

```
1 main.py:
2 import os
3 import json
4 from typing import Dict, List
5 import pandas as pd
6 from fastapi import FastAPI, HTTPException, Request, Response
7 from fastapi.responses import JSONResponse
8 from fastapi.staticfiles import StaticFiles
9 import uuid
10 import pickle
11 from fastapi.middleware.cors import CORSMiddleware
12
13 import tempfile
14 from io import BytesIO
15
16 import pickle # <-- отсутствовал
17
18
19
20
21
22
23 BASE_DIR = os.path.dirname(os.path.abspath(__file__))
24 SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
25 TEMPLATES_PATH = os.path.join(BASE_DIR, "formula_templates.json")
26 SIGNAL_INDEX_PATH = os.path.join(BASE_DIR, ".signal_index.pkl")
27
28 def load_templates() -> Dict:
29     if not os.path.exists(TEMPLATES_PATH):
30         return {"templates": []}
31     with open(TEMPLATES_PATH, "r", encoding="utf-8") as f:
32         return json.load(f)
33
34 def load_settings() -> Dict:
35     with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
36         return json.load(f)
37
38 def load_signals_from_folder(folder: str) -> List[Dict]:
39     # folder может быть относительным
40     folder_abs = folder if os.path.isabs(folder) else
41     os.path.normpath(os.path.join(BASE_DIR, folder))
42     if not os.path.isdir(folder_abs):
43         raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")
44
45     signals_map = {} # Tagname -> Description (последний wins)
46     for name in os.listdir(folder_abs):
47         if not name.lower().endswith(".csv"):
48             continue
49         path = os.path.join(folder_abs, name)
50         try:
51             df = pd.read_csv(path, sep=';')[['Tagname', 'Description', 'Engineering
Unit']]
52             df = df.dropna(subset=['Tagname'])
53             for _, row in df.iterrows():
54                 tag = str(row['Tagname']).strip()
55                 desc = "" if pd.isna(row['Description']) else
56                 str(row['Description']).strip()
57                 unit = "" if pd.isna(row['Engineering Unit']) else
58                 str(row['Engineering Unit']).strip()
59                 desc = ", ".join([desc, unit])
60                 if tag:
61                     signals_map[tag] = desc
62         except Exception as e:
63             # пропускаем "плохие" csv, но можно логировать
64             print(f"[WARN] failed to read {path}: {e}")
65
```

```
62
63     # в список
64     out = [{ 'Tagname': k, 'Description': v} for k, v in signals_map.items()]
65     out.sort(key=lambda x: x['Tagname'])
66     return out
67
68
69
70 app = FastAPI()
71 app.add_middleware(
72     CORSMiddleware,
73     allow_origins=["http://localhost:8501"],
74     allow_credentials=True,
75     allow_methods=["*"],
76     allow_headers=["*"],
77 )
78
79 # Кэш сигналов в памяти
80 STATE = {
81     "settings": None,
82     "signals": None,
83     "signal_index": None
84 }
85
86
87 def build_signal_index(folder: str) -> Dict[str, List[str]]:
88     """
89         При запуске: проходим по всем CSV файлам и создаем индекс
90         signal_name -> list of files where it's present
91     """
92     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
93         os.path.join(BASE_DIR, folder)
94     )
95
96     if not os.path.isdir(folder_abs):
97         raise FileNotFoundError(f"Signal data folder not found: {folder_abs}")
98
99     signal_index = {}
100
101    print(f"[INFO] Building signal index from {folder_abs}...")
102
103    for filename in os.listdir(folder_abs):
104        if not filename.lower().endswith(".csv"):
105            continue
106
107        filepath = os.path.join(folder_abs, filename)
108
109        try:
110            # Читаем только первую строку (заголовок)
111            df_header = pd.read_csv(
112                filepath,
113                nrows=0, # Читаем только заголовок
114                encoding="ISO-8859-2",
115                sep=";"
116            )
117
118            columns = df_header.columns.tolist()
119
120            # Удаляем служебные столбцы из индекса
121            signal_columns = [c for c in columns if c not in ["DATE", "TIME",
122 "datetime"]]
123
124            for signal_name in signal_columns:
125                if signal_name not in signal_index:
126                    signal_index[signal_name] = []
```

```
126             signal_index[signal_name].append(filepath)
127
128         print(f" ✓ {filename}: {len(signal_columns)} signals")
129
130     except Exception as e:
131         print(f" ✗ Failed to index {filename}: {e}")
132         continue
133
134     print(f"[OK] Total unique signals indexed: {len(signal_index)}")
135
136 # Сохраняем индекс в pickle для быстрого восстановления
137 try:
138     with open(SIGNAL_INDEX_PATH, "wb") as f:
139         pickle.dump(signal_index, f)
140         print(f"[OK] Signal index cached to {SIGNAL_INDEX_PATH}")
141 except Exception as e:
142     print(f"[WARN] Failed to cache signal index: {e}")
143
144 return signal_index
145
146
147 def load_signal_index(folder: str) -> Dict[str, List[str]]:
148     """
149     Загружает индекс либо из кэша, либо перестраивает его
150     """
151     if os.path.exists(SIGNAL_INDEX_PATH):
152         try:
153             with open(SIGNAL_INDEX_PATH, "rb") as f:
154                 index = pickle.load(f)
155                 print(f"[OK] Signal index loaded from cache")
156                 return index
157         except Exception as e:
158             print(f"[WARN] Failed to load cached index: {e}")
159
160     # Если кэша нет, перестраиваем
161     return build_signal_index(folder)
162
163 def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
164 pd.DataFrame]:
165     """
166     Загружает только нужные сигналы из только нужных файлов
167     Returns: {signal_name -> DataFrame}
168     """
169     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
170         os.path.join(BASE_DIR, folder))
171
172     signal_index = STATE.get("signal_index", {})
173     if not signal_index:
174         raise RuntimeError("Signal index not initialized")
175
176     signal_names_set = set(signal_names)
177     found_signals = {}
178     files_to_load = set()
179
180     # Определяем, какие файлы нужно загружать
181     for signal_name in signal_names_set:
182         if signal_name in signal_index:
183             files_to_load.update(signal_index[signal_name])
184
185     print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)} files")
186
187     # Загружаем данные из файлов
188     for filepath in files_to_load:
```

```
189     try:
190         df = pd.read_csv(
191             filepath,
192             encoding="ISO-8859-2",
193             sep=";",
194         )
195
196         # Обработка даты/времени
197         df["TIME"] = df["TIME"].str.replace(", ", ".", regex=False)
198         df["TIME"] = df["TIME"].str.split(".").str[0]
199         combined = df["DATE"] + " " + df["TIME"]
200         df["datetime"] = pd.to_datetime(
201             combined, format="%d.%m.%Y %H:%M:%S", errors="coerce"
202         )
203         df = df.dropna(subset=["datetime"])
204         df = df.drop(['DATE', 'TIME'], axis=1)
205
206         # Сортируем по datetime
207         df = df.sort_values("datetime")
208
209         # Извлекаем только нужные сигналы
210         available_columns = set(df.columns) & signal_names_set
211         for signal_name in available_columns:
212             if signal_name not in found_signals:
213                 # Сохраняем datetime и значение сигнала
214                 found_signals[signal_name] = df[['datetime', signal_name]].copy()
215                 found_signals[signal_name].columns = ["datetime", "value"]
216
217         except Exception as e:
218             print(f"[WARN] Failed to read {filepath}: {e}")
219             continue
220
221     return found_signals
222
223
224
225
226
227
228 @app.on_event("startup")
229 def startup():
230     settings = load_settings()
231     STATE["settings"] = settings
232     folder = settings.get("signalDataFolder")
233     if not folder:
234         raise RuntimeError("settings.json: signalDataFolder is required")
235     STATE["signals"] = load_signals_from_folder(folder)
236     STATE["templates"] = load_templates()
237     STATE["signal_index"] = load_signal_index(settings.get("signalArchiveFolder"))
238
239     print(f"[OK] loaded signals: {len(STATE['signals'])}")
240     print(f"[OK] signal index has {len(STATE['signal_index'])} unique signals")
241     print(f"[OK] loaded templates: {len(STATE['templates'].get('templates', []))}")
242
243 @app.get("/api/settings")
244 def api_settings():
245     return STATE["settings"]
246
247 @app.get("/api/signals")
248 def api_signals(q: str = "", limit: int = 50):
249     """
250         q - маска со * (например *MAA*CP*)
251         """
252     signals = STATE["signals"] or []
253     if not q:
```

```
254         return {"items": signals[:limit], "total": len(signals)}
255
256     # маска * -> regex
257     import re
258     escaped = re.escape(q).replace(r"\*", ".*")
259     rx = re.compile("^" + escaped + "$", re.IGNORECASE)
260
261     items = [s for s in signals if rx.match(s["Tagname"])]
262     return {"items": items[:max(1, min(limit, 500))], "total": len(items)}
263
264 # Helper: возвращает абсолютный путь к файлу проекта
265 def get_project_path(filename: str):
266     folder = STATE["settings"].get("projectDataFolder")
267     if not folder:
268         raise RuntimeError("projectDataFolder not configured")
269
270     # Нормализуем путь к папке проектов
271     project_dir = folder if os.path.isabs(folder) else
272     os.path.normpath(os.path.join(BASE_DIR, folder))
273
274     # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
275     if '..' in filename or '/' in filename or '\\\\' in filename:
276         raise HTTPException(status_code=400, detail="Invalid filename")
277
278     path = os.path.join(project_dir, filename)
279     # Проверяем, что итоговый путь лежит внутри разрешенной директории
280     if not path.startswith(project_dir):
281         raise HTTPException(status_code=400, detail="Path traversal attempt")
282
283     return path
284
285 @app.post("/api/project/save")
286 async def save_project(request: Request):
287     try:
288         data = await request.json()
289         filename = data.get("filename")
290         content = data.get("content")
291
292         if not filename or not content:
293             raise HTTPException(status_code=400, detail="Filename and content are
294 required")
295
296         path = get_project_path(filename)
297
298         # Сохраняем как JSON
299         with open(path, "w", encoding="utf-8") as f:
300             json.dump(content, f, indent=2)
301
302         return {"status": "ok", "message": f"Project saved to {filename}"}
303
304     except HTTPException as e:
305         raise e
306     except Exception as e:
307         print(f"Error saving project: {e}")
308         raise HTTPException(status_code=500, detail="Internal server error during
309 save")
310
311     @app.get("/api/project/load/{filename}")
312     def load_project(filename: str):
313         try:
314             path = get_project_path(filename)
315
316             if not os.path.exists(path):
317                 raise HTTPException(status_code=404, detail="Project not found")
```

```
316         with open(path, "r", encoding="utf-8") as f:
317             content = json.load(f)
318
319     return content
320
321     except HTTPException as e:
322         raise e
323     except Exception as e:
324         print(f"Error loading project: {e}")
325         raise HTTPException(status_code=500, detail="Internal server error during
load")
326
327 @app.get("/api/formula-templates")
328 def api_formula_templates():
329     return STATE.get("templates") or {"templates": []}
330
331 @app.get("/api/project/list")
332 def list_projects():
333     folder = STATE["settings"].get("projectDataFolder")
334     if not folder:
335         raise HTTPException(status_code=500, detail="Project folder not configured")
336
337     project_dir = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
338     os.makedirs(project_dir, exist_ok=True)
339
340     projects = []
341     for fname in sorted(os.listdir(project_dir)):
342         if not fname.endswith(".json"):
343             continue
344         path = os.path.join(project_dir, fname)
345         try:
346             with open(path, "r", encoding="utf-8") as f:
347                 payload = json.load(f)
348             except Exception:
349                 continue
350             project_meta = payload.get("project", {})
351             projects.append({
352                 "filename": fname,
353                 "code": project_meta.get("code") or project_meta.get("tagname") or "",
354                 "description": project_meta.get("description") or "",
355                 "type": project_meta.get("type") or ""
356             })
357     return {"projects": projects}
358
359 @app.post("/api/signal-data")
360 async def api_signal_data(request: Request):
361     """
362     POST с JSON телом:
363     {
364         "signal_names": ["SIGNAL1", "SIGNAL2", ...],
365         "format": "parquet" # или "json"
366     }
367
368     Returns: Parquet файл с данными или JSON
369     """
370     try:
371         data = await request.json()
372         signal_names = data.get("signal_names", [])
373         output_format = data.get("format", "parquet") # По умолчанию Parquet
374
375         if not signal_names:
376             raise HTTPException(status_code=400, detail="signal_names is required")
377
378         folder = STATE["settings"].get("signalArchiveFolder")
```

```
379         if not folder:
380             raise HTTPException(status_code=500, detail="signalArchiveFolder not
381 configured")
382
383         # Загружаем данные сигналов
384         signals_data = load_signal_data_optimized(signal_names, folder)
385
386         # Подготавливаем ответ
387         response = {
388             "found": list(signals_data.keys()),
389             "not_found": [s for s in signal_names if s not in signals_data],
390             "format": output_format
391         }
392
393         if not signals_data:
394             raise HTTPException(status_code=404, detail="No signals found")
395
396         # Экспортируем данные в зависимости от формата
397         if output_format == "parquet":
398             return await _export_parquet(signals_data, response)
399         else:
400             return await _export_json(signals_data, response)
401
402     except HTTPException as e:
403         raise e
404     except Exception as e:
405         print(f"Error in api_signal_data: {e}")
406         raise HTTPException(status_code=500, detail=str(e))
407
408     async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
409         """
410             Экспортирует данные в Parquet (НАМНОГО меньше чем JSON!)
411         """
412
413         from fastapi.responses import FileResponse
414
415         try:
416             # Создаем временный файл
417             with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
418                 tmp_path = tmp.name
419
420                 # Каждый сигнал сохраняем как отдельную таблицу в Parquet
421                 # Используем структуру: datetime, signal_name, value
422                 rows = []
423                 for signal_name, df in signals_data.items():
424                     df_copy = df.copy()
425                     df_copy["signal_name"] = signal_name
426                     rows.append(df_copy)
427
428                 combined = pd.concat(rows, ignore_index=True)
429                 combined.to_parquet(tmp_path, compression='snappy', index=False)
430
431                 file_size = os.path.getsize(tmp_path)
432                 print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size / 1024 / 1024:.2f} MB")
433
434             return FileResponse(
435                 tmp_path,
436                 media_type="application/octet-stream",
437                 filename="signal_data.parquet",
438                 headers={"X-Signal-Meta": json.dumps(meta)})
439
440         except Exception as e:
441             print(f"[ERROR] Parquet export failed: {e}")
442             raise
```

```
442
443
444     async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
445         """
446             Экспортирует данные в JSON (медленнее и больше, но совместимее)
447             """
448             from fastapi.responses import JSONResponse
449
450             try:
451                 # Формируем JSON с каждым сигналом отдельно
452                 data_dict = {}
453                 for signal_name, df in signals_data.items():
454                     df_copy = df.copy()
455                     df_copy["datetime"] = df_copy["datetime"].astype(str)
456                     data_dict[signal_name] = df_copy.to_dict(orient="records")
457
458                 response_data = {
459                     **meta,
460                     "data": data_dict
461                 }
462
463             return JSONResponse(response_data)
464
465         except Exception as e:
466             print(f"[ERROR] JSON export failed: {e}")
467             raise
468
469     # Простое файловое хранилище для сессий визуализации
470     VIS_SESSIONS_DIR = os.path.join(tempfile.gettempdir(), "viz_sessions")
471     os.makedirs(VIS_SESSIONS_DIR, exist_ok=True)
472
473     def _save_viz_session(data: Dict) -> str:
474         token = uuid.uuid4().hex
475         path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
476         with open(path, "wb") as f:
477             pickle.dump(data, f)
478         return token
479
480     def _load_viz_session(token: str) -> Dict:
481         path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
482         if not os.path.exists(path):
483             return None
484         with open(path, "rb") as f:
485             return pickle.load(f)
486
487     @app.post("/api/visualize/session")
488     async def create_visualize_session(payload: Dict):
489         signals = payload.get("signals", [])
490         code = payload.get("code", "")
491         if not isinstance(signals, list):
492             raise HTTPException(status_code=400, detail="signals must be a list")
493         token = _save_viz_session({"signals": signals, "code": code})
494         return {"token": token}
495
496     @app.get("/api/visualize/session/{token}")
497     def get_visualize_session(token: str):
498         data = _load_viz_session(token)
499         if not data:
500             raise HTTPException(status_code=404, detail="session not found")
501         return data
502
503
504     # Раздаём фронтенд
505     WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
506     app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")
```

```
507
508 code_signal.py:
509 import re
510 from typing import List, Tuple, Dict
511
512 import numpy as np
513 import pandas as pd
514
515
516 class CodeEvaluationError(Exception):
517     """Ошибка во время вычисления выражения CODE."""
518
519
520 def sanitize_numeric_column(series: pd.Series) -> pd.Series:
521     if series.dtype.kind in ("i", "u", "f"):
522         return series
523     text = series.astype(str).str.replace(", ", ".", regex=False)
524     return pd.to_numeric(text, errors="coerce")
525
526
527 def evaluate_code_expression(code_str: str, df_all: pd.DataFrame) -> Tuple[pd.Series,
List[str]]:
528     if df_all is None or df_all.empty:
529         raise CodeEvaluationError("Нет данных для расчёта синтетического сигнала.")
530     if not code_str or not code_str.strip():
531         raise CodeEvaluationError("Строка CODE пуста.")
532
533     index = df_all.index
534     numeric_df = df_all.apply(sanitize_numeric_column)
535     series_map = {col: numeric_df[col] for col in numeric_df.columns}
536     warnings: List[str] = []
537
538     # ----- обработка «неправильных» имён сигналов -----
539     safe_name_map: Dict[str, str] = {}
540     used_safe_names = set()
541
542     def _make_safe_name(original: str, idx: int) -> str:
543         base = re.sub(r"\W", "_", original)
544         if not base or not re.match(r"[A-Za-z_]", base):
545             base = f"SIG_{idx}"
546         while base in used_safe_names:
547             base += "_"
548         used_safe_names.add(base)
549         return base
550
551     sorted_signals = sorted(series_map.keys(), key=len, reverse=True)
552     for idx, sig_name in enumerate(sorted_signals):
553         safe = _make_safe_name(sig_name, idx)
554         safe_name_map[sig_name] = safe
555
556     def _replace_signal_names(expr: str) -> str:
557         result = []
558         i = 0
559         in_string = False
560         string_char = ""
561
562         while i < len(expr):
563             ch = expr[i]
564             if in_string:
565                 result.append(ch)
566                 if ch == string_char and expr[i - 1] != "\\":
567                     in_string = False
568                     i += 1
569                 continue
570
```

```
571         if ch in ("'", "'"):
572             in_string = True
573             string_char = ch
574             result.append(ch)
575             i += 1
576             continue
577
578         matched = None
579         for name in sorted_signals:
580             if expr.startswith(name, i):
581                 matched = name
582                 break
583         if matched:
584             result.append(safe_name_map[matched])
585             i += len(matched)
586         else:
587             result.append(ch)
588             i += 1
589
590     return "".join(result)
591
592 # ----- вспомогательные функции -----
593 def _ensure_series(value) -> pd.Series:
594     if isinstance(value, pd.Series):
595         return value.reindex(index)
596     if isinstance(value, pd.DataFrame):
597         if value.shape[1] == 1:
598             return value.iloc[:, 0].reindex(index)
599         raise CodeEvaluationError("Невозможно привести DataFrame с несколькими
599 колонками к Series.")
600     if isinstance(value, (list, tuple, np.ndarray)):
601         arr = np.asarray(value, dtype=float)
602         if arr.size == 1:
603             arr = np.full(len(index), arr.item())
604         elif arr.shape[0] != len(index):
605             return pd.Series(np.nan, index=index)
606         return pd.Series(arr, index=index)
607     if value is None or np.isscalar(value):
608         return pd.Series(value, index=index)
609     try:
610         return pd.Series(value, index=index)
611     except Exception as exc:
612         raise CodeEvaluationError(f"Невозможно преобразовать значение '{value}' к
612 Series.") from exc
613
614     def _aggregate_nanfunc(func, args, empty_value=np.nan):
615         if not args:
616             return pd.Series(empty_value, index=index)
617         stacked = np.vstack([_ensure_series(arg).values for arg in args])
618         return pd.Series(func(stacked, axis=0), index=index)
619
620     def GETPOINT(*_):
621         if "GETPOINT" not in warnings:
622             warnings.append("GETPOINT пока не поддержан – возвращается NaN.")
