection:hover {
7529      stroke: #e94560;
7530      stroke-width: 4;
7531  }
7532
7533  .connection.logic-conn { stroke: #a855f7; }
7534  .connection.numeric-conn { stroke: #3b82f6; }
7535  .connection.any-conn { stroke: #6b7280; }
7536  .connection.true-conn { stroke: #4ade80; }
7537  .connection.false-conn { stroke: #f87171; }
7538
7539  .connection.yes-conn { stroke: #4ade80; }
7540  .connection.no-conn { stroke: #f87171; }
7541
7542  .temp-connection {
7543      fill: none !important;  /* ← добавляем !important */
7544      stroke: #e94560;
7545      stroke-width: 2;
7546      stroke-dasharray: 5, 5;
7547  }
7548  .temp-connection.invalid { stroke: #ef4444; }
7549
7550  /* ====== ПРОЧЕЕ ====== */
7551  .drag-preview {
7552      position: fixed;
7553      pointer-events: none;
7554      opacity: 0.8;
7555      z-index: 1000;
7556      background: #0f3460;
7557      border: 2px solid #e94560;
7558      border-radius: 8px;
7559      padding: 10px 15px;
7560      color: #fff;
7561      font-size: 12px;
7562  }
7563
7564  #minimap {
7565      position: absolute;
7566      bottom: 20px;
7567      right: 20px;
7568      width: 200px;
7569      height: 150px;
7570      background: #16213e;
7571      border: 2px solid #0f3460;
7572      border-radius: 8px;
7573      overflow: hidden;
7574      z-index: 50;
7575  }
7576
7577  #minimap-canvas { width: 100%; height: 100%; }
7578  #minimap-viewport {
7579      position: absolute;
```

```
7580      border: 2px solid #e94560;
7581      background: rgba(233, 69, 96, 0.2);
7582      pointer-events: none;
7583  }
7584
7585  #viewport-info {
7586      position: absolute;
7587      bottom: 20px;
7588      left: 20px;
7589      background: rgba(22, 33, 62, 0.9);
7590      padding: 8px 12px;
7591      border-radius: 5px;
7592      font-size: 11px;
7593      color: #888;
7594      z-index: 50;
7595      display: flex;
7596      gap: 15px;
7597  }
7598  #selection-info { color: #e94560; }
7599
7600  #modal-overlay, .modal-overlay-class {
7601      display: none;
7602      position: fixed;
7603      top: 0; left: 0;
7604      width: 100%; height: 100%;
7605      background: rgba(0, 0, 0, 0.7);
7606      z-index: 1000;
7607      justify-content: center;
7608      align-items: center;
7609  }
7610
7611  #modal, .modal-class {
7612      background: #16213e;
7613      border-radius: 10px;
7614      padding: 20px;
7615      min-width: 400px;
7616      max-width: 600px;
7617      max-height: 80vh;
7618      overflow-y: auto;
7619      border: 2px solid #0f3460;
7620  }
7621
7622  #modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
7623  .modal-row { margin-bottom: 15px; }
7624  .modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
7625  .modal-row input, .modal-row select, .modal-row textarea {
7626      width: 100%;
7627      padding: 10px;
7628      background: #0f3460;
7629      border: 1px solid #4a90d9;
7630      border-radius: 5px;
7631      color: #eee;
7632      font-size: 14px;
7633  }
7634  .modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline:
      none; border-color: #e94560; }
7635  .modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
7636  .signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-
      radius: 5px; padding: 5px; margin-top: 5px; }
7637  .signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size:
      12px; }
7638  .signal-item:hover { background: #4a90d9; }
7639  .modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top:
      20px; }
7640  .modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer;
```

```
      font-size: 14px; transition: background 0.3s; }
7641  .modal-btn.save { background: #4ade80; color: #000; }
7642  .modal-btn.save:hover { background: #22c55e; }
7643  .modal-btn.cancel { background: #6b7280; color: #fff; }
7644  .modal-btn.cancel:hover { background: #4b5563; }
7645
7646  #context-menu {
7647      display: none;
7648      position: fixed;
7649      background: #16213e;
7650      border: 1px solid #0f3460;
7651      border-radius: 5px;
7652      padding: 5px 0;
7653      z-index: 1001;
7654      min-width: 150px;
7655      box-shadow: 0 5px 20px rgba(0,0,0,0.3);
7656  }
7657  .context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition:
      background 0.2s; }
7658  .context-item:hover { background: #0f3460; }
7659
7660  #file-input { display: none; }
7661
7662  .project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
7663  .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; }
7664  .project-type-btn:hover { border-color: #e94560; }
7665  .project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
7666  .project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
7667  .project-type-btn .type-name { font-weight: bold; }
7668  .project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }
7669
7670  .conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-
      radius: 8px; margin-top: 10px; }
7671  .conditional-fields.visible { display: block; }
7672
7673  ::-webkit-scrollbar { width: 8px; height: 8px; }
7674  ::-webkit-scrollbar-track { background: #0a0a1a; }
7675  ::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
7676  ::-webkit-scrollbar-thumb:hover { background: #e94560; }
7677
7678  /* Стили для выходов */
7679  .output-btn { position: relative; }
7680  .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; }
7681  .output-counter:empty, .output-counter[style*="display: none"] { display: none; }
7682  .element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
7683  .element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important;
      border-color: #fbbf24 !important; }
7684  .port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation:
      pulse-output 1.5s infinite; }
7685  @keyframes pulse-output {
7686      0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
7687      50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
7688  }
7689
7690  .outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-
      height: 250px; overflow-y: auto; }
7691  .outputs-section { margin-bottom: 15px; }
7692  .outputs-section:last-child { margin-bottom: 0; }
7693  .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;
}
.port.false-port.connected {
    background: #dc2626 !important;
}