623         return pd.Series(np.nan, index=index)
624
625     def PREV(param_name):
626         if param_name not in series_map:
627             return pd.Series(np.nan, index=index)
628         return series_map[param_name].shift(1)
629
630     def _history_series(param_name):
631         if param_name not in series_map:
632             return None
633         return series_map[param_name]
```

```
634
635     def _history_window(period):
636         try:
637             minutes = int(period)
638         except (TypeError, ValueError):
639             return None
640         if minutes <= 0:
641             return None
642         return f"{minutes}min"
643
644     def _history_apply(param_name, period, fn):
645         s = _history_series(param_name)
646         window = _history_window(period)
647         if s is None or window is None:
648             return pd.Series(np.nan, index=index)
649         return fn(s.rolling(window))
650
651     HISTORYAVG = lambda n, p: _history_apply(n, p, lambda r: r.mean())
652     HISTORYCOUNT = lambda n, p: _history_apply(n, p, lambda r: r.count())
653     HISTORYSUM = lambda n, p: _history_apply(n, p, lambda r: r.sum())
654     HISTORYMAX = lambda n, p: _history_apply(n, p, lambda r: r.max())
655     HISTORYMIN = lambda n, p: _history_apply(n, p, lambda r: r.min())
656     HISTORYDIFF = lambda n, p: _history_apply(n, p, lambda r: r.max() - r.min())
657
658     def HISTORYGRADIENT(param_name, period):
659         s = _history_series(param_name)
660         window = _history_window(period)
661         if s is None or window is None:
662             return pd.Series(np.nan, index=index)
663
664     def slope(window_series: pd.Series):
665         valid = window_series.dropna()
666         if len(valid) < 2:
667             return np.nan
668         x = valid.index.view(np.int64).astype(float) / 1e9
669         y = valid.values.astype(float)
670         x_mean = x.mean()
671         y_mean = y.mean()
672         denom = np.sum((x - x_mean) ** 2)
673         if denom == 0:
674             return np.nan
675         return np.sum((x - x_mean) * (y - y_mean)) / denom
676
677         return s.rolling(window).apply(slope, raw=False)
678
679     def ROUND(a, b=0):
680         a_values = _ensure_series(a).values
681         b_values = _ensure_series(b).values
682         decimals = [
683             0 if np.isnan(dec) else int(round(dec))
684             for dec in b_values
685         ]
686         rounded = np.array([
687             np.round(val, dec) if not np.isnan(val) else np.nan
688             for val, dec in zip(a_values, decimals)
689         ])
690         return pd.Series(rounded, index=index)
691
692     # ----- окружение eval -----
693     env = {
694         "np": np,
695         "ABS": lambda a: pd.Series(np.abs(_ensure_series(a).values), index=index),
696         "POW": lambda a, b: pd.Series(np.power(_ensure_series(a).values,
697             _ensure_series(b).values), index=index),
698         "MIN": lambda *args: _aggregate_nanfunc(np.nanmin, args),
```

```
698     "MAX": lambda *args: _aggregate_nanfunc(np.nanmax, args),
699     "AVG": lambda *args: _aggregate_nanfunc(np.nanmean, args, empty_value=0.0),
700     "MED": lambda *args: _aggregate_nanfunc(np.nanmedian, args),
701     "ROUND": ROUND,
702     "WHEN": lambda cond, t_val, f_val: pd.Series(
703         np.where(_ensure_series(cond).astype(bool).values,
704                  _ensure_series(t_val).values,
705                  _ensure_series(f_val).values),
706                  index=index,
707         ),
708     "PREV": PREV,
709     "HISTORYAVG": HISTORYAVG,
710     "HISTORYCOUNT": HISTORYCOUNT,
711     "HISTORYSUM": HISTORYSUM,
712     "HISTORYMAX": HISTORYMAX,
713     "HISTORYMIN": HISTORYMIN,
714     "HISTORYDIFF": HISTORYDIFF,
715     "HISTORYGRADIENT": HISTORYGRADIENT,
716     "GETPOINT": GETPOINT,
717 }
718
719 for original_name, safe_name in safe_name_map.items():
720     env[safe_name] = series_map[original_name]
721
722 def _normalize_expression(expr: str) -> str:
723     expr = re.sub(r"\bAND\b", "&", expr, flags=re.IGNORECASE)
724     expr = re.sub(r"\bOR\b", "|", expr, flags=re.IGNORECASE)
725     expr = re.sub(r"\bNOT\b", "~", expr, flags=re.IGNORECASE)
726     expr = expr.replace("<>", "!=")
727     expr = re.sub(r"(?<! [<>=!=])=(?! [<>=!=])", "==", expr)
728     return expr
729
730 normalized_code = _normalize_expression(code_str)
731 normalized_code = _replace_signal_names(normalized_code)
732
733 try:
734     raw_result = eval(normalized_code, {"__builtins__": {}}, env)
735 except Exception as exc:
736     raise CodeEvaluationError(str(exc)) from exc
737
738 result_series = _ensure_series(raw_result)
739 result_series.name = result_series.name or "CODE_RESULT"
740 return result_series, warnings
741
742 def compute_code_signal(
743     code_str: str,
744     df_all: pd.DataFrame,
745     warn_callback=lambda msg: None,
746 ) -> pd.Series:
747     """
748     Совместимость с визуализатором: считает синтетический сигнал по CODE
749     и прокидывает предупреждения через колбэк.
750     """
751     series, warnings = evaluate_code_expression(code_str, df_all)
752     for message in warnings:
753         warn_callback(message)
754     return series
755
756     visualizer_app.py:
757         import pandas as pd
758 import requests
759 import streamlit as st
760 import plotly.express as px
761 import numpy as np
762 import plotly.graph_objects as go
```

```
763
764     from code_signal import compute_code_signal, sanitize_numeric_column
765
766     st.set_page_config(page_title="Signal Visualizer", layout="wide")
767     st.title("📊 Визуализация сигналов")
768
769     query_params = st.query_params
770     session_token = query_params.get("session", None)
771     api_url = query_params.get("api_url", "http://localhost:8000")
772
773     signal_codes = query_params.get("signals", [])
774     if isinstance(signal_codes, str):
775         signal_codes = [signal_codes]
776
777     CODE = ""
778     if session_token:
779         try:
780             resp = requests.get(f"{api_url}/api/visualize/session/{session_token}")
781             resp.raise_for_status()
782             payload = resp.json()
783             signal_codes = payload.get("signals", signal_codes)
784             CODE = payload.get("code", CODE)
785         except Exception as e:
786             st.error(f"Не удалось получить данные сессии: {e}")
787
788     if "signals_data" not in st.session_state:
789         st.session_state.signals_data = None
790     if "selected_signals" not in st.session_state:
791         st.session_state.selected_signals = set()
792     if "plot_areas" not in st.session_state:
793         st.session_state.plot_areas = []
794     if "derived_signals" not in st.session_state:
795         st.session_state.derived_signals = {}
796     if "code_signal_name" not in st.session_state:
797         st.session_state.code_signal_name = None
798
799
800     def load_signals(signal_codes_list):
801         if not signal_codes_list:
802             st.info("Список сигналов пуст – ничего загружать.")
803             return None, [], []
804         try:
805             response = requests.post(
806                 f"{api_url}/api/signal-data",
807                 json={"signal_names": signal_codes_list, "format": "json"},
808             )
809             response.raise_for_status()
810             result = response.json()
811             found = result.get("found", [])
812             not_found = result.get("not_found", [])
813             data_dict = result.get("data", {})
814
815             if not data_dict:
816                 st.warning("Нет данных по запрошенным сигналам.")
817                 return None, found, not_found
818
819             frames = []
820             for sig, records in data_dict.items():
821                 if not records:
822                     continue
823                     df = pd.DataFrame(records)
824                     if "datetime" not in df or "value" not in df:
825                         continue
826                     df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
827                     df = df.dropna(subset=["datetime"])
828
829             st.session_state.signals_data = frames
830             st.session_state.selected_signals = set([sig for sig, _ in data_dict])
831             st.session_state.plot_areas = []
832             st.session_state.derived_signals = {}
833             st.session_state.code_signal_name = None
834
835             st.info("Сигналы загружены!")
836
837             return frames, found, not_found
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
```

```
828         df = df.set_index("datetime").sort_index()
829         df = df.rename(columns={"value": sig})
830         frames.append(df[[sig]])
831
832     if not frames:
833         return None, found, not_found
834     return pd.concat(frames, axis=1).sort_index(), found, not_found
835
836 except Exception as exc:
837     st.error(f"X Ошибка загрузки данных: {exc}")
838     return None, [], []
839
840
841 def get_all_signals_df(exclude: set[str] | None = None):
842     exclude = exclude or set()
843     base = st.session_state.signals_data
844     derived = st.session_state.derived_signals
845
846     dfs = []
847     if base is not None:
848         dfs.append(base)
849     for name, ddf in derived.items():
850         if name in exclude:
851             continue
852         dfs.append(ddf)
853
854     if not dfs:
855         return None
856     return pd.concat(dfs, axis=1).sort_index()
857
858
859 def compute_stats_numeric(df: pd.DataFrame) -> pd.DataFrame:
860     if df is None or df.empty:
861         return pd.DataFrame()
862
863     numeric = df.apply(sanitize_numeric_column)
864     valid_cols = [col for col in numeric.columns if numeric[col].count() > 0]
865     if not valid_cols:
866         return pd.DataFrame()
867
868     numeric = numeric[valid_cols]
869     stats = pd.DataFrame(index=numeric.columns)
870     stats["count"] = numeric.count()
871     stats["min"] = numeric.min()
872     stats["max"] = numeric.max()
873     stats["mean"] = numeric.mean()
874     stats["std"] = numeric.std()
875     stats["median"] = numeric.median()
876
877     starts, ends = [], []
878     for col in numeric.columns:
879         series = numeric[col].dropna()
880         starts.append(series.index.min() if not series.empty else pd.NaT)
881         ends.append(series.index.max() if not series.empty else pd.NaT)
882
883     stats["start"] = starts
884     stats["end"] = ends
885     return stats
886
887
888 def make_unique_name(base_name: str) -> str:
889     existing = set()
890     if st.session_state.signals_data is not None:
891         existing |= set(st.session_state.signals_data.columns)
892         existing |= set(st.session_state.derived_signals.keys())
```

```
893     if base_name not in existing:
894         return base_name
895     idx = 2
896     while f"{base_name}_{idx}" in existing:
897         idx += 1
898     return f"{base_name}_{idx}"
899
900
901 if signal_codes and st.session_state.signals_data is None:
902     with st.spinner("Загружаем данные сигналов..."):
903         df_base, found_codes, not_found_codes = load_signals(signal_codes)
904         st.session_state.signals_data = df_base
905         st.success(f"✅ Загружено сигналов: {len(found_codes)}")
906     if not_found_codes:
907         st.warning(f"⚠️ Не найдены: {', '.join(not_found_codes)}")
908
909 # --- синтетический сигнал из CODE (считаем один раз, потом не пересчитываем) ---
910 code_signal_name = st.session_state.code_signal_name
911 df_for_code = get_all_signals_df(exclude={code_signal_name} if code_signal_name else
None)
912
913 # Ключ "какой CODE мы уже считали" (можно оставить просто CODE; session_token добавил
на всякий)
914 code_key = (session_token, CODE)
915
916 already_have_series = (
917     st.session_state.code_signal_name is not None
918     and st.session_state.code_signal_name in st.session_state.derived_signals
919 )
920
921 if CODE and df_for_code is not None:
922     need_recalc = (st.session_state.get("code_key") != code_key) or (not
already_have_series)
923
924     if need_recalc:
925         try:
926             synthetic_series = compute_code_signal(
927                 CODE,
928                 df_for_code,
929                 warn_callback=lambda msg: st.warning(msg, icon="⚠️"),
930             )
931             target_name = code_signal_name or make_unique_name("CODE_RESULT")
932             synthetic_series.name = target_name
933
934             st.session_state.derived_signals[target_name] = pd.DataFrame({target_name:
synthetic_series})
935             st.session_state.code_signal_name = target_name
936             st.session_state.selected_signals.add(target_name)
937
938             st.session_state.code_key = code_key
939             st.success(f"Синтетический сигнал обновлён: {target_name}")
940         except Exception as exc:
941             st.warning(f"Не удалось вычислить CODE: {exc}")
942
943 elif not CODE:
944     # если CODE исчез – удаляем синтетический сигнал и сбрасываем ключ
945     if code_signal_name:
946         st.session_state.derived_signals.pop(code_signal_name, None)
947         st.session_state.selected_signals.discard(code_signal_name)
948         st.session_state.code_signal_name = None
949         st.session_state.code_key = None
950
951 # --- итоговый DataFrame со всеми сигналами ---
952 df_all_signals = get_all_signals_df()
953
```

```
954 with st.sidebar:
955     st.header("Выбор сигналов")
956
957     if df_all_signals is not None:
958         available_signals = df_all_signals.columns.tolist()
959         for signal in available_signals:
960             checked = st.checkbox(
961                 signal,
962                 value=(signal in st.session_state.selected_signals),
963             )
964             if checked:
965                 st.session_state.selected_signals.add(signal)
966             else:
967                 st.session_state.selected_signals.discard(signal)
968
969             st.divider()
970             st.subheader("Создать обрезанный сигнал")
971
972             base_df = st.session_state.signals_data
973             if base_df is not None and not base_df.empty:
974                 base_choice = st.selectbox("Исходный сигнал", base_df.columns)
975                 series = base_df[base_choice].dropna()
976                 if not series.empty:
977                     col1, col2 = st.columns(2)
978                     with col1:
979                         start_date = st.date_input(
980                             "Начало",
981                             value=series.index.min().date(),
982                         )
983                     with col2:
984                         end_date = st.date_input(
985                             "Конец",
986                             value=series.index.max().date(),
987                         )
988
989                     start_ts = pd.Timestamp(start_date)
990                     end_ts = pd.Timestamp(end_date) + pd.Timedelta(days=1) - pd.Timedelta(
991                         microseconds=1
992                     )
993
994                     default_name = f"{base_choice}__{start_ts.date()}_{end_ts.date()}"
995                     new_name = st.text_input("Имя нового сигнала", value=default_name)
996
997                     col3, col4 = st.columns(2)
998                     if col3.button("Создать"):
999                         name_unique = make_unique_name(new_name.strip())
1000                         cut_series = series[(series.index >= start_ts) & (series.index <=
end_ts)]
1001                         if cut_series.empty:
1002                             st.warning("В выбранном диапазоне нет точек.")
1003                         else:
1004                             st.session_state.derived_signals[name_unique] = pd.DataFrame(
1005                                 {name_unique: cut_series}
1006                             )
1007                             st.success(f"Создан обрезанный сигнал: {name_unique}")
1008                             st.rerun()
1009                     if col4.button("Очистить все обрезанные"):
1010                         st.session_state.derived_signals = {
1011                             k: v
1012                             for k, v in st.session_state.derived_signals.items()
1013                             if k == st.session_state.code_signal_name
1014                         }
1015                         st.session_state.selected_signals = {
1016                             sig
1017                             for sig in st.session_state.selected_signals
```

```
1018         if (st.session_state.signals_data is not None and sig in
1019             st.session_state.signals_data.columns)
1020                 or sig == st.session_state.code_signal_name
1021             }
1022             st.rerun()
1023
1024     if st.session_state.derived_signals:
1025         st.subheader("Удалить обрезанный/синтетический сигнал")
1026         derived_names = [name for name in st.session_state.derived_signals.keys()]
1027         delete_candidate = st.selectbox("Выберите", ["-"] + derived_names)
1028         if st.button("Удалить выбранный") and delete_candidate != "-":
1029             st.session_state.derived_signals.pop(delete_candidate, None)
1030             st.session_state.selected_signals.discard(delete_candidate)
1031             if delete_candidate == st.session_state.code_signal_name:
1032                 st.session_state.code_signal_name = None
1033             st.rerun()
1034
1035     st.divider()
1036     st.subheader("Области построения")
1037     col_a, col_b = st.columns(2)
1038     if col_a.button("Добавить график"):
1039         new_id = max([area.get("id", 0) for area in st.session_state.plot_areas] +
1040 [0]) + 1
1041         st.session_state.plot_areas.append({"id": new_id, "signals": []})
1042         st.rerun()
1043     if col_b.button("Очистить все"):
1044         st.session_state.plot_areas = []
1045         st.session_state.selected_signals = set()
1046         st.rerun()
1047     else:
1048         st.info("⚠️ Данные сигналов еще не загружены.")
1049
1050 if df_all_signals is not None and st.session_state.selected_signals:
1051     if not st.session_state.plot_areas:
1052         st.session_state.plot_areas.append(
1053             {"id": 1, "signals": list(st.session_state.selected_signals)})
1054
1055     for i, plot_area in enumerate(st.session_state.plot_areas):
1056         with st.container():
1057             col1, col2 = st.columns([3, 1])
1058             with col1:
1059                 st.subheader(f"График #{plot_area['id']}")  

1060                 if st.button("Удалить", key=f"remove_area_{i}"):
1061                     st.session_state.plot_areas.pop(i)
1062                     st.rerun()
1063
1064                 selected = st.multiselect(
1065                     "Выберите сигнал(ы):",
1066                     list(st.session_state.selected_signals),
1067                     default=plot_area.get("signals", []),
1068                     key=f"signals_sel_{i}",
1069                 )
1070                 st.session_state.plot_areas[i]["signals"] = selected
1071
1072                 if selected:
1073                     df_plot = df_all_signals[selected].copy()
1074
1075                     # Для графика приводим к числам (поддержка запятых)
1076                     df_plot_num = df_plot.apply(sanitize_numeric_column)
1077
1078                     valid_index = df_plot_num.dropna(how="all").index
1079                     if len(valid_index) == 0:
1080                         st.warning("Нет числовых данных для выбранных сигналов.")
```

```
1081
1082     else:
1083         ts_idx = st.slider(
1084             "Вертикальная линия (время)",
1085             min_value=0,
1086             max_value=len(valid_index) - 1,
1087             value=len(valid_index) - 1,
1088             key=f"vline_{i}",
1089         )
1090         ts = valid_index[ts_idx]
1091
1092     # график с вертикальной линией
1093     fig = px.line(
1094         df_plot_num,
1095         x=df_plot_num.index,
1096         y=selected,
1097         title=f"График #{plot_area['id']}",
1098         render_mode="webgl"
1099     )
1100     fig.add_vline(x=ts, line_width=2, line_dash="dash",
1101     line_color="red")
1102
1103     fig.update_layout(
1104         uirevision=f"plot_area_{plot_area['id']}",
1105         height=650,
1106         legend_title_text="Сигналы",
1107         xaxis_title="Время",
1108         yaxis_title="Значение",
1109         margin=dict(l=20, r=20, t=40, b=20),
1110     )
1111     st.plotly_chart(fig, use_container_width=True)
1112
1113     # значения на линии
1114     nearest =
1115         df_plot_num.reindex(df_plot_num.index.union([ts])).sort_index()
1116         nearest = nearest.ffill().loc[ts]
1117
1118         # статистика + колонка значений на линии
1119         st.markdown("**📊 Статистика (по всему сигналу):**")
1120         stats_df = compute_stats_numeric(df_plot)
1121         if stats_df.empty:
1122             st.info("Нет числовых данных для расчёта статистики.")
1123         else:
1124             stats_view = stats_df.copy()
1125             stats_view["value"] = nearest.reindex(stats_view.index)
1126             stats_view["start"] = (
1127                 pd.to_datetime(stats_view["start"], errors="coerce")
1128                 .dt.strftime("%Y-%m-%d %H:%M:%S")
1129             )
1130             stats_view["end"] = (
1131                 pd.to_datetime(stats_view["end"], errors="coerce")
1132                 .dt.strftime("%Y-%m-%d %H:%M:%S")
1133             )
1134             st.dataframe(
1135                 stats_view.style.format(
1136                     {
1137                         "count": "{:.0f}",
1138                         "min": "{:.6g}",
1139                         "max": "{:.6g}",
1140                         "mean": "{:.6g}",
1141                         "std": "{:.6g}",
1142                         "median": "{:.6g}",
1143                         "value_at_line": "{:.6g}",
1144                     },
1145                     na_rep="",
1146                 ),
1147                 use_container_width=True,
```

```
1144                     )
1145             else:
1146                 st.info("Выберите сигналы для отображения.")
1147             st.divider()
1148
1149 elif df_all_signals is None:
1150     st.info("⚠️ Данные сигналов ещё не загружены.")
1151 else:
1152     st.info("👉 Выберите сигналы слева для визуализации.")
1153
1154 if df_all_signals is not None:
1155     with st.expander("ℹ️ Информация о данных"):
1156         col1, col2, col3 = st.columns(3)
1157         with col1:
1158             st.metric("Всего сигналов (вкл. обрезанные/синтет.)",
1159             len(df_all_signals.columns))
1160             with col2:
1161                 st.metric("Количество записей", len(df_all_signals))
1162             with col3:
1163                 try:
1164                     dt_range = df_all_signals.index.max() - df_all_signals.index.min()
1165                     st.metric("Диапазон времени", str(dt_range).split(".")[0])
1166                 except Exception:
1167                     st.metric("Диапазон времени", "-")
1168
1169 if CODE:
1170     with st.expander("✳️ Сгенерированный код (оригинал)"):
1171         st.code(CODE, language="text")
1172
1173 index.html:
1174 <!DOCTYPE html>
1175 <html lang="ru">
1176     <head>
1177         <meta charset="UTF-8">
1178         <meta name="viewport" content="width=device-width, initial-scale=1.0">
1179         <title>Редактор логических схем</title>
1180         <link rel="stylesheet" href="css/styles.css">
1181     </head>
1182     <body>
1183         <div id="app">
1184             <div id="menu">
1185                 <button class="menu-btn" id="btn-new">📝 Новый</button>
1186                 <button class="menu-btn" id="btn-save">💾 Сохранить</button>
1187                 <button class="menu-btn" id="btn-load">📁 Загрузить</button>
1188                 <button class="menu-btn" id="btn-generate-code">✳️ Код</button>
1189                 <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</button>
1190             <button class="menu-btn" id="btn-visualize">📈 Визуализировать</button>
1191             <div class="menu-separator"></div>
1192             <div class="zoom-controls">
1193                 <button class="menu-btn zoom-btn" id="btn-zoom-out">-</button>
1194                 <span id="zoom-level">100%</span>
1195                 <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
1196                 <button class="menu-btn" id="btn-zoom-fit">➡ Вписать</button>
1197                 <button class="menu-btn" id="btn-zoom-reset">1:1</button>
1198             </div>
1199             <input type="file" id="file-input" accept=".json">
1200         </div>
1201
1202         <div id="main">
1203             <div id="palette">
1204                 <h3>📦 Элементы</h3>
1205                 <div class="palette-section">
1206                     <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>
```

```
1207             <div class="palette-item" data-type="group">
1208                 <svg viewBox="0 0 60 40">
1209                     <rect x="6" y="8" width="48" height="24" rx="4"
1210                         fill="none" stroke="#6b7280" stroke-width="2" stroke-
1211                         dasharray="4,2"/>
1212                     <text x="14" y="25" fill="#6b7280" font-size="10" font-
1213                         weight="bold">GROUP</text>
1214                     </svg>
1215                     <div class="palette-item-name">Группа</div>
1216                 </div>
1217             </div>
1218
1219             <div class="palette-section">
1220                 <div class="palette-section-title">ВХОДЫ</div>
1221
1222                 <div class="palette-item" data-type="input-signal">
1223                     <svg viewBox="0 0 60 40">
1224                         <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460"
1225                         stroke="#4a90d9" stroke-width="2"/>
1226                     <text x="12" y="24" fill="#eee" font-size="10">IN</text>
1227                     </svg>
1228                     <div class="palette-item-name">Входной сигнал</div>
1229                 </div>
1230             </div>
1231             <div class="palette-section">
1232                 <div class="palette-section-title">ВыХОДЫ</div>
1233
1234                 <div class="palette-item" data-type="output">
1235                     <svg viewBox="0 0 60 40">
1236                         <rect x="5" y="5" width="50" height="30" rx="6"
1237                         fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
1238                     <text x="12" y="24" fill="#10b981" font-size="9">ВыХОД</
1239                     text>
1240                     </svg>
1241                     <div class="palette-item-name">Выход</div>
1242                 </div>
1243             </div>
1244
1245             <div class="palette-section">
1246                 <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>
1247
1248                 <div class="palette-item" data-type="and">
1249                     <svg viewBox="0 0 60 40">
1250                         <rect x="5" y="5" width="50" height="30" rx="5"
1251                         fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
1252                     <text x="22" y="25" fill="#eee" font-size="12" font-
1253                         weight="bold">И</text>
1254                     </svg>
1255                     <div class="palette-item-name">И (AND)</div>
1256                 </div>
1257
1258                 <div class="palette-item" data-type="or">
1259                     <svg viewBox="0 0 60 40">
1260                         <rect x="5" y="5" width="50" height="30" rx="5"
1261                         fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
1262                     <text x="12" y="25" fill="#eee" font-size="11" font-
1263                         weight="bold">ИЛИ</text>
1264                     </svg>
1265                     <div class="palette-item-name">ИЛИ (OR)</div>
1266                 </div>
1267
1268                 <div class="palette-item" data-type="not">
1269                     <svg viewBox="0 0 60 40">
1270                         <rect x="5" y="5" width="50" height="30" rx="5"
1271                         fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
```

```
1262                         <text x="12" y="25" fill="#eee" font-size="11" font-
1263             weight="bold">НЕТ</text>
1264                     </svg>
1265                     <div class="palette-item-name">НЕТ (NOT)</div>
1266                 </div>
1267             </div>
1268             <div class="palette-section">
1269                 <div class="palette-section-title">СРАВНЕНИЕ</div>
1270
1271                 <div class="palette-item" data-type="if">
1272                     <svg viewBox="0 0 60 40">
1273                         <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460" stroke="#e94560" stroke-width="2"/>
1274                     <text x="14" y="24" fill="#eee" font-size="9" font-
1275             weight="bold">ЕСЛИ</text>
1276                     </svg>
1277                     <div class="palette-item-name">ЕСЛИ (IF)</div>
1278                 </div>
1279             </div>
1280             <div class="palette-section">
1281                 <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>
1282
1283                 <div class="palette-item" data-type="separator">
1284                     <svg viewBox="0 0 60 40">
1285                         <rect x="5" y="8" width="50" height="24" rx="3" fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
1286                         <text x="8" y="25" fill="#f59e0b" font-size="10" font-
1287             weight="bold">/></text>
1288                     </svg>
1289                     <div class="palette-item-name">Сепаратор</div>
1290                 </div>
1291             </div>
1292             <div class="palette-section">
1293                 <div class="palette-section-title">ЗНАЧЕНИЯ</div>
1294
1295                 <div class="palette-item" data-type="const">
1296                     <svg viewBox="0 0 60 40">
1297                         <rect x="10" y="8" width="40" height="24" rx="3" fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
1298                         <text x="24" y="25" fill="#3b82f6" font-size="14" font-
1299             weight="bold">C</text>
1300                     </svg>
1301                     <div class="palette-item-name">Константа</div>
1302                 </div>
1303
1304                 <div class="palette-item" data-type="formula">
1305                     <svg viewBox="0 0 60 40">
1306                         <rect x="5" y="5" width="50" height="30" rx="5" fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
1307                         <text x="12" y="25" fill="#f59e0b" font-size="11" font-
1308             weight="bold">f(x)</text>
1309                     </svg>
1310                     <div class="palette-item-name">Формула</div>
1311                 </div>
1312
1313                 <div class="type-legends">
1314                     <div class="type-legends-item">
1315                         <div class="type-legends-dot logic"></div>
1316                         <span>Логический</span>
1317                     </div>
1318                 </div>
```

```
1318                     <div class="type-legend-dot number"></div>
1319                     <span>Числовой</span>
1320                 </div>
1321             </div>
1322
1323
1324         <div id="workspace-container">
1325             <svg id="connections-svg"></svg>
1326             <div id="workspace"></div>
1327             <!-- Прямоугольник для выделения элементов --&gt;
1328             &lt;div id="selection-rect"&gt;&lt;/div&gt;
1329
1330             <!-- Мини-карта --&gt;
1331             &lt;div id="minimap"&gt;
1332                 &lt;div id="minimap-viewport"&gt;&lt;/div&gt;
1333                 &lt;canvas id="minimap-canvas"&gt;&lt;/canvas&gt;
1334             &lt;/div&gt;
1335
1336             <!-- Координаты и информация --&gt;
1337             &lt;div id="viewport-info"&gt;
1338                 &lt;span id="cursor-pos"&gt;X: 0, Y: 0&lt;/span&gt;
1339                 &lt;span id="selection-info"&gt;&lt;/span&gt;
1340             &lt;/div&gt;
1341         &lt;/div&gt;
1342     &lt;/div&gt;
1343 &lt;/div&gt;
1344
1345     <!-- Модальные окна --&gt;
1346     &lt;div id="modal-overlay"&gt;
1347         &lt;div id="modal"&gt;
1348             &lt;h3 id="modal-title"&gt;Свойства элемента&lt;/h3&gt;
1349             &lt;div id="modal-content"&gt;&lt;/div&gt;
1350             &lt;div class="modal-buttons"&gt;
1351                 &lt;button class="modal-btn cancel" id="modal-cancel"&gt;Отмена&lt;/button&gt;
1352                 &lt;button class="modal-btn save" id="modal-save"&gt;Сохранить&lt;/button&gt;
1353             &lt;/div&gt;
1354         &lt;/div&gt;
1355     &lt;/div&gt;
1356
1357     <!-- Модальное окно свойств проекта --&gt;
1358     &lt;div id="project-modal-overlay" class="modal-overlay-class"&gt;
1359         &lt;div id="project-modal" class="modal-class"&gt;
1360             &lt;h3&gt;Свойства проекта&lt;/h3&gt;
1361             &lt;div id="project-modal-content"&gt;&lt;/div&gt;
1362             &lt;div class="modal-buttons"&gt;
1363                 &lt;button class="modal-btn cancel" id="project-modal-cancel"&gt;Отмена&lt;/
1364 button&gt;
1365                 &lt;button class="modal-btn save" id="project-modal-save"&gt;Сохранить&lt;/
1366 button&gt;
1367             &lt;/div&gt;
1368
1369         &lt;div id="code-modal-overlay" class="modal-overlay-class"&gt;
1370             &lt;div id="code-modal" class="modal-class"&gt;
1371                 &lt;h3&gt;Сгенерированный код&lt;/h3&gt;
1372                 &lt;textarea id="code-output" style="width:100%; height:300px;"&gt;&lt;/textarea&gt;
1373                 &lt;div class="modal-buttons"&gt;
1374                     &lt;button class="modal-btn cancel" id="code-modal-close"&gt;Закрыть&lt;/
1375 button&gt;
1376                     &lt;/div&gt;
1377                 &lt;/div&gt;
1378
1379         &lt;div id="context-menu"&gt;</pre>
```

```
1380         <div class="context-item" id="ctx-properties"> Свойства</div>
1381         <div class="context-item" id="ctx-delete"> Удалить</div>
1382     </div>
1383
1384     <!-- Модули JavaScript -->
1385     <!-- Модули JavaScript -->
1386     <script src="js/config.js"></script>
1387     <script src="js/state.js"></script>
1388     <script src="js/utils.js"></script>
1389     <script src="js/viewport.js"></script>
1390     <script src="js/elements.js"></script>
1391     <script src="js/connections.js"></script>
1392     <script src="js/outputs.js"></script> <!-- ← Этот файл опционален теперь -->
1393     <script src="js/modal.js"></script>
1394     <script src="js/project.js"></script>
1395     <script src="js/codegen_graph.js"></script>
1396     <script src="js/codegen_optimizer.js"></script>
1397     <script src="js/codegen.js"></script>
1398     <script src="js/settings.js"></script>
1399
1400     <script src="js/app.js"></script>
1401
1402     <div id="modal-project-list" class="modal hidden">
1403         <div class="modal__content modal__content--wide">
1404             <h2 class="modal__title">Выбор проекта</h2>
1405
1406             <div class="project-list__toolbar">
1407                 <input id="project-search" type="text" placeholder="Фильтр по имени или
описанию..." />
1408                 <button id="project-refresh" class="btn btn-secondary">Обновить</button>
1409             </div>
1410
1411             <div class="project-list__table-container">
1412                 <table class="project-list__table">
1413                     <thead>
1414                         <tr>
1415                             <th>Файл</th>
1416                             <th>Tagname</th>
1417                             <th>Description</th>
1418                             <th>Тип</th>
1419                         </tr>
1420                     </thead>
1421                     <tbody id="project-list-body">
1422                         <tr><td colspan="4" class="project-list__empty">Загрузка...</td></tr>
1423                     </tbody>
1424                 </table>
1425             </div>
1426
1427             <div class="modal__actions">
1428                 <button id="project-cancel" class="btn btn-secondary">Отмена</button>
1429                 <button id="project-load" class="btn btn-primary" disabled>Загрузить</
button>
1430                 </div>
1431             </div>
1432         </div>
1433     </body>
1434     </html>
1435
1436
1437     app.js:
1438     /**
1439      * Главный модуль приложения
1440      */
1441
1442     const App = {
```

```
1443     /**
1444      * Инициализация приложения
1445      */
1446     init() {
1447         Settings.init().catch(console.error);
1448         //Settings.init().then(() => {
1449         //    // если хочешь – можно обновить UI (например, статус “Сигналы
загружены”)
1450         //    console.log('Settings loaded, signals:', Settings.signals.length);
1451         //}).catch(err => console.error(err));
1452         //console.log('signals loaded:', Settings.signals.slice(0, 5));
1453         this.setupPaletteDragDrop();
1454         this.setupGlobalMouseHandlers();
1455         this.setupContextMenu();
1456         this.setupWorkspaceClick();
1457         this.setupOutputCounter();
1458         this.setupMultiSelection();
1459
1460         // Инициализация модулей
1461         Viewport.init();
1462         Modal.init();
1463         Project.init();
1464
1465         // Первоначальное определение выходов (только если модуль загружен)
1466         if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1467             Outputs.updateOutputStatus();
1468         }
1469
1470         console.log('Logic Scheme Editor initialized');
1471         document.getElementById('btn-generate-code').addEventListener('click', () => {
1472             const code = CodeGen.generate();
1473             document.getElementById('code-output').value = code;
1474             document.getElementById('code-modal-overlay').style.display = 'flex';
1475         });
1476
1477         document.getElementById('code-modal-close').addEventListener('click', () => {
1478             document.getElementById('code-modal-overlay').style.display = 'none';
1479         });
1480         document.getElementById('btn-visualize').addEventListener('click', () => {
1481             App.openSignalVisualizer();
1482         });
1483     },
1484
1485     openSignalVisualizer() {
1486         try {
1487             // 1) Собираем входные сигналы
1488             const signals = Object.values(AppState.elements)
1489             .filter(e => e && e.type === 'input-signal')
1490             .map(e => e.props?.name || e.id);
1491             const uniqSignals = [...new Set(signals)];
1492             if (uniqSignals.length === 0) {
1493                 alert('Нет входных сигналов в схеме.');
1494                 return;
1495             }
1496
1497             // 2) Генерируем код (может быть длинным)
1498             let codeStr = '';
1499             if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate ===
'function') {
1500                 codeStr = CodeGen.generate() || '';
1501             }
1502
1503             // 3) Создаём сессию на backend, чтобы не тащить код в URL
1504             fetch('/api/visualize/session', {
1505                 method: 'POST',

```

```
1506     headers: { 'Content-Type': 'application/json' },
1507     body: JSON.stringify({ signals: uniqSignals, code: codeStr })
1508   })
1509   .then(r => {
1510     if (!r.ok) throw new Error('Failed to create visualize session');
1511     return r.json();
1512   })
1513   .then(data => {
1514     const token = data.token;
1515     const apiUrl = window.location.origin; // http://localhost:8000
1516     const params = new URLSearchParams();
1517     params.set('session', token);
1518     params.set('api_url', apiUrl);
1519     // signals можно не передавать – визуализатор возьмет их из session
1520     const visualizerUrl = `http://localhost:8501?${params.toString()}`;
1521     window.open(visualizerUrl, '_blank');
1522   })
1523   .catch(err => {
1524     console.error(err);
1525     alert('Не удалось открыть визуализатор: ' + err.message);
1526   });
1527
1528 } catch (e) {
1529   console.error(e);
1530   alert('Ошибка при подготовке визуализации: ' + e.message);
1531 }
1532 },
1533 /**
1534 * Отмена состояния drag из палитры (helper)
1535 */
1536 cancelPaletteDrag() {
1537   if (AppState.dragPreview) {
1538     try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
1539     AppState.dragPreview = null;
1540   }
1541   AppState.isDraggingFromPalette = false;
1542   AppState.dragType = null;
1543 },
1544
1545 /**
1546 * Настройка счётчика выходов в меню
1547 */
1548 setupOutputCounter() {
1549   // Не создавать повторно, если уже есть
1550   if (document.getElementById('btn-outputs')) return;
1551
1552   const menu = document.getElementById('menu');
1553
1554   // Создаём кнопку с счётчиком выходов
1555   const outputBtn = document.createElement('button');
1556   outputBtn.className = 'menu-btn output-btn';
1557   outputBtn.id = 'btn-outputs';
1558   outputBtn.innerHTML =
1559     `  Выходы
1560     <span id="output-counter" class="output-counter">0</span>
1561   `;
1562
1563   // Вставляем после кнопки свойств проекта
1564   const projectBtn = document.getElementById('btn-project-settings');
1565   if (projectBtn) {
1566     projectBtn.after(outputBtn);
1567   } else {
1568     menu.appendChild(outputBtn);
1569   }
1570 }
```

```
1571
1572     outputBtn.addEventListener('click', () => {
1573         Modal.showProjectPropertiesModal();
1574     });
1575
1576
1577 /**
1578 * Настройка drag & drop из палитры
1579 */
1580 setupPaletteDragDrop() {
1581     document.querySelectorAll('.palette-item').forEach(item => {
1582         item.addEventListener('mousedown', (e) => {
1583             // Только левая кнопка мыши должна запускать drag из палитры
1584             if (e.button !== 0) return;
1585             e.preventDefault();
1586
1587             AppState.isDraggingFromPalette = true;
1588             AppState.dragType = item.dataset.type;
1589
1590             AppState.dragPreview = document.createElement('div');
1591             AppState.dragPreview.className = 'drag-preview';
1592             AppState.dragPreview.textContent =
1593                 ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
1594             AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1595             AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1596             document.body.appendChild(AppState.dragPreview);
1597         });
1598     },
1599
1600 /**
1601 * Глобальные обработчики мыши
1602 */
1603 /**
1604 * Глобальные обработчики мыши
1605 */
1606 setupGlobalMouseHandlers() {
1607     document.addEventListener('mousemove', (e) => {
1608         if (AppState.isDraggingFromPalette && AppState.dragPreview) {
1609             AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1610             AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1611         }
1612         if (AppState.resizing) {
1613             Elements.handleResize(e);
1614             return;
1615         }
1616         if (AppState.draggingElement) {
1617             Elements.handleDrag(e);
1618         }
1619         if (AppState.tempLine && AppState.connectingFrom) {
1620             Connections.drawTempConnection(e);
1621         }
1622     });
1623
1624     document.addEventListener('mouseup', (e) => {
1625         if (AppState.resizing) {
1626             AppState.resizing = null;
1627             if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1628         }
1629
1630         if (AppState.isDraggingFromPalette) {
1631             try {
1632                 if (AppState.dragPreview) {
1633                     AppState.dragPreview.remove();
1634                     AppState.dragPreview = null;
```

```
1635 }
1636
1637     const container = document.getElementById('workspace-container');
1638     const rect = container.getBoundingClientRect();
1639
1640     if (e.clientX >= rect.left && e.clientX <= rect.right &&
1641         e.clientY >= rect.top && e.clientY <= rect.bottom) {
1642
1643         const canvasPos = screenToCanvas(e.clientX, e.clientY);
1644         const config = ELEMENT_TYPES[AppState.dragType];
1645         if (config) {
1646             const defaultWidth = config.minWidth || 120;
1647             const defaultHeight = config.minHeight || 60;
1648
1649             // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
1650             // обработать
1651             const newElement = Elements.addElement(
1652                 AppState.dragType,
1653                 canvasPos.x - defaultWidth / 2,
1654                 canvasPos.y - defaultHeight / 2
1655             );
1656
1657             if (newElement && typeof Outputs !== 'undefined') {
1658                 Outputs.updateOutputStatus();
1659             } else {
1660                 console.error('Неизвестный тип элемента при drop:',
1661                             AppState.dragType);
1662             }
1663         } finally {
1664             App.cancelPaletteDrag();
1665         }
1666     }
1667
1668     if (AppState.draggingElement) {
1669         AppState.draggingElement = null;
1670     }
1671
1672     Connections.clearConnectionState();
1673 });
1674
1675 document.addEventListener('keydown', (e) => {
1676     if (e.key === 'Delete' && AppState.selectedElement) {
1677         Elements.deleteElement(AppState.selectedElement);
1678         if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1679     }
1680     if (e.key === 'Escape') {
1681         Elements.deselectAll();
1682         Connections.clearConnectionState();
1683         if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
1684     }
1685 });
1686 },
1687 /**
1688 * Настройка контекстного меню
1689 */
1690 setupContextMenu() {
1691     document.addEventListener('click', (e) => {
1692         const menu = document.getElementById('context-menu');
1693         if (!menu.contains(e.target)) {
1694             menu.style.display = 'none';
1695         }
1696     });
1697 }
```

```
1698
1699     document.getElementById('ctx-properties').addEventListener('click', () => {
1700         const elemId = document.getElementById('context-menu').dataset.elementId;
1701         document.getElementById('context-menu').style.display = 'none';
1702         const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
1703         if (config?.hasProperties) {
1704             Modal.showPropertiesModal(elemId);
1705         }
1706     });
1707
1708     document.getElementById('ctx-delete').addEventListener('click', () => {
1709         const elemId = document.getElementById('context-menu').dataset.elementId;
1710         document.getElementById('context-menu').style.display = 'none';
1711         Elements.deleteElement(elemId);
1712         // Обновляем выходы только если модуль загружен
1713         if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1714             Outputs.updateOutputStatus();
1715         }
1716     });
1717 },
1718 /**
1719 * Клик по рабочей области
1720 */
1721 setupWorkspaceClick() {
1722     const workspace = document.getElementById('workspace');

1723     workspace.addEventListener('click', (e) => {
1724         if (e.target === workspace) {
1725             Elements.deselectAll();
1726         }
1727     });
1728 },
1729 /**
1730 * --- Выделение рамкой и множественное перемещение ---
1731 */
1732 setupMultiSelection() {
1733     const container = document.getElementById('workspace-container');
1734     const rectEl = document.getElementById('selection-rect');

1735     container.addEventListener('mousedown', (e) => {
1736         if (e.button !== 0) return;
1737         if (e.target !== document.getElementById('workspace')) return;

1738         const pos = screenToCanvas(e.clientX, e.clientY);
1739         AppState.multiSelecting = true;
1740         AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
1741 pos.y, w: 0, h: 0 };

1742         rectEl.style.left = e.clientX + 'px';
1743         rectEl.style.top = e.clientY + 'px';
1744         rectEl.style.width = '0px';
1745         rectEl.style.height = '0px';
1746         rectEl.style.display = 'block';
1747     });

1748     document.addEventListener('mousemove', (e) => {
1749         if (!AppState.multiSelecting) return;

1750         const pos = screenToCanvas(e.clientX, e.clientY);
1751         const sx = AppState.selectionRect.startX;
1752         const sy = AppState.selectionRect.startY;
1753         const x = Math.min(sx, pos.x);
1754         const y = Math.min(sy, pos.y);
1755         const w = Math.abs(pos.x - sx);
```

```
1762         const h = Math.abs(pos.y - sy);
1763
1764         rectEl.style.left = x * AppState.viewport.zoom + AppState.viewport.panX +
1765         'px';
1766         rectEl.style.top = y * AppState.viewport.zoom + AppState.viewport.panY +
1767         'px';
1768         rectEl.style.width = w * AppState.viewport.zoom + 'px';
1769         rectEl.style.height = h * AppState.viewport.zoom + 'px';
1770
1771         const selected = [];
1772         for (const [id, elData] of Object.entries(AppState.elements)) {
1773             if (!elData || elData.type === 'output-frame') continue;
1774             if (
1775                 elData.x >= x && elData.x + elData.width <= x + w &&
1776                 elData.y >= y && elData.y + elData.height <= y + h
1777             ) selected.push(id);
1778         }
1779
1780         AppState.selectedElements = selected;
1781         document.querySelectorAll('.element').forEach(el =>
1782             el.classList.toggle('selected', selected.includes(el.id))
1783         );
1784     });
1785
1786     document.addEventListener('mouseup', () => {
1787         if (AppState.multiSelecting) {
1788             AppState.multiSelecting = false;
1789             rectEl.style.display = 'none';
1790         }
1791     });
1792
1793 // Запуск приложения при загрузке страницы
1794 document.addEventListener('DOMContentLoaded', () => {
1795     App.init();
1796 });
1797
1798 codegen_graph.js
1799 // js/codegen_graph.js
1800
1801
1802 const CodeGenGraph = {
1803     /**
1804     * Собрать все условия вверх по цепочке cond-портов (до корня).