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

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

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

```

```
7751  /* === Кастомный элемент “Группа” === */
7752  .element.group {
7753    background: rgba(107, 114, 128, 0.12);
7754    border: 2px dashed #6b7280;
7755    border-radius: 8px;
7756    position: absolute;
7757    z-index: 1;              /* ниже обычных элементов (у них z-index: 10) */
7758  }
7759
7760  .element.group .group-title {
7761    pointer-events: auto;
7762  }
7763
7764  .group-title {
7765    position: absolute;
7766    top: -20px;
7767    left: 5px;
7768    font-size: 11px;
7769    color: #ccc;
7770    background: #16213e;
7771    padding: 2px 6px;
7772    border-radius: 4px;
7773    pointer-events: auto; /* можно кликнуть для выбора */
7774  }
7775
7776  .modal.hidden { display: none; }
7777  .modal { position: fixed; inset: 0; display: flex; align-items: center; justify-
      content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
7778  .modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px;
      max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow:
      hidden; }
7779  .modal__content--wide { width: 800px; }
7780  .modal__title { margin: 0; }
7781
7782  .project-list__toolbar { display: flex; gap: 12px; }
7783  .project-list__toolbar input { flex: 1; padding: 6px 10px; }
7784  .project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd;
      border-radius: 6px; }
7785  .project-list__table { width: 100%; border-collapse: collapse; }
7786  .project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom:
      1px solid #eee; }
7787  .project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
7788  .project-list__table tbody tr:hover { background: #f0f6ff; }
7789  .project-list__empty { text-align: center; color: #888; padding: 16px; }
7790  .modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
7791  .project-list__table th,
7792  .project-list__table td {
7793    color: #111;          /* насыщённый чёрный текст */
7794    padding: 8px 12px;
7795    border-bottom: 1px solid #eee;
7796  }
7797  .modal__content--wide {
7798    width: 860px;
7799    max-height: 90vh;    /* занимает 90% экрана */
7800  }
7801
7802  .project-list__table-container {
7803    flex: 1;
7804    overflow: auto;
7805    border: 1px solid #ddd;
7806    border-radius: 6px;
7807    max-height: 60vh;     /* много строк */
7808  }
7809
7810  .element-comment {
```

```
7811    padding: 6px 10px 10px;
7812    font-size: 11px;
7813    color: #cbd5e1;
7814    opacity: 0.9;
7815    border-top: 1px solid rgba(255,255,255,0.08);
7816    white-space: pre-wrap;
7817    word-break: break-word;
7818  }
7819
7820  .element-comment:empty { display: none; }
7821
7822  /* Tooltip для шаблонов формул */
7823  .template-item {
7824      position: relative;
7825  }
7826
7827  .template-tooltip {
7828      position: fixed;
7829      background: #1a1a2e;
7830      border: 1px solid #4a90d9;
7831      border-radius: 6px;
7832      padding: 8px 12px;
7833      color: #e0e0e0;
7834      font-size: 12px;
7835      max-width: 280px;
7836      line-height: 1.4;
7837      box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
7838      z-index: 10000;
7839      pointer-events: none;
7840      opacity: 0;
7841      transition: opacity 0.15s ease;
7842  }
7843
7844  .template-tooltip.visible {
7845      opacity: 1;
7846  }
7847
7848  .template-tooltip::before {
7849      content: '';
7850      position: absolute;
7851      top: -6px;
7852      left: 20px;
7853      border-left: 6px solid transparent;
7854      border-right: 6px solid transparent;
7855      border-bottom: 6px solid #4a90d9;
7856  }
7857
7858  .template-tooltip-title {
7859      font-weight: 600;
7860      color: #4a90d9;
7861      margin-bottom: 4px;
7862  }
```