1805     * Возвращает null или объединённое через AND условие.
1806     */
1807     /**
1808     * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
1809     */
1810     collectAllCond(graph) {
1811         if (!graph) return null;
1812
1813         let c = null;
1814         const elem = graph.elem;
1815
1816         // 1. Собираем условия через cond-порт (как было)
1817         if (graph.condInput) {
1818             const condConn = graph.condInput.conn;
1819             const fromGraph = graph.condInput.fromGraph;
1820             const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
1821             c = oneCond;
1822
1823             // Рекурсивно идём вверх по cond-цепочке
1824             const upCond = this.collectAllCond(fromGraph);
```

```
1825         if (upCond) {
1826             c = c ? Optimizer.And(c, upCond) : upCond;
1827         }
1828     }
1829
1830     // 2. НОВОЕ: если это separator – учитываем контекст его входа
1831     if (elem.type === 'separator' && graph.inputs.length > 0) {
1832         const inputGraph = graph.inputs[0].fromGraph;
1833         const inputContext = this.collectAllCond(inputGraph);
1834         if (inputContext) {
1835             c = c ? Optimizer.And(c, inputContext) : inputContext;
1836         }
1837     }
1838
1839     return c;
1840 },
1841 buildDependencyGraph(elementId) {
1842     const graph = {
1843         nodeId: elementId,
1844         elem: AppState.elements[elementId],
1845         inputs: [],
1846         condInput: null,
1847     };
1848
1849     if (!graph.elem) return null;
1850
1851     const inConns = AppState.connections
1852         .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
1853         .sort((a, b) => {
1854             const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
1855             const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
1856             return ai - bi;
1857         });
1858
1859     inConns.forEach(conn => {
1860         graph.inputs.push({
1861             conn,
1862             fromGraph: this.buildDependencyGraph(conn.fromElement)
1863         });
1864     });
1865
1866     const condConn = AppState.connections.find(c =>
1867         c.toElement === elementId && c.toPort === 'cond-0'
1868     );
1869     if (condConn) {
1870         graph.condInput = {
1871             conn: condConn,
1872             fromGraph: this.buildDependencyGraph(condConn.fromElement)
1873         };
1874     }
1875
1876     return graph;
1877 },
1878 /**
1879 * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
1880 */
1881 evalLogic(graph) {
1882     if (!graph) return Optimizer.TrueCond;
1883     const elem = graph.elem;
1884
1885     switch (elem.type) {
1886         case 'if':
1887             const left = graph.inputs[0]?.fromGraph;
1888             const right = graph.inputs[1]?.fromGraph;
```

```
1890
1891         const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
1892         const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);
1893
1894         const op = elem.props.operator || '=';
1895         return this.buildIfLogic(leftVal, op, rightVal);
1896     }
1897
1898     case 'and': {
1899         let result = null;
1900         for (const inp of graph.inputs) {
1901             const inLogic = this.evalLogic(inp.fromGraph);
1902             result = result ? Optimizer.And(result, inLogic) : inLogic;
1903         }
1904         return result || Optimizer.TrueCond;
1905     }
1906
1907     case 'or': {
1908         let result = null;
1909         for (const inp of graph.inputs) {
1910             const inLogic = this.evalLogic(inp.fromGraph);
1911             result = result ? Optimizer.Or(result, inLogic) : inLogic;
1912         }
1913         return result || Optimizer.FalseCond;
1914     }
1915
1916     case 'not': {
1917         const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
1918         return Optimizer.Not(inLogic);
1919     }
1920
1921     case 'separator': {
1922         return this.evalLogic(graph.inputs[0]?.fromGraph);
1923     }
1924
1925     default:
1926         return Optimizer.TrueCond;
1927     }
1928 },
1929
1930 /**
1931 * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
1932 */
1933 evalValue(graph) {
1934     if (!graph) return Optimizer.Const(0);
1935     const elem = graph.elem;
1936
1937     switch (elem.type) {
1938         case 'input-signal':
1939             return Optimizer.Var(elem.props.name || graph.nodeId);
1940
1941         case 'const':
1942             return Optimizer.Const(Number(elem.props.value) || 0);
1943
1944         case 'formula': {
1945             const expr = this.buildFormulaExpr(elem);
1946             return Optimizer.Var(expr);
1947         }
1948
1949         case 'separator':
1950             return this.evalValue(graph.inputs[0]?.fromGraph);
1951
1952         default:
1953             return Optimizer.Const(0);
1954 }
```

```
1955     },
1956
1957     // js/codegen_graph.js
1958
1959     /**
1960      * Рекурсивно собрать полный контекст условий для элемента
1961      * через всю цепочку cond-портов вверх
1962      */
1963     // В codegen_graph.js, в evalFullContext добавь:
1964
1965     evalFullContext(graph) {
1966         if (!graph) return null;
1967
1968         let context = null;
1969         const elem = graph.elem;
1970
1971         console.log(`evalFullContext для ${elem.id} (${elem.type})`);
1972
1973         // 1. Если сам элемент имеет cond-порт – собираем его условие
1974         if (graph.condInput) {
1975             const condConn = graph.condInput.conn;
1976             console.log(` → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.${condConn.fromPort}`);
1977
1978             const condLogic = this.evalConditionFromPort(
1979                 graph.condInput.fromGraph,
1980                 condConn.fromPort
1981             );
1982             console.log(` → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
1983             context = condLogic;
1984
1985             // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
1986             // порт
1987             const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
1988             if (upstreamContext) {
1989                 console.log(` → upstreamContext: ${Optimizer.printCond(upstreamContext)}`);
1990                 context = context ? Optimizer.And(context, upstreamContext) :
1991                     upstreamContext;
1992             } else {
1993                 console.log(` → нет cond-0`);
1994             }
1995
1996             console.log(` → итоговый контекст: ${Optimizer.printCond(context)}`);
1997             return context;
1998         },
1999
2000         /**
2001          * Получить УСЛОВИЕ для cond-порта элемента
2002          * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
2003          */
2004         evalConditionFromPort(graph, fromPort) {
2005             if (!graph) return null;
2006             const elem = graph.elem;
2007
2008             // Если это сепаратор – вычисляем его вход и применяем ветвление
2009             if (elem.type === 'separator') {
2010                 const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
2011
2012                 if (fromPort === 'out-0') {
2013                     return inputLogic;
2014                 } else if (fromPort === 'out-1') {
2015                     return Optimizer.Not(inputLogic);
2016                 }
2017             }
2018         }
2019     }
2020 }
```

```
2016     }
2017
2018     // Если это логический элемент (AND/OR/NOT/IF) – просто вычисляем логику
2019     if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' || 
2020     elem.type === 'if') {
2021         return this.evalLogic(graph);
2022     }
2023
2024     return null;
2025 },
2026 /**
2027 * Главная функция: получить {cond, expr} для элемента
2028 */
2029 evalGraphValue(graph) {
2030
2031     if (!graph) return { cond: null, expr: Optimizer.Const(0) };
2032
2033     const elem = graph.elem;
2034     //let cond = null;
2035
2036     // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
2037     let cond = this.collectAllCond(graph);
2038
2039     let expr = null;
2040
2041     switch (elem.type) {
2042         case 'input-signal':
2043             expr = Optimizer.Var(elem.props.name || graph.nodeId);
2044             break;
2045
2046         case 'const':
2047             expr = Optimizer.Const(Number(elem.props.value) || 0);
2048             break;
2049
2050         case 'formula': {
2051             // Для формулы также собираем условия от всех входных элементов
2052             const inputConds = graph.inputs.map(inp => {
2053                 const inResult = this.evalGraphValue(inp.fromGraph);
2054                 return inResult.cond;
2055             }).filter(c => c);
2056
2057             // Объединяем cond-порт с условиями от входов
2058             for (const inCond of inputConds) {
2059                 cond = cond ? Optimizer.And(cond, inCond) : inCond;
2060             }
2061
2062             expr = Optimizer.Var(this.buildFormulaExpr(elem));
2063             break;
2064         }
2065
2066         case 'separator':
2067             // Сепаратор – просто пробрасываем значение дальше
2068             return this.evalGraphValue(graph.inputs[0]?.fromGraph);
2069
2070             // Логические элементы не должны здесь быть
2071             case 'and':
2072             case 'or':
2073             case 'not':
2074             case 'if':
2075             default:
2076                 expr = Optimizer.Const(0);
2077             }
2078
2079     return { cond, expr };

```

```
2080     },
2081
2082     buildIfLogic(leftVal, op, rightVal) {
2083         const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
2084         const rightName = rightVal.type === 'var' ? rightVal.name :
2085             String(rightVal.n);
2086         const leftZero = leftVal.type === 'const' && leftVal.n === 0;
2087         const rightZero = rightVal.type === 'const' && rightVal.n === 0;
2088
2089         switch (op) {
2090             case '=':
2091                 if (rightZero) return Optimizer.Eq0(leftName);
2092                 if (leftZero) return Optimizer.Eq0(rightName);
2093                 return Optimizer.Cmp(leftName, '=', rightName);
2094             case '!=':
2095                 if (rightZero) return Optimizer.Ne0(leftName);
2096                 if (leftZero) return Optimizer.Ne0(rightName);
2097                 return Optimizer.Cmp(leftName, '!=', rightName);
2098             case '>':
2099             case '<':
2100             case '>=':
2101             case '<=':
2102                 return Optimizer.Cmp(leftName, op, rightName);
2103             default:
2104                 return Optimizer.TrueCond;
2105         }
2106     },
2107
2108
2109     buildFormulaExpr(elem) {
2110         let result = elem.props.expression || '0';
2111
2112         // 1) Сначала раскрываем шаблоны (h и др.)
2113         const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
2114             ? Settings.getTemplatesMap()
2115             : null;
2116         result = expandFormulaTemplates(result, map);
2117
2118         // 2) Потом раскрываем ссылки на формулы
2119         const formulaRefs = result.match(/formula[_]\d+/g) || [];
2120         for (const ref of formulaRefs) {
2121             const refElem = AppState.elements[ref];
2122             if (refElem && refElem.type === 'formula') {
2123                 const refExpr = this.buildFormulaExpr(refElem);
2124                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
2125             }
2126         }
2127
2128         return result;
2129     }
2130 };
2131
2132 window.CodeGenGraph = CodeGenGraph;
2133
2134 codegen_optimizer.js
2135
2136 // js/codegen_optimizer.js
2137
2138 let _depth = 0;
2139 const MAX_DEPTH = 200;
2140
2141 // === Конструкторы ===
2142 function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
2143 function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
```

```
2144 function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
2145 function And(a, b) {
2146     if (!a) return b;
2147     if (!b) return a;
2148     return { kind: 'cond', type: 'and', a, b };
2149 }
2150 function Or(a, b) {
2151     if (!a) return b;
2152     if (!b) return a;
2153     return { kind: 'cond', type: 'or', a, b };
2154 }
2155 function Not(x) {
2156     if (!x) return null;
2157     return { kind: 'cond', type: 'not', x };
2158 }
2159 const TrueCond = { kind: 'cond', type: 'true' };
2160 const FalseCond = { kind: 'cond', type: 'false' };
2161
2162 function Const(n) { return { kind: 'expr', type: 'const', n }; }
2163 function Var(name) { return { kind: 'expr', type: 'var', name }; }
2164 function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
2165 function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }
2166
2167 // === Утилиты ===
2168 function atomKey(c) {
2169     if (!c) return null;
2170     switch (c.type) {
2171         case 'eq0': return `eq0:${c.v}`;
2172         case 'ne0': return `ne0:${c.v}`;
2173         case 'cmp': return `cmp:${c.l}: ${c.op}: ${c.r}`;
2174         case 'true': return 'true';
2175         case 'false': return 'false';
2176         default: return null;
2177     }
2178 }
2179
2180 function splitAndCond(c) {
2181     if (!c || c.type !== 'and') return null;
2182     return [c.a, c.b];
2183 }
2184
2185 function findSharedAndComplement(c1, c2) {
2186     const p1 = splitAndCond(c1);
2187     const p2 = splitAndCond(c2);
2188     if (!p1 || !p2) return null;
2189
2190     const combos = [
2191         [p1[0], p1[1], p2[0], p2[1]],
2192         [p1[0], p1[1], p2[1], p2[0]],
2193         [p1[1], p1[0], p2[0], p2[1]],
2194         [p1[1], p1[0], p2[1], p2[0]],
2195     ];
2196
2197     for (const [s1, x1, s2, x2] of combos) {
2198         if (condEq(s1, s2) && condNegationEq(x1, x2)) {
2199             return { shared: s1 };
2200         }
2201     }
2202     return null;
2203 }
2204
2205 function negateOp(op) {
2206     switch (op) {
2207         case '=': return '!=';
2208         case '!=': return '=';
```

```
2209         case '>': return '<=';
2210         case '<': return '>=';
2211         case '>=': return '<';
2212         case '<=': return '>';
2213         default: return null;
2214     }
2215 }
2216
2217 // Преобразует cmp-условие в интервал по одной переменной
2218 // Возвращает { varName, min, minInc, max, maxInc } или null
2219 function cmpToInterval(c) {
2220     if (!c || c.type !== 'cmp') return null;
2221
2222     const lNum = parseNumberLiteral(c.l);
2223     const rNum = parseNumberLiteral(c.r);
2224
2225     let varName, op, val;
2226
2227     if (lNum == null && rNum != null) {
2228         // var OP const
2229         varName = c.l;
2230         op = c.op;
2231         val = rNum;
2232     } else if (lNum != null && rNum == null) {
2233         // const OP var -> var (OP') const
2234         varName = c.r;
2235         op = reverseOp(c.op);
2236         if (!op) return null;
2237         val = lNum;
2238     } else {
2239         // Либо обе стороны числа, либо обе не числа – не трогаем
2240         return null;
2241     }
2242
2243     // Интересуют только упорядочивающие операторы
2244     switch (op) {
2245         case '<':
2246         case '<=':
2247         case '>':
2248         case '>=':
2249         case '=':
2250             break;
2251         default:
2252             return null;
2253     }
2254
2255     let min = Number.NEGATIVE_INFINITY;
2256     let max = Number.POSITIVE_INFINITY;
2257     let minInc = false;
2258     let maxInc = false;
2259
2260     switch (op) {
2261         case '<':
2262             max = val; maxInc = false; break;
2263         case '<=':
2264             max = val; maxInc = true; break;
2265         case '>':
2266             min = val; minInc = false; break;
2267         case '>=':
2268             min = val; minInc = true; break;
2269         case '=':
2270             min = val; minInc = true;
2271             max = val; maxInc = true;
2272             break;
2273     }
```

```
2274
2275     return { varName, min, minInc, max, maxInc };
2276 }
2277
2278 function intervalSubset(a, b) {
2279     if (!a || !b) return false;
2280
2281     // Нижняя граница: a.min >= b.min
2282     const amin = a.min, bmin = b.min;
2283     if (amin === Number.NEGATIVE_INFINITY) {
2284         if (bmin !== Number.NEGATIVE_INFINITY) return false;
2285         // оба -∞ – ок
2286     } else if (bmin === Number.NEGATIVE_INFINITY) {
2287         // b начинается “раньше” – ок
2288     } else if (amin > bmin) {
2289         // a стартует правее b – ок
2290     } else if (amin < bmin) {
2291         // a захватывает меньшее значение – не подмножество
2292         return false;
2293     } else {
2294         // amin === bmin
2295         if (a.minInc && !b.minInc) {
2296             // a включает границу, а b – нет → в a есть точка, не входящая в b
2297             return false;
2298         }
2299     }
2300
2301     // Верхняя граница: a.max <= b.max
2302     const amax = a.max, bmax = b.max;
2303     if (amax === Number.POSITIVE_INFINITY) {
2304         if (bmax !== Number.POSITIVE_INFINITY) return false;
2305     } else if (bmax === Number.POSITIVE_INFINITY) {
2306         // b идёт дальше – ок
2307     } else if (amax < bmax) {
2308         // a заканчивается раньше – ок
2309     } else if (amax > bmax) {
2310         return false;
2311     } else {
2312         // amax === bmax
2313         if (a.maxInc && !b.maxInc) {
2314             return false;
2315         }
2316     }
2317
2318     return true;
2319 }
2320
2321 // Удаляет избыточные cmp-условия в массиве атомов
2322 // mode: 'and' | 'or'
2323 function removeRedundantCmpAtoms(atoms, mode) {
2324     if (!atoms || atoms.length < 2) return atoms;
2325
2326     const keep = new Array(atoms.length).fill(true);
2327
2328     for (let i = 0; i < atoms.length; i++) {
2329         if (!keep[i]) continue;
2330         const a = atoms[i];
2331         if (!a || a.type !== 'cmp') continue;
2332
2333         for (let j = 0; j < atoms.length; j++) {
2334             if (i === j || !keep[j]) continue;
2335             const b = atoms[j];
2336             if (!b || b.type !== 'cmp') continue;
2337
2338             const rel = cmpImplicationRelation(a, b);
```

```
2339         if (!rel) continue;
2340
2341         if (rel === 'a_in_b') {
2342             if (mode === 'or') {
2343                 // A ⊆ B → A OR B = B → A лишнее
2344                 keep[i] = false;
2345                 break;
2346             } else if (mode === 'and') {
2347                 // A ⊆ B → A AND B = A → B лишнее
2348                 keep[j] = false;
2349             }
2350         } else if (rel === 'b_in_a') {
2351             if (mode === 'or') {
2352                 // B ⊆ A → A OR B = A → B лишнее
2353                 keep[j] = false;
2354             } else if (mode === 'and') {
2355                 // B ⊆ A → A AND B = B → A лишнее
2356                 keep[i] = false;
2357                 break;
2358             }
2359         }
2360     }
2361 }
2362
2363     return atoms.filter(_ , idx) => keep[idx]);
2364 }
2365
2366 // Отношение между двумя cmp-условиями через интервалы
2367 // 'a_in_b' – A ⊆ B
2368 // 'b_in_a' – B ⊆ A
2369 // 'equal' – одинаковые интервалы (редко используем)
2370 // null – не можем определить
2371 function cmpImplicationRelation(c1, c2) {
2372     const i1 = cmpToInterval(c1);
2373     const i2 = cmpToInterval(c2);
2374     if (!i1 || !i2) return null;
2375     if (i1.varName !== i2.varName) return null;
2376
2377     const aInB = intervalSubset(i1, i2);
2378     const bInA = intervalSubset(i2, i1);
2379
2380     if (aInB && bInA) return 'equal';
2381     if (aInB) return 'a_in_b';
2382     if (bInA) return 'b_in_a';
2383     return null;
2384 }
2385
2386 // Разворот оператора при перестановке аргументов (левый/правый)
2387 function reverseOp(op) {
2388     switch (op) {
2389         case '<': return '>';
2390         case '>': return '<';
2391         case '<=': return '>=';
2392         case '>=': return '<=';
2393         case '=':
2394         case '!=':
2395             return op;
2396         default:
2397             return null;
2398     }
2399 }
2400
2401 // Аккуратный парсер числового литерала.
2402 // Возвращает число или null, если строка не чисто числовая.
2403 function parseNumberLiteral(s) {
```

```
2404     if (typeof s !== 'string') return null;
2405     const trimmed = s.trim().replace(',', '.');
2406
2407     // Только простые вещи: -123, 45, 3.14
2408     if (!/^-?\d+(\.\d+)?$/_.test(trimmed)) return null;
2409
2410     const n = Number(trimmed);
2411     return Number.isFinite(n) ? n : null;
2412 }
2413
2414
2415 function negateAtomKey(key) {
2416     if (!key) return null;
2417     if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
2418     if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
2419     if (key.startsWith('cmp:')) {
2420         const parts = key.slice(4).split(':');
2421         if (parts.length === 3) {
2422             const negOp = negateOp(parts[1]);
2423             if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
2424         }
2425     }
2426     return null;
2427 }
2428
2429 function isNegation(a, b) {
2430     if (!a || !b) return false;
2431     if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
2432     if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
2433     if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
2434         return a.op === negateOp(b.op);
2435     }
2436     if (a.type === 'not' && condEq(a.x, b)) return true;
2437     if (b.type === 'not' && condEq(b.x, a)) return true;
2438     return false;
2439 }
2440
2441 function isAtomCond(t) {
2442     return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
2443 }
2444
2445 function pruneOrByContext(orTerm, contextAtoms) {
2446     const branches = flattenOr(orTerm);
2447     const kept = [];
2448
2449     for (const br of branches) {
2450         let contradicts = false;
2451
2452         for (const ctx of contextAtoms) {
2453             if (isNegation(br, ctx)) {
2454                 contradicts = true;
2455                 break;
2456             }
2457         }
2458
2459         if (!contradicts) kept.push(br);
2460     }
2461
2462     if (kept.length === 0) return FalseCond;
2463     if (kept.length === 1) return kept[0];
2464     return buildOr(kept);
2465 }
2466
2467 function condNegationEq(a, b) {
2468     if (!a || !b) return false;
```

```
2469
2470     // Простая проверка: a == NOT(b)
2471     if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;
2472
2473     // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
2474     // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
2475     if (a.type === 'or' && b.type === 'and') {
2476         return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2477             condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2478     }
2479     // Симметрично
2480     if (a.type === 'and' && b.type === 'or') {
2481         return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2482             condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2483     }
2484
2485     // Проверка атомов: (X = 0) vs (X != 0)
2486     if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
2487     if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
2488
2489     // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
2490     if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
2491         return a.op === negateOp(b.op);
2492     }
2493
2494     return false;
2495 }
2496
2497
2498
2499 function condEq(a, b) {
2500     if (a === b) return true;
2501     if (!a || !b) return false;
2502     if (a.type !== b.type) return false;
2503
2504     switch (a.type) {
2505         case 'eq0':
2506         case 'ne0':
2507             return a.v === b.v;
2508         case 'cmp':
2509             return a.l === b.l && a.op === b.op && a.r === b.r;
2510         case 'true':
2511         case 'false':
2512             return true;
2513         case 'not':
2514             return condEq(a.x, b.x);
2515         case 'and':
2516         case 'or':
2517             return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
2518                 (condEq(a.a, b.b) && condEq(a.b, b.a));
2519         default:
2520             return false;
2521     }
2522 }
2523
2524 function flattenAnd(c) {
2525     if (!c) return [];
2526     if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
2527     return [c];
2528 }
2529
2530 function flattenOr(c) {
2531     if (!c) return [];
2532     if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
2533     return [c];
```

```
2534 }
2535
2536 function buildAnd(terms) {
2537     if (terms.length === 0) return TrueCond;
2538     let result = terms[0];
2539     for (let i = 1; i < terms.length; i++) {
2540         result = And(result, terms[i]);
2541     }
2542     return result;
2543 }
2544
2545 function buildOr(terms) {
2546     if (terms.length === 0) return FalseCond;
2547     let result = terms[0];
2548     for (let i = 1; i < terms.length; i++) {
2549         result = Or(result, terms[i]);
2550     }
2551     return result;
2552 }
2553
2554 // Поглощение для AND: X AND (X OR Y) = X
2555 function applyAndAbsorption(terms) {
2556     if (!terms || terms.length < 2) return terms;
2557
2558     const keep = new Array(terms.length).fill(true);
2559
2560     for (let i = 0; i < terms.length; i++) {
2561         if (!keep[i]) continue;
2562         const ti = terms[i];
2563         if (!ti || ti.type !== 'or') continue;
2564
2565         const orParts = flattenOr(ti);
2566         let drop = false;
2567
2568         outer:
2569             for (const part of orParts) {
2570                 for (let j = 0; j < terms.length; j++) {
2571                     if (j === i || !keep[j]) continue;
2572                     if (condEq(part, terms[j])) {
2573                         drop = true;
2574                         break outer;
2575                     }
2576                 }
2577             }
2578
2579             if (drop) {
2580                 keep[i] = false;
2581             }
2582     }
2583
2584     return terms.filter((_, idx) => keep[idx]);
2585 }
2586
2587 // Поглощение для OR: X OR (X AND Y) = X
2588 function applyOrAbsorption(terms) {
2589     if (!terms || terms.length < 2) return terms;
2590
2591     const keep = new Array(terms.length).fill(true);
2592
2593     for (let i = 0; i < terms.length; i++) {
2594         if (!keep[i]) continue;
2595         const ti = terms[i];
2596         if (!ti || ti.type !== 'and') continue;
2597
2598         const andParts = flattenAnd(ti);
```

```
2599     let drop = false;
2600
2601     outer:
2602     for (const part of andParts) {
2603         for (let j = 0; j < terms.length; j++) {
2604             if (j === i || !keep[j]) continue;
2605             if (condEq(part, terms[j])) {
2606                 drop = true;
2607                 break outer;
2608             }
2609         }
2610     }
2611
2612     if (drop) {
2613         keep[i] = false;
2614     }
2615 }
2616
2617     return terms.filter(_ => keep[idx]);
2618 }
2619
2620 // === Упрощение условий ===
2621 function simplifyCond(c) {
2622     _depth++;
2623     if (_depth > MAX_DEPTH) {
2624         _depth--;
2625         return c;
2626     }
2627
2628     try {
2629         return simplifyCondCore(c);
2630     } finally {
2631         _depth--;
2632     }
2633 }
2634
2635 function simplifyCondCore(c) {
2636     if (!c || c.kind !== 'cond') return c;
2637
2638     switch (c.type) {
2639         case 'true':
2640         case 'false':
2641         case 'eq0':
2642         case 'ne0':
2643         case 'cmp':
2644             return c;
2645
2646         case 'not': {
2647             const x = simplifyCondCore(c.x);
2648             if (!x) return TrueCond;
2649             if (x.type === 'true') return FalseCond;
2650             if (x.type === 'false') return TrueCond;
2651             if (x.type === 'not') return simplifyCondCore(x.x);
2652             if (x.type === 'eq0') return Ne0(x.v);
2653             if (x.type === 'ne0') return Eq0(x.v);
2654             if (x.type === 'cmp') {
2655                 const neg0p = negateOp(x.op);
2656                 if (neg0p) return Cmp(x.l, neg0p, x.r);
2657             }
2658             if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
2659             if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
2660             return Not(x);
2661         }
2662     }
2663     case 'and': {
```

```
2664     const a = simplifyCondCore(c.a);
2665     const b = simplifyCondCore(c.b);
2666
2667     if (!a) return b;
2668     if (!b) return a;
2669     if (a.type === 'false' || b.type === 'false') return FalseCond;
2670     if (a.type === 'true') return b;
2671     if (b.type === 'true') return a;
2672
2673     const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
2674
2675     // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
2676     const eq0Vars = new Map(); // var -> term
2677     const ne0Vars = new Map(); // var -> term
2678     const cmpTerms = [];
2679     const otherTerms = [];
2680
2681     for (const t of allTerms) {
2682         if (t.type === 'true') continue;
2683         if (t.type === 'false') return FalseCond;
2684
2685         if (t.type === 'eq0') {
2686             // Проверка на противоречие сразу
2687             if (ne0Vars.has(t.v)) {
2688                 console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
2689                 return FalseCond;
2690             }
2691             eq0Vars.set(t.v, t);
2692         } else if (t.type === 'ne0') {
2693             // Проверка на противоречие сразу
2694             if (eq0Vars.has(t.v)) {
2695                 console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
2696                 return FalseCond;
2697             }
2698             ne0Vars.set(t.v, t);
2699         } else if (t.type === 'cmp') {
2700             cmpTerms.push(t);
2701         } else if (t.type === 'or') {
2702             // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
2703             const orTerms = flattenOr(t);
2704             const validBranches = [];
2705
2706             for (const branch of orTerms) {
2707                 let branchValid = true;
2708
2709                 if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
2710                     console.log(`OR ветка ${branch.v} != 0 противоречит контексту ${branch.v} = 0`);
2711                     branchValid = false;
2712                 } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
2713                     console.log(`OR ветка ${branch.v} = 0 противоречит контексту ${branch.v} != 0`);
2714                     branchValid = false;
2715                 }
2716
2717                 if (branchValid) {
2718                     validBranches.push(branch);
2719                 }
2720             }
2721
2722             if (validBranches.length === 0) {
2723                 console.log(`Все ветки OR противоречат контексту → FALSE`);
2724                 return FalseCond;
2725             } else if (validBranches.length === 1) {
2726                 // Если осталась только одна ветка OR, добавляем её напрямую
```

```
2727         const singleBranch = validBranches[0];
2728         if (singleBranch.type === 'eq0') {
2729             if (ne0Vars.has(singleBranch.v)) return FalseCond;
2730             eq0Vars.set(singleBranch.v, singleBranch);
2731         } else if (singleBranch.type === 'ne0') {
2732             if (eq0Vars.has(singleBranch.v)) return FalseCond;
2733             ne0Vars.set(singleBranch.v, singleBranch);
2734         } else {
2735             otherTerms.push(singleBranch);
2736         }
2737     } else {
2738         // Перестраиваем OR только с валидными ветками
2739         otherTerms.push(buildOr(validBranches));
2740     }
2741 } else {
2742     otherTerms.push(t);
2743 }
2744 }
2745
2746 // Собираем уникальные атомы
2747 const atomMap = new Map();
2748
2749 for (const [v, term] of eq0Vars) {
2750     const key = atomKey(term);
2751     if (key) atomMap.set(key, term);
2752 }
2753
2754 for (const [v, term] of ne0Vars) {
2755     const key = atomKey(term);
2756     if (key) atomMap.set(key, term);
2757 }
2758
2759 for (const term of cmpTerms) {
2760     const key = atomKey(term);
2761     if (key) {
2762         const negKey = negateAtomKey(key);
2763         if (negKey && atomMap.has(negKey)) {
2764             return FalseCond;
2765         }
2766         if (!atomMap.has(key)) {
2767             atomMap.set(key, term);
2768         }
2769     }
2770 }
2771
2772 let uniqueAtoms = Array.from(atomMap.values());
2773 uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
2774
2775 let result = [...uniqueAtoms, ...otherTerms];
2776
2777 // Поглощение: X AND (X OR Y) = X
2778 // === НОВОЕ: выбрасываем из OR ветки, противоречавшие контексту AND ===
2779 const contextAtoms = result.filter(t => isAtomCond(t));
2780 result = result.map(t => {
2781     if (t.type !== 'or') return t;
2782     return pruneOrByContext(t, contextAtoms);
2783 }).filter(t => t.type !== 'true'); // на всякий случай
2784
2785 result = applyAndAbsorption(result);
2786
2787 if (result.length === 0) return TrueCond;
2788 if (result.length === 1) return result[0];
2789
2790 return buildAnd(result);
2791 }
```

```
2792
2793     case 'or': {
2794         const a = simplifyCondCore(c.a);
2795         const b = simplifyCondCore(c.b);
2796
2797         if (!a) return b;
2798         if (!b) return a;
2799         if (a.type === 'true' || b.type === 'true') return TrueCond;
2800         if (a.type === 'false') return b;
2801         if (b.type === 'false') return a;
2802
2803         const allTerms = [...flattenOr(a), ...flattenOr(b)];
2804         const atomMap = new Map();
2805         const otherTerms = [];
2806
2807         for (const t of allTerms) {
2808             if (t.type === 'true') return TrueCond;
2809             if (t.type === 'false') continue;
2810
2811             const key = atomKey(t);
2812             if (key) {
2813                 const negKey = negateAtomKey(key);
2814                 if (negKey && atomMap.has(negKey)) {
2815                     return TrueCond;
2816                 }
2817                 if (!atomMap.has(key)) {
2818                     atomMap.set(key, t);
2819                 }
2820             } else {
2821                 otherTerms.push(t);
2822             }
2823         }
2824
2825         let uniqueAtoms = Array.from(atomMap.values());
2826         uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
2827
2828         let result = [...uniqueAtoms, ...otherTerms];
2829
2830         // Поглощение: X OR (X AND Y) = X
2831         result = applyOrAbsorption(result);
2832
2833         if (result.length === 0) return FalseCond;
2834         if (result.length === 1) return result[0];
2835
2836         return buildOr(result);
2837     }
2838
2839     default:
2840         return c;
2841     }
2842 }
2843
2844 // === Сравнение выражений ===
2845 function exprEq(a, b) {
2846     if (a === b) return true;
2847     if (!a && !b) return true;
2848     if (!a || !b) return false;
2849     if (a.type !== b.type) return false;
2850
2851     switch (a.type) {
2852         case 'const': return a.n === b.n;
2853         case 'var': return a.name === b.name;
2854         case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
2855         case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
2856     default: return false;
```

```
2857     }
2858 }
2859
2860 // === Упрощение выражений ===
2861 function simplifyExpr(expr) {
2862     _depth++;
2863     if (_depth > MAX_DEPTH) {
2864         _depth--;
2865         return expr;
2866     }
2867
2868     try {
2869         return simplifyExprCore(expr);
2870     } finally {
2871         _depth--;
2872     }
2873 }
2874
2875 function simplifyExprCore(expr) {
2876     if (!expr || expr.kind !== 'expr') return expr;
2877
2878     switch (expr.type) {
2879         case 'const':
2880         case 'var':
2881             return expr;
2882
2883         case 'op': {
2884             const l = simplifyExprCore(expr.l);
2885             const r = simplifyExprCore(expr.r);
2886
2887             if (expr.op === '+') {
2888                 if (r?.type === 'const' && r.n === 0) return l;
2889                 if (l?.type === 'const' && l.n === 0) return r;
2890             }
2891             if (expr.op === '*') {
2892                 if (l?.type === 'const' && l.n === 0) return Const(0);
2893                 if (r?.type === 'const' && r.n === 0) return Const(0);
2894                 if (l?.type === 'const' && l.n === 1) return r;
2895                 if (r?.type === 'const' && r.n === 1) return l;
2896             }
2897             return Op(expr.op, l, r);
2898         }
2899
2900         case 'when': {
2901             const c = simplifyCond(expr.c);
2902             const t = simplifyExprCore(expr.t);
2903             const e = simplifyExprCore(expr.e);
2904
2905             if (c?.type === 'true') return t;
2906             if (c?.type === 'false') return e;
2907             if (exprEq(t, e)) return t;
2908             // ✅ HOBOE: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
2909             if (e && e.type === 'when') {
2910                 const c2 = simplifyCond(e.c);
2911                 const t2 = simplifyExprCore(e.t);
2912                 const e2 = simplifyExprCore(e.e);
2913
2914                 if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
2915                     return When(c, t, t2);
2916                 }
2917             }
2918             // Узкое правило: WHEN(A&B, t1, WHEN(A&¬B, t2, WHEN(¬A, t3, e3))) -> ...
2919             if (e && e.type === 'when') {
2920                 const c2 = e.c, t2 = e.t, e2 = e.e;
```

```
2921             if (e2 && e2.type === 'when') {
2922                 const c3 = e2.c, t3 = e2.t;
2923
2924                 const shared = findSharedAndComplement(c, c2);
2925                 if (shared && condNegationEq(c3, shared.shared)) {
2926                     return When(c, t, When(c2, t2, t3));
2927                 }
2928             }
2929         }
2930     }
2931
2932     return When(c, t, e);
2933 }
2934
2935     default:
2936         return expr;
2937     }
2938 }
2939
2940 // === Печать ===
2941 function printCond(c) {
2942     if (!c) return 'TRUE';
2943
2944     switch (c.type) {
2945         case 'eq0': return `${c.v} = 0`;
2946         case 'ne0': return `${c.v} != 0`;
2947         case 'cmp': return `${c.l} ${c.op} ${c.r}`;
2948         case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
2949         case 'or': return `(${printCond(c.a)} OR ${printCond(c.b)})`;
2950         case 'not': return `NOT(${printCond(c.x)})`;
2951         case 'true': return 'TRUE';
2952         case 'false': return 'FALSE';
2953         default: return '?';
2954     }
2955 }
2956
2957 function printExpr(e) {
2958     if (!e) return '0';
2959
2960     switch (e.type) {
2961         case 'const': return String(e.n);
2962         case 'var': return e.name;
2963         case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
2964         case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, ${printExpr(e.e)})`;
2965         default: return '?';
2966     }
2967 }
2968
2969 window.Optimizer = {
2970     Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
2971     Const, Var, Op, When,
2972     simplifyCond, simplifyExpr,
2973     printCond, printExpr,
2974     condEq, exprEq
2975 };
2976
2977 codegen.js
2978
2979 // js/codegen.js
2980
2981 const CodeGen = {
2982     _cache: {},
2983     _branchCache: {},
2984     _resolveCache: {},
```

```
2985     _visiting: new Set(),
2986
2987     reset() {
2988         this._cache = {};
2989         this._branchCache = {};
2990         this._resolveCache = {};
2991         this._visiting = new Set();
2992     },
2993
2994     toExpr(valueStr) {
2995         const s = String(valueStr).trim();
2996         if (s === '0') return Optimizer.Const(0);
2997         const num = parseFloat(s);
2998         if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
2999         return Optimizer.Var(s);
3000     },
3001
3002     exprToName(exprAst) {
3003         if (!exprAst) return '0';
3004         if (exprAst.type === 'var') return exprAst.name;
3005         if (exprAst.type === 'const') return String(exprAst.n);
3006         return Optimizer.printExpr(exprAst);
3007     },
3008
3009     mergeCond(a, b) {
3010         if (!a && !b) return null;
3011         if (!a) return b;
3012         if (!b) return a;
3013         if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
3014         return Optimizer.And(a, b);
3015     },
3016
3017     getConn(toId, toPort) {
3018         return AppState.connections.find(c => c.toElement === toId && c.toPort ===
3019             toPort);
3020     },
3021
3022     getConns(toId, prefix) {
3023         return AppState.connections.filter(c => c.toElement === toId &&
3024             c.toPort.startsWith(prefix));
3025
3026     buildFormulaExpr(elem) {
3027         let result = elem.props.expression || '0';
3028
3029         // 1) Сначала раскрываем шаблоны (h и др.)
3030         const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
3031             ? Settings.getTemplatesMap()
3032             : null;
3033         result = expandFormulaTemplates(result, map);
3034
3035         // 2) Потом раскрываем ссылки на формулы
3036         const formulaRefs = result.match(/formula[-]\d+/g) || [];
3037         for (const ref of formulaRefs) {
3038             const refElem = AppState.elements[ref];
3039             if (refElem && refElem.type === 'formula') {
3040                 const refExpr = this.buildFormulaExpr(refElem);
3041                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
3042             }
3043
3044         return result;
3045     },
3046
3047     // === Получить ЧИСТУЮ логику элемента ===
```

```
3048     getPureLogic(id) {
3049         const cacheKey = `logic:${id}`;
3050         if (cacheKey in this._cache) {
3051             return this._cache[cacheKey];
3052         }
3053
3054         const elem = AppState.elements[id];
3055         if (!elem) return null;
3056
3057         let logic = null;
3058
3059         switch (elem.type) {
3060             case 'if': {
3061                 const leftConn = this.getConn(id, 'in-0');
3062                 const rightConn = this.getConn(id, 'in-1');
3063
3064                 const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
3065                     Optimizer.Const(0);
3066                 const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
3067                     Optimizer.Const(0);
3068
3069                 const op = (elem.props.operator || '=').trim();
3070                 const leftName = this.exprToName(leftVal);
3071                 const rightName = this.exprToName(rightVal);
3072
3073                 const leftZero = leftVal.type === 'const' && leftVal.n === 0;
3074                 const rightZero = rightVal.type === 'const' && rightVal.n === 0;
3075
3076                 switch (op) {
3077                     case '=':
3078                         if (rightZero) {
3079                             logic = Optimizer.Eq0(leftName);
3080                         } else if (leftZero) {
3081                             logic = Optimizer.Eq0(rightName);
3082                         } else {
3083                             logic = Optimizer.Cmp(leftName, '=', rightName);
3084                         }
3085                         break;
3086                     case '!=':
3087                         if (rightZero) {
3088                             logic = Optimizer.Ne0(leftName);
3089                         } else if (leftZero) {
3090                             logic = Optimizer.Ne0(rightName);
3091                         } else {
3092                             logic = Optimizer.Cmp(leftName, '!=', rightName);
3093                         }
3094                         break;
3095                     case '>':
3096                     case '<':
3097                     case '>=':
3098                     case '<=':
3099                         logic = Optimizer.Cmp(leftName, op, rightName);
3100                         break;
3101                     default:
3102                         logic = Optimizer.TrueCond;
3103                 }
3104             }
3105             case 'and':
3106             case 'or': {
3107                 const isAnd = elem.type === 'and';
3108                 const count = elem.props.inputCount || 2;
3109                 let result = null;
3110             }
3111         }
3112     }
```

```
3111         for (let i = 0; i < count; i++) {
3112             const conn = this.getConn(id, `in-${i}`);
3113             if (!conn) continue;
3114
3115             const val = this.getPureLogic(conn.fromElement);
3116             if (!val) continue;
3117
3118             if (result === null) {
3119                 result = val;
3120             } else {
3121                 result = isAnd ? Optimizer.And(result, val) :
3122 Optimizer.Or(result, val);
3123             }
3124             logic = result || Optimizer.FalseCond;
3125             break;
3126         }
3127
3128     case 'not': {
3129         const conn = this.getConn(id, 'in-0');
3130         const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
3131         logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
3132         break;
3133     }
3134
3135     case 'separator': {
3136         const conn = this.getConn(id, 'in-0');
3137         logic = conn ? this.getPureLogic(conn.fromElement) :
3138 Optimizer.FalseCond;
3139         break;
3140     }
3141     default:
3142         logic = null;
3143     }
3144
3145     // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
3146     if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
3147 elem.type === 'not') {
3148         const ctx = this.getConditionFromPort(id);
3149         if (ctx) {
3150             if (logic) {
3151                 logic = Optimizer.And(ctx, logic);
3152             } else {
3153                 logic = ctx;
3154             }
3155         }
3156
3157         this._cache[cacheKey] = logic;
3158         return logic;
3159     },
3160
3161     // === Получить значение ===
3162     getValue(id) {
3163         const elem = AppState.elements[id];
3164         if (!elem) return Optimizer.Const(0);
3165
3166         switch (elem.type) {
3167             case 'input-signal':
3168                 // Имя сигнала или id как Var(...)
3169                 return this.toExpr(elem.props.name || id);
3170
3171             case 'const':
3172                 return Optimizer.Const(Number(elem.props.value) || 0);
3173         }
3174     }
3175 }
```

```
3173
3174     case 'formula': {
3175         // Используем текст формулы как выражение
3176         const exprStr = this.buildFormulaExpr(elem) || '0';
3177         return this.toExpr(exprStr);
3178     }
3179
3180     default:
3181         // На всякий случай – даём символьическое имя, а не 0
3182         if (elem.props && typeof elem.props.name === 'string') {
3183             return this.toExpr(elem.props.name);
3184         }
3185         return this.toExpr(id);
3186     }
3187 },
3188
3189 // === Получить ПОЛНОЕ условие для ветки сепаратора ===
3190 getBranchCondition(sepId, fromPort) {
3191     const cacheKey = `${sepId}:${fromPort}`;
3192     if (cacheKey in this._branchCache) {
3193         return this._branchCache[cacheKey];
3194     }
3195
3196     const sep = AppState.elements[sepId];
3197     if (!sep || sep.type !== 'separator') return null;
3198
3199     const inputLogic = this.getPureLogic(sepId);
3200     const sepContext = this.getConditionFromPort(sepId);
3201
3202     let branchLogic;
3203     if (fromPort === 'out-1') {
3204         branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
3205     } else {
3206         branchLogic = inputLogic || Optimizer.TrueCond;
3207     }
3208
3209     let result;
3210     if (sepContext) {
3211         result = Optimizer.And(sepContext, branchLogic);
3212     } else {
3213         result = branchLogic;
3214     }
3215
3216     this._branchCache[cacheKey] = result;
3217     return result;
3218 },
3219
3220 // === Получить условие от cond-порта ===
3221 getConditionFromPort(id) {
3222     const conn = this.getConn(id, 'cond-0');
3223     if (!conn) return null;
3224
3225     const sourceElem = AppState.elements[conn.fromElement];
3226     if (!sourceElem) return null;
3227
3228     if (sourceElem.type === 'separator') {
3229         return this.getBranchCondition(conn.fromElement, conn.fromPort);
3230     }
3231
3232     return this.getPureLogic(conn.fromElement);
3233 },
3234
3235 // === Основная функция разрешения ===
3236 resolve(id) {
3237     if (id in this._resolveCache) {
```

```
3238         return this._resolveCache[id];
3239     }
3240
3241     if (this._visiting.has(id)) {
3242         return null;
3243     }
3244     this._visiting.add(id);
3245
3246     const elem = AppState.elements[id];
3247     if (!elem) {
3248         this._visiting.delete(id);
3249         return null;
3250     }
3251
3252     let result = null;
3253
3254     try {
3255         switch (elem.type) {
3256             case 'input-signal':
3257                 result = {
3258                     isValue: true,
3259                     cond: null,
3260                     expr: this.toExpr(elem.props.name || id)
3261                 };
3262                 break;
3263
3264             case 'const':
3265                 const cond = this.getConditionFromPort(id);
3266                 result = {
3267                     isValue: true,
3268                     cond: cond,
3269                     expr: Optimizer.Const(Number(elem.props.value) || 0)
3270                 };
3271                 break;
3272             }
3273
3274             case 'formula':
3275                 let cond = this.getConditionFromPort(id);
3276
3277                 const inConns = this.getConns(id, 'in-');
3278                 for (const conn of inConns) {
3279                     const inputNode = this.resolve(conn.fromElement);
3280                     if (inputNode && inputNode.cond) {
3281                         cond = this.mergeCond(cond, inputNode.cond);
3282                     }
3283                 }
3284
3285                 const fullExpr = this.buildFormulaExpr(elem);
3286                 result = {
3287                     isValue: true,
3288                     cond: cond,
3289                     expr: Optimizer.Var(fullExpr)
3290                 };
3291                 break;
3292             }
3293
3294             default:
3295                 result = null;
3296             }
3297         } finally {
3298             this._visiting.delete(id);
3299         }
3300
3301         this._resolveCache[id] = result;
3302         return result;
3303     }
```

```
3303     },
3304
3305     generate() {
3306         console.log('==> Генерация кода (граф) ==>');
3307         this.reset();
3308
3309         try {
3310             const outputs = Object.values(AppState.elements).filter(e => e.type ===
3311             'output');
3312
3313             if (outputs.length === 0) {
3314                 return /* Нет выходов */;
3315             }
3316
3317             const allVariants = [];
3318
3319             for (const out of outputs) {
3320                 const conns = this.getConns(out.id, 'in-');
3321
3322                 for (const conn of conns) {
3323                     console.log(`\n==> Обработка выхода ${out.id}, вход от $` +
3324             {conn.fromElement} ==>`);
3325                     const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
3326                     const result = CodeGenGraph.evalGraphValue(graph);
3327                     console.log(`Результат: cond=${Optimizer.printCond(result.cond)},` +
3328             `expr=${Optimizer.printExpr(result.expr)} `);
3329
3330                     if (!result || !result.expr) continue;
3331
3332                     const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
3333             null;
3334                     const isZero = result.expr.type === 'const' && result.expr.n ===
3335             0;
3336
3337                     if (isZero && !cond) continue;
3338
3339                     allVariants.push({
3340                         cond,
3341                         expr: result.expr,
3342                         isZero
3343                     });
3344                 }
3345             }
3346
3347             console.log('Варианты:', allVariants.map(v => ({
3348                 cond: Optimizer.printCond(v.cond),
3349                 expr: Optimizer.printExpr(v.expr)
3350             })));
3351
3352             if (allVariants.length === 0) return '0';
3353
3354             const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
3355             if (valueVariants.length === 0) return '0';
3356
3357             let result = Optimizer.Const(0);
3358
3359             for (let i = valueVariants.length - 1; i >= 0; i--) {
3360                 const v = valueVariants[i];
3361                 if (v.cond) {
3362                     result = Optimizer.When(v.cond, v.expr, result);
3363                 } else {
3364                     result = v.expr;
3365                 }
3366             }
3367         }
3368     }
```

```
3363         const simplified = Optimizer.simplifyExpr(result);
3364         return Optimizer.printExpr(simplified);
3365     }
3366     } catch (err) {
3367         console.error('Ошибка:', err);
3368         return `/* Ошибка: ${err.message} */`;
3369     }
3370 }
3371 };
3372
3373 windowCodeGen = CodeGen;
3374
3375 config.js:
3376 /**
3377 * Конфигурация приложения
3378 */
3380
3381 // Типы сигналов
3382 const SIGNAL_TYPE = {
3383     NUMERIC: 'numeric',      // Числовой сигнал
3384     LOGIC: 'logic',          // Логический (может быть TRUE или FALSE)
3385     TRUE: 'true',            // Явно ИСТИНА
3386     FALSE: 'false',          // Явно ЛОЖЬ
3387     ANY: 'any'               // Любой тип
3388 };
3389
3390 // Типы проекта
3391 const PROJECT_TYPE = {
3392     PARAMETER: 'parameter',
3393     RULE: 'rule'
3394 };
3395
3396 // Конфигурация элементов
3397 const ELEMENT_TYPES = {
3398     'input-signal': {
3399         name: 'Вход',
3400         inputs: 0,
3401         outputs: 1,
3402         outputLabels: ['out'],
3403         outputTypes: [SIGNAL_TYPE.NUMERIC],
3404         color: '#4a90d9',
3405         hasProperties: true,
3406         defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
3407         resizable: true,
3408         minWidth: 150,
3409         minHeight: 50
3410     },
3411     'and': {
3412         name: 'И',
3413         inputs: 2, // По умолчанию 2, но может быть изменено
3414         outputs: 1,
3415         inputLabels: ['A', 'B'],
3416         inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3417         outputLabels: ['результат'],
3418         outputTypes: [SIGNAL_TYPE.LOGIC],
3419         color: '#a855f7',
3420         hasProperties: true, // ← Теперь есть свойства (для изменения количества
3421         // входов)
3422         resizable: true,
3423         minWidth: 120,
3424         minHeight: 80,
3425         hasConditionPort: true,
3426         conditionPortType: SIGNAL_TYPE.LOGIC,
3427         defaultProps: {
```

```
3427         inputCount: 2 // ← Новое свойство
3428     }
3429 },
3430 'or': {
3431     name: 'ИЛИ',
3432     inputs: 2, // По умолчанию 2
3433     outputs: 1,
3434     inputLabels: ['A', 'B'],
3435     inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3436     outputLabels: ['результат'],
3437     outputTypes: [SIGNAL_TYPE.LOGIC],
3438     color: '#a855f7',
3439     hasProperties: true, // Теперь есть свойства
3440     resizable: true,
3441     minWidth: 120,
3442     minHeight: 80,
3443     hasConditionPort: true,
3444     conditionPortType: SIGNAL_TYPE.LOGIC,
3445     defaultProps: {
3446         inputCount: 2 // ← Новое свойство
3447     }
3448 },
3449 'not': {
3450     name: 'НЕ',
3451     inputs: 1,
3452     outputs: 1,
3453     inputLabels: ['A'],
3454     inputTypes: [SIGNAL_TYPE.LOGIC],
3455     outputLabels: ['¬A'],
3456     outputTypes: [SIGNAL_TYPE.LOGIC],
3457     color: '#a855f7',
3458     hasProperties: true,
3459     resizable: true,
3460     minWidth: 100,
3461     minHeight: 60,
3462     hasConditionPort: true,
3463     conditionPortType: SIGNAL_TYPE.LOGIC
3464 },
3465 'if': {
3466     name: 'ЕСЛИ',
3467     inputs: 2,
3468     outputs: 1, // ← Только один выход!
3469     inputLabels: ['A', 'B'],
3470     inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
3471     outputLabels: ['результат'], // ← Просто результат
3472     outputTypes: [SIGNAL_TYPE.LOGIC], // ← Выход типа LOGIC
3473     color: '#e94560',
3474     hasProperties: true,
3475     defaultProps: { operator: '=' },
3476     resizable: true,
3477     minWidth: 120,
3478     minHeight: 80,
3479     hasConditionPort: true,
3480     conditionPortType: SIGNAL_TYPE.LOGIC
3481 },
3482 'separator': { // ← НОВЫЙ ЭЛЕМЕНТ
3483     name: 'Сепаратор',
3484     inputs: 1,
3485     outputs: 2,
3486     inputLabels: ['сигнал'],
3487     inputTypes: [SIGNAL_TYPE.LOGIC],
3488     outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
3489     outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE], // ← TRUE и FALSE
3490     color: '#f59e0b',
3491     hasProperties: true,
```

```
3492     resizable: true,
3493     minWidth: 120,
3494     minHeight: 80,
3495     hasConditionPort: true,
3496     conditionPortType: SIGNAL_TYPE.LOGIC
3497   },
3498   'const': {
3499     name: 'Константа',
3500     inputs: 0,
3501     outputs: 1,
3502     outputLabels: ['out'],
3503     outputTypes: [SIGNAL_TYPE.NUMERIC],
3504     color: '#3b82f6',
3505     hasProperties: true,
3506     defaultProps: { value: 0 },
3507     resizable: true,
3508     minWidth: 120,
3509     minHeight: 60,
3510     hasConditionPort: true,
3511     conditionPortType: SIGNAL_TYPE.LOGIC
3512   },
3513   'formula': {
3514     name: 'Формула',
3515     inputs: 2,
3516     outputs: 1,
3517     inputLabels: ['in1', 'in2'],
3518     inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
3519     outputLabels: ['результат'],
3520     outputTypes: [SIGNAL_TYPE.NUMERIC],
3521     color: '#f59e0b',
3522     hasProperties: true,
3523     resizable: true,
3524     minWidth: 140,
3525     minHeight: 80,
3526     defaultProps: {
3527       expression: '',
3528       inputCount: 2
3529     },
3530     hasConditionPort: true,
3531     conditionPortType: SIGNAL_TYPE.LOGIC
3532   },
3533   'output': {
3534     name: 'Выход',
3535     inputs: 1,
3536     outputs: 0,
3537     inputLabels: ['сигнал'],
3538     inputTypes: [SIGNAL_TYPE.ANY],
3539     color: '#10b981',
3540     hasProperties: true,
3541     defaultProps: { label: 'Выход', outputGroup: '' },
3542     resizable: true,
3543     minWidth: 150,
3544     minHeight: 60,
3545   }, // ← важно, если предыдущий элемент не заканчивается запятой
3546   'group': {
3547     name: 'Группа',
3548     inputs: 0,
3549     outputs: 0,
3550     color: '#6b7280',
3551     resizable: true,
3552     minWidth: 200,
3553     minHeight: 120,
3554     hasProperties: true,
3555     defaultProps: { title: 'Группа' }
3556 }
```

```
3557 };
3558
3559 const VIEWPORT_CONFIG = {
3560     minZoom: 0.1,
3561     maxZoom: 3,
3562     zoomStep: 0.1,
3563     panSpeed: 1,
3564     canvasWidth: 5000,
3565     canvasHeight: 5000
3566 };
3567
3568 const MINIMAP_CONFIG = {
3569     width: 200,
3570     height: 150,
3571     padding: 10
3572 };
3573
3574 connections.js:
3575 /**
3576 * Модуль работы с соединениями
3577 */
3578
3579 const Connections = {
3580     /**
3581     * Настройка обработчиков порта
3582     */
3583     setupPortHandlers(port) {
3584         port.addEventListener('mousedown', (e) => {
3585             e.stopPropagation();
3586
3587             if (port.classList.contains('output')) {
3588                 const elemId = port.dataset.element;
3589                 const portName = port.dataset.port;
3590                 const signalType = getOutputPortType(elemId, portName);
3591
3592                 AppState.connectingFrom = {
3593                     element: elemId,
3594                     port: portName
3595                 };
3596                 AppState.connectingFromType = signalType;
3597
3598                 this.highlightCompatiblePorts(signalType);
3599
3600                 const svg = document.getElementById('connections-svg');
3601                 const startPos = this._getPortCanvasCenter(port);
3602
3603                 AppState.tempLine = document.createElementNS('http://www.w3.org/2000/
3604                 svg', 'path');
3605                 AppState.tempLine.setAttribute('class', 'temp-connection');
3606                 AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L ${
3607                 startPos.x} ${startPos.y}`);
3608                 svg.appendChild(AppState.tempLine);
3609             }
3610         });
3611
3612         port.addEventListener('mouseup', (e) => {
3613             e.stopPropagation();
3614             e.preventDefault();
3615
3616             if (AppState.connectingFrom && port.classList.contains('input')) {
3617                 const toElement = port.dataset.element;
3618                 const toPortName = port.dataset.port;
3619                 const inputType = getInputPortType(toElement, toPortName);
```

```
3620         if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
3621             this.clearConnectionState();
3622             return;
3623         }
3624
3625         if (AppState.connectingFrom.element !== toElement) {
3626             const targetElem = AppState.elements[toElement];
3627             const allowMultipleInputs = targetElem?.type === 'output';
3628
3629             const exists = AppState.connections.some(c =>
3630                 c.toElement === toElement && c.toPort === toPortName
3631             );
3632
3633             if (!exists || allowMultipleInputs) {
3634                 AppState.connections.push({
3635                     fromElement: AppState.connectingFrom.element,
3636                     fromPort: AppState.connectingFrom.port,
3637                     toElement,
3638                     toPort: toPortName,
3639                     signalType: AppState.connectingFromType
3640                 });
3641
3642                 port.classList.add('connected');
3643                 this.drawConnections();
3644                 this.clearConnectionState();
3645                 return;
3646             }
3647         }
3648     }
3649
3650     this.clearConnectionState();
3651 });
3652
3653 port.addEventListener('mouseenter', () => {
3654     if (AppState.connectingFrom && port.classList.contains('input')) {
3655         const toPortName = port.dataset.port;
3656         const inputType = getInputPortType(port.dataset.element, toPortName);
3657
3658         if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
3659             if (AppState.tempLine) {
3660                 AppState.tempLine.classList.add('invalid');
3661             }
3662         }
3663     }
3664 });
3665
3666 port.addEventListener('mouseleave', () => {
3667     if (AppState.tempLine) {
3668         AppState.tempLine.classList.remove('invalid');
3669     }
3670 });
3671 },
3672 /**
3673 * Подсветка совместимых портов
3674 */
3675 highlightCompatiblePorts(signalType) {
3676     document.querySelectorAll('.port.input').forEach(port => {
3677         const inputType = getInputPortType(port.dataset.element,
3678         port.dataset.port);
3679
3680         if (areTypesCompatible(signalType, inputType)) {
3681             port.classList.add('compatible-highlight');
3682         } else {
3683             port.classList.add('incompatible');
```

```
3684         }
3685     });
3686 },
3687 /**
3688 * Очистка состояния соединения
3689 */
3690 clearConnectionState() {
3691     if (AppState.tempLine) {
3692         AppState.tempLine.remove();
3693         AppState.tempLine = null;
3694     }
3695     AppState.connectingFrom = null;
3696     AppState.connectingFromType = null;
3697
3698     document.querySelectorAll('.port').forEach(port => {
3699         port.classList.remove('compatible-highlight', 'incompatible');
3700     });
3701 },
3702 /**
3703 * Отрисовка временной линии соединения
3704 */
3705 drawTempConnection(e) {
3706     if (!AppState.tempLine || !AppState.connectingFrom) return;
3707
3708     const fromElem = document.getElementById(AppState.connectingFrom.element);
3709     if (!fromElem) return;
3710
3711     const fromPort = fromElem.querySelector(`[data-port="${AppState.connectingFrom.port}"]`);
3712     if (!fromPort) return;
3713
3714     const startPos = this._getPortCanvasCenter(fromPort);
3715     const endPos = screenToCanvas(e.clientX, e.clientY);
3716
3717     const horizontalDist = Math.abs(endPos.x - startPos.x);
3718     const controlDist = Math.max(horizontalDist * 0.4, 50);
3719
3720     // Тянем всегда от выхода (вектор 1, 0)
3721     const cx1 = startPos.x + controlDist;
3722     const cy1 = startPos.y;
3723
3724     // Вторая точка контроля для плавности за курсором
3725     const cx2 = endPos.x - controlDist;
3726     const cy2 = endPos.y;
3727
3728     AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
3729     AppState.tempLine.setAttribute('fill', 'none');
3730 },
3731
3732 /**
3733 * Отрисовка всех соединений
3734 */
3735 drawConnections() {
3736     const svg = document.getElementById('connections-svg');
3737
3738     // 1. Очистка старых линий
3739     svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
3740
3741     // 2. Сброс визуального состояния портов
3742     document.querySelectorAll('.port.connected').forEach(port => {
3743         port.classList.remove('connected');
3744     });
3745 }
```

```
3747
3748     // 3. Перебор всех соединений из AppState
3749     AppState.connections.forEach(conn => {
3750         const fromElem = document.getElementById(conn.fromElement);
3751         const toElem = document.getElementById(conn.toElement);
3752
3753         if (!fromElem || !toElem) return;
3754
3755         const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
3756         const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
3757
3758         if (!fromPort || !toPort) return;
3759
3760         fromPort.classList.add('connected');
3761         toPort.classList.add('connected');
3762
3763         const startPos = this._getPortCanvasCenter(fromPort);
3764         const endPos = this._getPortCanvasCenter(toPort);
3765
3766         if (!startPos || !endPos) return;
3767
3768         // Расстояние для изгиба кривой
3769         const horizontalDist = Math.abs(endPos.x - startPos.x);
3770         const verticalDist = Math.abs(endPos.y - startPos.y);
3771         const controlDist = Math.max(horizontalDist * 0.4, 50);
3772
3773         // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
3774         let d;
3775         let cx1 = startPos.x;
3776         let cy1 = startPos.y;
3777         let cx2 = endPos.x;
3778         let cy2 = endPos.y;
3779
3780         // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
3781         cx1 = startPos.x + controlDist;
3782         cy1 = startPos.y;
3783
3784         // ВХОД (Target):
3785         if (conn.toPort === 'cond-0') {
3786             // Технический порт: Касательная (0, 1) в декартовой (вверх)
3787             // В экранных координатах Y инвертирован, поэтому отнимаем от Y
3788             cx2 = endPos.x;
3789             cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
3790         } else {
3791             // Обычный вход: Касательная (-1, 0) -> Слева направо
3792             cx2 = endPos.x - controlDist;
3793             cy2 = endPos.y;
3794         }
3795
3796         d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
3797         ${endPos.y}`;
3798
3799         const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
3800         path.setAttribute('d', d);
3801         path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
3802
3803         // --- ЛОГИКА ЦВЕТА (Классы) ---
3804         let cssClass = 'connection';
3805         const type = conn.signalType;
3806
3807         // Приоритет новым типам сигналов
3808         if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
3809         else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
3810         else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
3811         else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
```

```
3811         else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
3812
3813         path.setAttribute('class', cssClass);
3814
3815         // Обработчики событий
3816         path.style.pointerEvents = 'stroke';
3817         path.style.cursor = 'pointer';
3818         path.addEventListener('click', () => this.handleConnectionClick(conn));
3819
3820         svg.appendChild(path);
3821     });
3822
3823     if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
3824         Outputs.updateOutputStatus();
3825     }
3826     Viewport.updateMinimap();
3827 },
3828 /**
3829 * Обработка клика по соединению (удаление)
3830 */
3831 handleConnectionClick(conn) {
3832     if (confirm('Удалить соединение?')) {
3833         AppState.connections = AppState.connections.filter(c =>
3834             !(c.fromElement === conn.fromElement &&
3835                 c.fromPort === conn.fromPort &&
3836                 c.toElement === conn.toElement &&
3837                 c.toPort === conn.toPort)
3838         );
3839
3840         this.drawConnections();
3841     }
3842 },
3843 /**
3844 * Получение центра порта в координатах Canvas
3845 */
3846 _getPortCanvasCenter(portEl) {
3847     if (!portEl) return null;
3848
3849     const rect = portEl.getBoundingClientRect();
3850     return screenToCanvas(
3851         rect.left + rect.width / 2,
3852         rect.top + rect.height / 2
3853     );
3854 }
3855 };
3856 };
3857
3858 elements.js:
3859 /**
3860 * Модуль работы с элементами схемы
3861 */
3862
3863 const Elements = {
3864     /**
3865      * Генерация HTML для элемента
3866      */
3867      createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
3868          const config = ELEMENT_TYPES[elemType];
3869          if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
3870
3871          const safe = (value, fallback = '') => (value === null || value ===
3872 undefined) ? fallback : String(value);
3873          const w = width ?? config.minWidth ?? 120;
3874          const h = height ?? config.minHeight ?? 60;
```

```
3875
3876     const getPortClass = (signalType, direction) => {
3877         const base = direction === 'output' ? 'port output' : 'port input';
3878         if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
3879         if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
3880         return `${base} any-port`;
3881     };
3882
3883     // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
3884     innerHTML
3885     // Она тут остается, но ее результат не встраивается в HTML-строку
3886     // напрямую, кроме формулы
3887     const buildConditionPortHTML = () => {
3888         return `
3889             <div class="condition-port-wrapper">
3890                 <div class="condition-port-label">условие</div>
3891                 <div class="port input condition-port"
3892                     data-port="cond-0"
3893                     data-element="${elemId}"
3894                     data-signal-type="${SIGNAL_TYPE.LOGIC}"
3895                     title="Техническое условие">
3896                 </div>
3897             </div>`;
3898     };
3899
3900     const buildInputPorts = (count, types = [], labels = []) => {
3901         let html = '';
3902         for (let i = 0; i < count; i++) {
3903             const type = types[i] ?? types[types.length - 1] ??
3904             SIGNAL_TYPE.ANY;
3905             html += `<div class="${getPortClass(type, 'input')}" data-
3906             port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
3907             || `Вход ${i+1}`}"></div>`;
3908         }
3909         return html;
3910     };
3911
3912     const buildOutputPorts = (count, types = [], labels = []) => {
3913         let html = '';
3914         for (let i = 0; i < count; i++) {
3915             const type = types[i] ?? types[types.length - 1] ??
3916             SIGNAL_TYPE.ANY;
3917             html += `<div class="${getPortClass(type, 'output')}" data-
3918             port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
3919             || `Выход ${i+1}`}"></div>`;
3920         }
3921         return html;
3922     };
3923
3924     const resizeHandles = config.resizable ? `<div class="resize-handle
3925     handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
3926     direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
3927     '';
3928
3929     // hasCondClass будет добавляться в createElement
3930     // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
3931     '';
3932
3933     let innerHTML = '';
3934
3935     if (elemType === 'input-signal') {
3936         const name = safe(props.name, 'Сигнал');
3937         const type = props.signalType || SIGNAL_TYPE.NUMBER;
3938         const symbol = type === SIGNAL_TYPE.LOGIC ? '☒' : '☒';
3939         innerHTML =
```

```
3928             <div class="element-header" style="background:$
3929             {config.color};">Источник</div>
3930                 <div class="element-body">
3931                     <div class="element-symbol">
3932                         <span class="input-signal-icon">${symbol}</span>
3933                         <span class="input-signal-name">${name}</span>
3934                     </div>
3935                     <div class="ports-right">
3936                         ${buildOutputPorts(1, [type], ['Выход'])}
3937                     </div>`;
3938     }
3939     else if (elemType === 'const') {
3940         innerHTML = `
3941             <div class="element-header" style="background:$
3942             {config.color};">Константа</div>
3943                 <div class="element-body">
3944                     <div class="element-symbol">${props.value ?? 0}</div>
3945                     <div class="ports-right">
3946                         ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
3947                     </div>`;
3948     }
3949     else if (elemType === 'separator') {
3950         innerHTML = `
3951             <div class="element-header" style="background:$
3952             {config.color};">Сепаратор</div>
3953                 <div class="element-body">
3954                     <div class="ports-left">${buildInputPorts(1,
3955 config.inputTypes, config.inputLabels)}</div>
3956                     <div class="element-symbol">/x</div>
3957                     <div class="ports-right">
3958                         <div class="port output logic-port true-port" data-
3959                         port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
3960                         title="ИСТИНА"></div>
3961                         <div class="port output logic-port false-port" data-
3962                         port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
3963                         title="ЛОЖЬ"></div>
3964                     </div>
3965                 </div>`;
3966     }
3967     else if (elemType === 'and' || elemType === 'or') {
3968         const gateSymbol = elemType === 'and' ? 'Λ' : '∨';
3969         const inputCount = props.inputCount || config.defaultProps?.inputCount
3970         || 2;
3971
3972         // Генерируем динамические входы
3973         let inputsHTML = '';
3974         for (let i = 0; i < inputCount; i++) {
3975             inputsHTML += `<div class="port input logic-port" data-port="in-$
3976             ${i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход $
3977             ${i+1}"></div>`;
3978         }
3979
3980         innerHTML = `
3981             <div class="element-header" style="background:${config.color};">$
3982             {config.name}</div>
3983                 <div class="element-body">
3984                     <div class="ports-left">
3985                         ${inputsHTML}
3986                     </div>
3987                     <div class="element-symbol">${gateSymbol}</div>
3988                     <div class="ports-right">
3989                         <div class="port output logic-port" data-port="out-0"
3990                         data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></div>
```

```
div>
3980                     </div>
3981                     `;
```

```
        }
3982     else if (elemType === 'if') {
3983         const op = safe(props.operator, '=');
3984         innerHTML =
3985             <div class="element-header" style="background:$
3986 {config.color};">Условие</div>
3987             <div class="element-body">
3988                 <div class="ports-left">${buildInputPorts(2,
3989 config.inputTypes, config.inputLabels)}</div>
3990                     <div class="element-symbol">${op}</div>
3991                     <div class="ports-right">
3992                         ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
3993                     </div>
3994                 </div>`;
3995     else if (elemType === 'not') {
3996         innerHTML =
3997             <div class="element-header" style="background:$
3998 {config.color};">НЕ</div>
3999             <div class="element-body">
4000                 <div class="ports-left">${buildInputPorts(1,
4001 [SIGNAL_TYPE.LOGIC], ['A'])}</div>
4002                     <div class="element-symbol">¬</div>
4003                     <div class="ports-right">
4004                         ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
4005                     </div>`;
4006     else if (elemType === 'formula') {
4007         const inputCount = props.inputCount || config.defaultProps?.inputCount
4008         || config.inputs || 2;
4009         const expression = safe(props.expression);
4010         const displayExpression = expression
4011             ? (expression.length > 12 ? `${expression.slice(0, 12)}...` :
4012             : 'f(x)';
4013
4014         innerHTML = `
4015             ${buildConditionPortHTML()}
4016             <div class="element-header" style="background:$
4017 {config.color};">Формула</div>
4018             <div class="element-body">
4019                 <div class="ports-left">${buildInputPorts(inputCount,
4020 config.inputTypes, config.inputLabels)}</div>
4021                     <div class="element-symbol">${displayExpression}</div>
4022                     <div class="ports-right">
4023                         ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER],
4024 ['Результат'])}
4025                     </div>
4026                 </div>`;
4027             }
4028     else if (elemType === 'output') {
4029         innerHTML =
4030             <div class="element-header" style="background:$
4031 {config.color};">Выход</div>
4032             <div class="element-body">
4033                 <div class="ports-left">
4034                     ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
4035                 </div>
4036                 <div class="element-symbol">${safe(props.label, 'Выход')}</
4037 div>
4038                 <div class="ports-right"></div>
```

```
4033                     `
```

```
4034             `
```

```
4035         `
```

```
4036     else if (elementType === 'group') {
4037         const title = props.title || 'Группа';
4038         innerHTML = `
4039             <div class="group-content">
4040                 <div class="group-title">${title}</div>
4041             </div>`;
4042     }
4043
4044     else { // Для любых других (fallback)
4045         innerHTML = `
4046             <div class="element-header" style="background:${config.color};">${config.name}</div>
4047                 <div class="element-body">
4048                     <div class="ports-left">${buildInputPorts(config.inputs || 0,
4049 config.inputTypes, config.inputLabels)}</div>
4050                     <div class="element-symbol">${config.name}</div>
4051                     <div class="ports-right">
4052                         ${buildOutputPorts(config.outputs || 0,
4053 config.outputTypes, config.outputLabels)}
4054                     </div>
4055                 </div>`;
4056
4057
4058     const html = `
4059         <div class="element ${elementType}" id="${elemId}"
4060             style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;
4061 data-type="${elementType}">
4062             ${innerHTML}
4063             ${commentHtml}
4064             ${resizeHandles}
4065         </div>`;
4066
4067         return { html, width: w, height: h };
4068     },
4069
4070     /**
4071      * Добавление элемента
4072      */
4073     addElement(elementType, x, y, props = {}, elemId = null, customWidth = null,
4074 customHeight = null) {
4075         const config = ELEMENT_TYPES[elementType];
4076         if (!config) {
4077             console.error(`Неизвестный тип элемента: ${elementType}`);
4078             return null;
4079         }
4080
4081         if (!elemId) {
4082             elemId = `${elementType}_${++AppState.elementCounter}`;
4083         }
4084
4085         let width = customWidth;
4086         let height = customHeight;
4087
4088         if (width === null || width === undefined) {
4089             width = config.minLength || 140;
4090         }
4091         if (height === null || height === undefined) {
4092             height = config.minLength || 70;
4093         }
4094     }
4095 }
```

```
4092
4093         try {
4094             const result = this.createElementHTML(elemType, elemId, x, y, props,
4095             width, height);
4096             if (!result || !result.html) {
4097                 console.error('createElementHTML вернул пустой результат');
4098                 return null;
4099             }
4100
4101             const workspace = document.getElementById('workspace');
4102             const wrapper = document.createElement('div');
4103             wrapper.innerHTML = result.html.trim();
4104             const element = wrapper.firstChild;
4105             if (!element) {
4106                 console.error('Не удалось создать DOM элемент из HTML');
4107                 return null;
4108             }
4109
4110             // Добавляем класс для отступа
4111             if (config.hasConditionPort) {
4112                 element.classList.add('has-condition-port');
4113             }
4114
4115             workspace.appendChild(element);
4116
4117             AppState.elements[elemId] = {
4118                 id: elemId,
4119                 type: elemType,
4120                 x,
4121                 y,
4122                 width: result.width || width,
4123                 height: result.height || height,
4124                 props: { ... (config.defaultProps || {}), ... (props || {}) }
4125             };
4126
4127             // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ
4128             // ЕГО В HTML)
4129             if (config.hasConditionPort && elemType !== 'formula') {
4130                 const condPortWrapper = document.createElement('div');
4131                 condPortWrapper.innerHTML =
4132                     `<div class="condition-port-wrapper">
4133                         <div class="condition-port-label">условие</div>
4134                         <div class="port input condition-port"
4135                             data-port="cond-0"
4136                             data-element="${elemId}"
4137                             data-signal-type="${SIGNAL_TYPE.LOGIC}"
4138                             title="Техническое условие">
4139                             </div>
4140                         </div>`;
4141             element.prepend(condPortWrapper.firstChild); // Вставляем в
4142             самое начало элемента
4143         }
4144
4145         this.setupElementHandlers(elemId); // Передаем ID элемента
4146
4147         // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
4148         // element.querySelectorAll('.port').forEach(port => {
4149         //     Connections.setupPortHandlers(port);
4150         // });
4151
4152         Connections.drawConnections(); // Перерисовываем соединения, чтобы
4153         // учесть новые порты
4154         Viewport.updateMinimap();
4155         return elemId;
```

```
4153         } catch (err) {
4154             console.error(`Ошибка при добавлении элемента ${elementType}:`, err);
4155             return null;
4156         }
4157     },
4158
4159     /**
4160      * Обновление входов логического элемента (AND, OR)
4161      */
4162     updateLogicGateInputs(elemId, inputCount) {
4163         const elem = document.getElementById(elemId);
4164         if (!elem) return;
4165
4166         const portsLeft = elem.querySelector('.ports-left');
4167         if (!portsLeft) return;
4168
4169         // Удаляем соединения к портам, которые больше не существуют
4170         AppState.connections = AppState.connections.filter(c => {
4171             if (c.toElement === elemId && c.toPort.startsWith('in-')) {
4172                 const portNum = parseInt(c.toPort.split('-')[1], 10);
4173                 return portNum < inputCount;
4174             }
4175             return true;
4176         });
4177
4178         // Генерируем новые входы
4179         let inputsHTML = '';
4180         for (let i = 0; i < inputCount; i++) {
4181             inputsHTML += `
4182                 <div class="port input logic-port"
4183                     data-port="in-${i}"
4184                     data-element="${elemId}"
4185                     data-signal-type="${SIGNAL_TYPE.LOGIC}"
4186                     title="Вход ${i+1}">
4187                     </div>
4188                 `;
4189         }
4190         portsLeft.innerHTML = inputsHTML;
4191
4192         // Переподключаем обработчики
4193         portsLeft.querySelectorAll('.port').forEach(port =>
4194             Connections.setupPortHandlers(port)
4195         );
4196
4197         Connections.drawConnections();
4198     },
4199
4200     /**
4201      * Удаление элемента
4202      */
4203     deleteElement(elemId) {
4204         AppState.connections = AppState.connections.filter(c =>
4205             c.fromElement !== elemId && c.toElement !== elemId
4206         );
4207
4208         const elem = document.getElementById(elemId);
4209         if (elem) elem.remove();
4210
4211         delete AppState.elements[elemId];
4212
4213         if (AppState.selectedElement === elemId) {
4214             AppState.selectedElement = null;
4215         }
4216
4217         Connections.drawConnections();
```

```
4218     Viewport.updateMinimap();
4219 },
4220 /**
4221 * Выделение элемента
4222 */
4223 selectElement(elemId) {
4224     if (AppState.selectedElement) {
4225         const oldElem = document.getElementById(AppState.selectedElement);
4226         if (oldElem) oldElem.classList.remove('selected');
4227     }
4228
4229     AppState.selectedElement = elemId;
4230     const elem = document.getElementById(elemId);
4231     if (elem) elem.classList.add('selected');
4232
4233     const elemData = AppState.elements[elemId];
4234     if (elemData) {
4235         document.getElementById('selection-info').textContent =
4236             `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
4237     }
4238 },
4239
4240 /**
4241 * Снять выделение
4242 */
4243 deselectAll() {
4244     if (AppState.selectedElement) {
4245         const elem = document.getElementById(AppState.selectedElement);
4246         if (elem) elem.classList.remove('selected');
4247         AppState.selectedElement = null;
4248     }
4249     document.getElementById('selection-info').textContent = '';
4250 },
4251
4252 /**
4253 * Настройка обработчиков элемента
4254 */
4255 setupElementHandlers(elemId) {
4256     try {
4257         const elem = document.getElementById(elemId);
4258         if (!elem) return;
4259
4260         elem.addEventListener('mousedown', (e) => {
4261             if (e.target.classList.contains('port')) return;
4262             if (e.target.classList.contains('resize-handle')) return;
4263
4264             e.preventDefault();
4265             e.stopPropagation();
4266
4267             this.selectElement(elemId);
4268
4269             AppState.draggingElement = elemId;
4270             const canvasPos = screenToCanvas(e.clientX, e.clientY);
4271             const elemData = AppState.elements[elemId];
4272             AppState.dragOffset.x = canvasPos.x - elemData.x;
4273             AppState.dragOffset.y = canvasPos.y - elemData.y;
4274         });
4275
4276         elem.addEventListener('dblclick', (e) => {
4277             if (e.target.classList.contains('port')) return;
4278             const config = ELEMENT_TYPES[AppState.elements[elemId].type];
4279             if (config?.hasProperties) {
4280                 Modal.showPropertiesModal(elemId);
4281             }
4282         });
4283     }
4284 }
```

```
4283         });
4284
4285         elem.addEventListener('contextmenu', (e) => {
4286             e.preventDefault();
4287             this.showContextMenu(e.clientX, e.clientY, elemId);
4288         });
4289
4290         const handles = elem.querySelectorAll('.resize-handle');
4291         handles.forEach(handle => this.setupResizeHandlers(handle, elemId));
4292
4293         const ports = elem.querySelectorAll('.port');
4294         ports.forEach(port => Connections.setupPortHandlers(port));
4295
4296     } catch (err) {
4297         console.error('setupElementHandlers error for', elemId, err);
4298     }
4299 },
4300
4301 /**
4302 * Контекстное меню
4303 */
4304 showContextMenu(x, y, elemId) {
4305     const menu = document.getElementById('context-menu');
4306     menu.style.left = `${x}px`;
4307     menu.style.top = `${y}px`;
4308     menu.style.display = 'block';
4309     menu.dataset.elementId = elemId;
4310 },
4311
4312 /**
4313 * Настройка resize
4314 */
4315 setupResizeHandlers(handle, elemId) {
4316     handle.addEventListener('mousedown', (e) => {
4317         e.stopPropagation();
4318         e.preventDefault();
4319
4320         const elemData = AppState.elements[elemId];
4321
4322         AppState.resizing = {
4323             elemId,
4324             handle: handle.dataset.direction,
4325             startX: e.clientX,
4326             startY: e.clientY,
4327             startWidth: elemData.width,
4328             startHeight: elemData.height,
4329             startLeft: elemData.x,
4330             startTop: elemData.y
4331         };
4332     });
4333 },
4334
4335 /**
4336 * Обработка resize
4337 */
4338 handleResize(e) {
4339     if (!AppState.resizing) return;
4340
4341     const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
4342     startTop } = AppState.resizing;
4343     const elem = document.getElementById(elemId);
4344     const elemData = AppState.elements[elemId];
4345     const config = ELEMENT_TYPES[elemData.type];
4346
4347     const dx = (e.clientX - startX) / AppState.viewport.zoom;
```

```
4347     const dy = (e.clientY - startY) / AppState.viewport.zoom;
4348
4349     let newWidth = startWidth;
4350     let newHeight = startHeight;
4351     let newLeft = startLeft;
4352     let newTop = startTop;
4353
4354     if (handle.includes('e')) {
4355         newWidth = Math.max(config.minWidth, startWidth + dx);
4356     }
4357     if (handle.includes('w')) {
4358         newWidth = Math.max(config.minWidth, startWidth - dx);
4359         newLeft = startLeft + (startWidth - newWidth);
4360     }
4361     if (handle.includes('s')) {
4362         newHeight = Math.max(config.minLength, startHeight + dy);
4363     }
4364     if (handle.includes('n')) {
4365         newHeight = Math.max(config.minLength, startHeight - dy);
4366         newTop = startTop + (startHeight - newHeight);
4367     }
4368
4369     elem.style.width = `${newWidth}px`;
4370     elem.style.height = `${newHeight}px`;
4371     elem.style.left = `${newLeft}px`;
4372     elem.style.top = `${newTop}px`;
4373
4374     elemData.width = newWidth;
4375     elemData.height = newHeight;
4376     elemData.x = newLeft;
4377     elemData.y = newTop;
4378
4379     Connections.drawConnections();
4380 },
4381
4382 /**
4383 * Обработка перетаскивания элемента
4384 */
4385 handleDrag(e) {
4386     if (!AppState.draggingElement) return;
4387
4388     const canvasPos = screenToCanvas(e.clientX, e.clientY);
4389     const elemId = AppState.draggingElement;
4390     const elemData = AppState.elements[elemId];
4391     if (!elemData) return;
4392
4393     const newX = canvasPos.x - AppState.dragOffset.x;
4394     const newY = canvasPos.y - AppState.dragOffset.y;
4395     const dx = newX - elemData.x;
4396     const dy = newY - elemData.y;
4397
4398     // если выделено несколько
4399     const group = AppState.selectedElements && AppState.selectedElements.length >
1
4400         ? AppState.selectedElements
4401         : [elemId];
4402
4403     for (const id of group) {
4404         const elData = AppState.elements[id];
4405         if (!elData) continue;
4406         elData.x += dx;
4407         elData.y += dy;
4408         const el = document.getElementById(id);
4409         if (el) {
4410             el.style.left = elData.x + 'px';
```

```
4411         el.style.top = elData.y + 'px';
4412     }
4413 }
4414
4415     Connections.drawConnections();
4416 },
4417
4418 /**
4419 * Обновление входов формулы
4420 */
4421 updateFormulaInputs(elemId, inputCount) {
4422     const elem = document.getElementById(elemId);
4423     if (!elem) return;
4424
4425     const portsLeft = elem.querySelector('.ports-left');
4426     if (!portsLeft) return;
4427
4428     AppState.connections = AppState.connections.filter(c => {
4429         if (c.toElement === elemId && c.toPort.startsWith('in-')) {
4430             const portNum = parseInt(c.toPort.split('-')[1], 10);
4431             return portNum < inputCount;
4432         }
4433         return true;
4434     });
4435
4436     let inputsHTML = '';
4437     for (let i = 0; i < inputCount; i++) {
4438         inputsHTML += `
4439             <div class="port input any-port"
4440                 data-port="in-${i}"
4441                 data-element="${elemId}"
4442                 data-signal-type="${SIGNAL_TYPE.ANY}"
4443                 title="in${i} (Любой)">
4444                 </div>
4445             `;
4446     }
4447     portsLeft.innerHTML = inputsHTML;
4448
4449     portsLeft.querySelectorAll('.port').forEach(port =>
4450         Connections.setupPortHandlers(port)
4451     );
4452
4453     Connections.drawConnections();
4454 },
4455
4456 /**
4457 * Рассчитать оптимальный размер элемента на основе количества портов
4458 */
4459 calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
4460     const elem = AppState.elements[elemId];
4461     if (!elem) return null;
4462
4463     const config = ELEMENT_TYPES[elem.type];
4464     if (!config || !config.resizable) return null;
4465
4466     // Базовая высота
4467     let baseHeight = config.minLength || 60;
4468
4469     // Каждый порт требует примерно 25-30px высоты
4470     const portSpacing = 28;
4471     const maxPorts = Math.max(inputCount, outputCount);
4472
4473     // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
4474     const additionalHeight = (maxPorts - 1) * portSpacing;
4475     const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
```

```
4476             return newHeight;
4477         },
4478
4479     /**
4480      * Обновление размера элемента при изменении портов
4481      */
4482     updateElementSize(elemId) {
4483         const elem = document.getElementById(elemId);
4484         const elemData = AppState.elements[elemId];
4485
4486         if (!elem || !elemData) return;
4487
4488         const config = ELEMENT_TYPES[elemData.type];
4489         if (!config || !config.resizable) return;
4490
4491         let inputCount = 0;
4492         let outputCount = config.outputs || 1;
4493
4494         // Определяем количество входов
4495         if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
4496 'formula') {
4497             inputCount = elemData.props.inputCount || config.inputs || 2;
4498         } else {
4499             inputCount = config.inputs || 0;
4500         }
4501
4502         // Рассчитываем новую высоту
4503         const newHeight = this.calculateOptimalHeight(elemId, inputCount,
4504 outputCount);
4505
4506         if (newHeight && newHeight !== elemData.height) {
4507             elemData.height = newHeight;
4508             elem.style.height = `${newHeight}px`;
4509
4510             // Перерисовываем соединения, т.к. изменился размер элемента
4511             Connections.drawConnections();
4512             Viewport.updateMinimap();
4513         }
4514
4515     };
4516 }
4517
4518 modal.js:
4519 /**
4520  * Модуль модальных окон
4521 */
4522
4523 const Modal = {
4524     /**
4525      * Инициализация модальных окон
4526      */
4527     init() {
4528         // Модальное окно свойств элемента
4529         document.getElementById('modal-save').addEventListener('click', () => {
4530             this.saveElementProperties();
4531         });
4532
4533         document.getElementById('modal-cancel').addEventListener('click', () => {
4534             this.hideModal('modal-overlay');
4535         });
4536
4537         document.getElementById('modal-overlay').addEventListener('click', (e) => {
4538             if (e.target.id === 'modal-overlay') {
```

```
4539         this.hideModal('modal-overlay');
4540     }
4541   });
4542 
4543   // Модальное окно свойств проекта
4544   document.getElementById('project-modal-save').addEventListener('click', () =>
{
4545     this.saveProjectProperties();
4546   });
4547 
4548   document.getElementById('project-modal-cancel').addEventListener('click', () => {
4549     this.hideModal('project-modal-overlay');
4550   });
4551 
4552   document.getElementById('project-modal-overlay').addEventListener('click', (e) => {
4553     if (e.target.id === 'project-modal-overlay') {
4554       this.hideModal('project-modal-overlay');
4555     }
4556   }, );
4557 
4558 /**
4559 * Показать модальное окно
4560 */
4561 showModal(modalId) {
4562   document.getElementById(modalId).style.display = 'flex';
4563 },
4564 
4565 /**
4566 * Скрыть модальное окно
4567 */
4568 hideModal(modalId) {
4569   document.getElementById(modalId).style.display = 'none';
4570 },
4571 
4572 /**
4573 * Показать свойства элемента
4574 */
4575 showPropertiesModal(elemId) {
4576   const elemData = AppState.elements[elemId];
4577   const elemType = elemData.type;
4578   const props = elemData.props;
4579   const config = ELEMENT_TYPES[elemType];
4580 
4581   const modalOverlay = document.getElementById('modal-overlay');
4582   const modalTitle = document.getElementById('modal-title');
4583   const modalContent = document.getElementById('modal-content');
4584 
4585   modalTitle.textContent = `Свойства: ${config.name}`;
4586 
4587   let contentHTML = '';
4588 
4589   if (elemType === 'input-signal') {
4590     const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
4591 
4592     contentHTML = `
4593       <div class="modal-row">
4594         <label>Название сигнала:</label>
4595         <input type="text" id="prop-name" value="${props.name || ''}" 
placeholder="Например: 10LBA..." />
4596         <small style="color:#999;">
4597           Поиск по маске через * (например: *MAA*CP*)
4598         </small>
4599       
```

```
4600     <div id="signal-filter-results"
4601         style="max-height:160px; overflow-y:auto; background:#0f3460; border-
4602         radius:5px; margin-top:6px; display:none;">
4603     </div>
4604
4605     <div class="modal-row">
4606         <label>Описание сигнала:</label>
4607         <textarea id="prop-description" readonly>${props.description || ''}</textarea>
4608     </div>
4609
4610     <div class="modal-row">
4611         <label>Тип сигнала:</label>
4612         <select id="prop-signal-type">
4613             <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
4614 'selected' : ''}>Числовой</option>
4615             <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
4616 'selected' : ''}>Логический</option>
4617         </select>
4618     </div>
4619     `;
4620
4621     // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
4622     // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
4623     // (Смотри пункт 2 – небольшая вставка в конце showPropertiesModal)
4624 } else if (elementType === 'if') {
4625     contentHTML =
4626         <div class="modal-row">
4627             <label>Оператор сравнения:</label>
4628             <select id="prop-operator">
4629                 <option value="=" ${props.operator === '=' ? 'selected' : ''}>=
4630                 (равно)</option>
4631                 <option value=">" ${props.operator === '>' ? 'selected' : ''}>>
4632                 (больше)</option>
4633                 <option value="<" ${props.operator === '<' ? 'selected' : ''}><
4634                 (меньше)</option>
4635                 <option value=">=" ${props.operator === '>=' ? 'selected' :
4636                   ''}>= (больше или равно)</option>
4637                 <option value="<=" ${props.operator === '<=' ? 'selected' :
4638                   ''}>= (меньше или равно)</option>
4639                 <option value="!=" ${props.operator === '!='
4640                   ? 'selected' : ''}>!= (не равно)</option>
4641             </select>
4642         </div>
4643     `;
4644 } else if (elementType === 'and' || elementType === 'or') {
4645     contentHTML =
4646         <div class="modal-row">
4647             <label>Количество входов:</label>
4648             <input type="number" id="prop-input-count" value="$
4649             {props.inputCount || 2}" min="2" max="10">
4650             </div>
4651             <div class="modal-row">
4652                 <p style="color: #aaa; font-size: 12px;">
4653                     Измените количество входных портов для этого логического
4654                     элемента.
4655                     Лишние соединения будут автоматически удалены.
4656                 </p>
4657             </div>
4658     `;
4659 } else if (elementType === 'const') {
4660     contentHTML =
4661         <div class="modal-row">
4662             <label>Значение:</label>
```

```
4653             <input type="number" id="prop-value" value="${props.value ?? 0}"  
4654     step="any">  
4655         `;  
4656     }  
4657     else if (elementType === 'group') {  
4658         contentHTML = `  
4659             <div class="modal-row">  
4660                 <label>Название группы:</label>  
4661                 <input type="text" id="prop-title" value="${props.title || 'Группа'}">  
4662             </div>`;  
4663     }  
4664  
4665     else if (elementType === 'formula') {  
4666         let signalsHTML = '';  
4667         AppState.connections.forEach(conn => {  
4668             if (conn.toElement === elemId) {  
4669                 const fromElem = AppState.elements[conn.fromElement];  
4670                 if (fromElem) {  
4671                     const signalName = fromElem.props?.name || fromElem.id;  
4672                     signalsHTML += `<div class="signal-item" data-signal="$  
4673 {signalName}">${signalName} (${conn.toPort})</div>`;  
4674             }  
4675         });  
4676  
4677         // ... (где-то выше код сбора signalsHTML) ...  
4678  
4679         contentHTML = `  
4680             <div class="modal-row">  
4681                 <label>Количество входов:</label>  
4682                 <input type="number" id="prop-input-count" value="$  
4683 {props.inputCount || 2}" min="1" max="10">  
4684             </div>  
4685  
4686             <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->  
4687             <div style="display: flex; gap: 15px; margin-bottom: 15px; height:  
4688                 140px;">  
4689                 <!-- Левая колонка: Сигналы -->  
4690                 <div style="flex: 1; display: flex; flex-direction: column;">  
4691                     <label style="margin-bottom: 5px; display:block;">Входные  
4692                     сигналы:</label>  
4693                     <div class="signal-list" id="signal-list" style="flex: 1;  
4694                         overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px  
4695                         solid #4a90d9;">  
4696                         ${signalsHTML || '<div style="color:#888;padding:5px;">Нет  
4697                         сигналов</div>'}  
4698                     </div>  
4699                 </div>  
4700  
4701                 <!-- Правая колонка: Шаблоны -->  
4702                 <div style="flex: 1; display: flex; flex-direction: column;">  
4703                     <label style="margin-bottom: 5px; display:block;">Шаблоны:</  
4704                     label>  
4705                     <div class="signal-list" id="template-list" style="flex: 1;  
4706                         overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px  
4707                         solid #4a90d9;">  
4708                         <div style="color:#888;padding:5px;">Загрузка...</div>  
4709                     </div>  
4710                 </div>  
4711             </div>  
4712  
4713             <!-- Нижний блок: Поле формулы (во всю ширину) -->  
4714             <div class="modal-row">  
4715                 <label>Выражение формулы:</label>
```

```
4707             <textarea id="prop-expression"
4708                 style="width: 100%; min-height: 80px; font-family:
4709                 monospace; font-size: 14px; line-height: 1.4;" 
4710                 spellcheck="false">${props.expression || ''}</textarea>
4711                 <small style="color:#999; display:block; margin-top:4px;">
4712                     Двойной клик по сигналу или шаблону вставит его в позицию
4713                     курсора (или заменит выделенный текст).
4714                     </small>
4715             `;
4716         }
4717         if (!contentHTML) {
4718             contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных
4719             свойств.</div>`;
4720             }
4721             contentHTML += `
4722                 <div class="modal-row">
4723                     <label>Комментарий:</label>
4724                     <textarea id="prop-comment" placeholder="Комментарий к элементу...">$
4725             {props.comment || ''}</textarea>
4726                 </div>
4727             `;
4728
4729             modalContent.innerHTML = contentHTML;
4730             if (elemType === 'formula') {
4731                 const listEl = document.getElementById('template-list');
4732                 (async () => {
4733                     try {
4734                         const data = await Settings.fetchFormulaTemplates();
4735                         const items = data.templates || [];
4736                         if (!items.length) {
4737                             listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет
4738                             шаблонов</div>';
4739                         }
4740                     return;
4741                 })
4742                     .join('');
4743
4744                     listEl.querySelectorAll('.template-item').forEach(div => {
4745                         div.addEventListener('dblclick', () => {
4746                             const insert = div.dataset.insert;
4747                             const textarea = document.getElementById('prop-expression');

4748                             // БЫЛО: textarea.value += ...;
4749                             // СТАЛО:
4750                             insertAtCursor(textarea, insert);
4751                         });
4752                     });
4753                     } catch (e) {
4754                         console.error(e);
4755                         listEl.innerHTML = '<div style="color:#888;padding:5px;">Ошибка
4756                         загрузки</div>';
4757                     }
4758                 })());
4759             }
4760
4761
4762             // --- post init handlers (когда DOM модалки уже существует) ---
4763             if (elemType === 'input-signal') {
4764                 const input = document.getElementById('prop-name');
```

```
4765     const results = document.getElementById('signal-filter-results');
4766     const descField = document.getElementById('prop-description');
4767
4768     let timer = null;
4769
4770     const renderList = (items) => {
4771         if (!items || items.length === 0) {
4772             results.innerHTML = '<div style="color:#666;padding:6px;">Нет
4773             совпадений</div>';
4774             results.style.display = 'block';
4775             return;
4776         }
4777
4778         results.innerHTML = items.map(s => `
4779             <div class="signal-result-item"
4780                 style="padding:6px 8px; cursor:pointer; border-bottom:1px solid
4781                 rgba(255,255,255,0.08);">
4782                 <div style="font-weight:600;">${s.Tagname}</div>
4783                 <div style="color:#aaa; font-size:11px;">${s.Description} || ''</
4784             div>
4785             </div>
4786             `).join('');
4787
4788         results.style.display = 'block';
4789
4790         results.querySelectorAll('.signal-result-item').forEach((div, i) => {
4791             div.addEventListener('click', () => {
4792                 const chosen = items[i];
4793                 input.value = chosen.Tagname;
4794                 descField.value = chosen.Description || '';
4795                 results.style.display = 'none';
4796             });
4797         });
4798     };
4799
4800     const search = async () => {
4801         const mask = (input.value || '').trim();
4802
4803         // Показываем список только если пользователь реально использует маску
4804         if (!mask.includes('*')) {
4805             results.style.display = 'none';
4806             return;
4807         }
4808
4809         results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</
4810         div>';
4811
4812         results.style.display = 'block';
4813
4814         try {
4815             // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
4816             const data = await Settings.fetchSignals(mask, 50);
4817             renderList(data.items || []);
4818         } catch (e) {
4819             results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка
4820             загрузки сигналов</div>';
4821             results.style.display = 'block';
4822             console.error(e);
4823         }
4824
4825         input.addEventListener('input', () => {
4826             clearTimeout(timer);
4827             timer = setTimeout(search, 200); // debounce
4828         });
4829     };
4830
4831     input.addEventListener('input', () => {
4832         clearTimeout(timer);
4833         timer = setTimeout(search, 200); // debounce
4834     });
4835
4836     input.addEventListener('input', () => {
4837         clearTimeout(timer);
4838         timer = setTimeout(search, 200); // debounce
4839     });
4840
4841     input.addEventListener('input', () => {
4842         clearTimeout(timer);
4843         timer = setTimeout(search, 200); // debounce
4844     });
4845
4846     input.addEventListener('input', () => {
4847         clearTimeout(timer);
4848         timer = setTimeout(search, 200); // debounce
4849     });
4850
4851     input.addEventListener('input', () => {
4852         clearTimeout(timer);
4853         timer = setTimeout(search, 200); // debounce
4854     });
4855
4856     input.addEventListener('input', () => {
4857         clearTimeout(timer);
4858         timer = setTimeout(search, 200); // debounce
4859     });
4860
4861     input.addEventListener('input', () => {
4862         clearTimeout(timer);
4863         timer = setTimeout(search, 200); // debounce
4864     });
4865
4866     input.addEventListener('input', () => {
4867         clearTimeout(timer);
4868         timer = setTimeout(search, 200); // debounce
4869     });
4870
4871     input.addEventListener('input', () => {
4872         clearTimeout(timer);
4873         timer = setTimeout(search, 200); // debounce
4874     });
4875
4876     input.addEventListener('input', () => {
4877         clearTimeout(timer);
4878         timer = setTimeout(search, 200); // debounce
4879     });
4880
4881     input.addEventListener('input', () => {
4882         clearTimeout(timer);
4883         timer = setTimeout(search, 200); // debounce
4884     });
4885
4886     input.addEventListener('input', () => {
4887         clearTimeout(timer);
4888         timer = setTimeout(search, 200); // debounce
4889     });
4890
4891     input.addEventListener('input', () => {
4892         clearTimeout(timer);
4893         timer = setTimeout(search, 200); // debounce
4894     });
4895
4896     input.addEventListener('input', () => {
4897         clearTimeout(timer);
4898         timer = setTimeout(search, 200); // debounce
4899     });
4900
4901     input.addEventListener('input', () => {
4902         clearTimeout(timer);
4903         timer = setTimeout(search, 200); // debounce
4904     });
4905
4906     input.addEventListener('input', () => {
4907         clearTimeout(timer);
4908         timer = setTimeout(search, 200); // debounce
4909     });
4910
4911     input.addEventListener('input', () => {
4912         clearTimeout(timer);
4913         timer = setTimeout(search, 200); // debounce
4914     });
4915
4916     input.addEventListener('input', () => {
4917         clearTimeout(timer);
4918         timer = setTimeout(search, 200); // debounce
4919     });
4920
4921     input.addEventListener('input', () => {
4922         clearTimeout(timer);
4923         timer = setTimeout(search, 200); // debounce
4924     });
4925
4926     input.addEventListener('input', () => {
4927         clearTimeout(timer);
4928         timer = setTimeout(search, 200); // debounce
4929     });
4930
4931     input.addEventListener('input', () => {
4932         clearTimeout(timer);
4933         timer = setTimeout(search, 200); // debounce
4934     });
4935
4936     input.addEventListener('input', () => {
4937         clearTimeout(timer);
4938         timer = setTimeout(search, 200); // debounce
4939     });
4940
4941     input.addEventListener('input', () => {
4942         clearTimeout(timer);
4943         timer = setTimeout(search, 200); // debounce
4944     });
4945
4946     input.addEventListener('input', () => {
4947         clearTimeout(timer);
4948         timer = setTimeout(search, 200); // debounce
4949     });
4950
4951     input.addEventListener('input', () => {
4952         clearTimeout(timer);
4953         timer = setTimeout(search, 200); // debounce
4954     });
4955
4956     input.addEventListener('input', () => {
4957         clearTimeout(timer);
4958         timer = setTimeout(search, 200); // debounce
4959     });
4960
4961     input.addEventListener('input', () => {
4962         clearTimeout(timer);
4963         timer = setTimeout(search, 200); // debounce
4964     });
4965
4966     input.addEventListener('input', () => {
4967         clearTimeout(timer);
4968         timer = setTimeout(search, 200); // debounce
4969     });
4970
4971     input.addEventListener('input', () => {
4972         clearTimeout(timer);
4973         timer = setTimeout(search, 200); // debounce
4974     });
4975
4976     input.addEventListener('input', () => {
4977         clearTimeout(timer);
4978         timer = setTimeout(search, 200); // debounce
4979     });
4980
4981     input.addEventListener('input', () => {
4982         clearTimeout(timer);
4983         timer = setTimeout(search, 200); // debounce
4984     });
4985
4986     input.addEventListener('input', () => {
4987         clearTimeout(timer);
4988         timer = setTimeout(search, 200); // debounce
4989     });
4990
4991     input.addEventListener('input', () => {
4992         clearTimeout(timer);
4993         timer = setTimeout(search, 200); // debounce
4994     });
4995
4996     input.addEventListener('input', () => {
4997         clearTimeout(timer);
4998         timer = setTimeout(search, 200); // debounce
4999     });
4999
```

```
4825 // опционально: закрывать список кликом вне
4826 document.addEventListener('mousedown', (e) => {
4827     if (!results.contains(e.target) && e.target !== input) {
4828         results.style.display = 'none';
4829     }
4830 }, { once: true });
4831
4832 modalOverlay.dataset.elementId = elemId;
4833 this.showModal('modal-overlay');
4834
4835 // Функция для умной вставки текста в позицию курсора
4836 const insertAtCursor = (field, text) => {
4837     if (!field) return;
4838
4839     // Получаем позиции выделения
4840     const startPos = field.selectionStart;
4841     const endPos = field.selectionEnd;
4842     const currentValue = field.value;
4843
4844     // Вставляем текст: (текст до) + (новый текст) + (текст после)
4845     field.value = currentValue.substring(0, startPos) +
4846         text +
4847             currentValue.substring(endPos, currentValue.length);
4848
4849     // Возвращаем фокус и ставим курсор сразу после вставленного текста
4850     field.focus();
4851     const newCursorPosition = startPos + text.length;
4852     field.setSelectionRange(newCursorPosition, newCursorPosition);
4853 };
4854
4855 // Обработчик вставки сигналов для формулы
4856 if (elementType === 'formula') {
4857     document.querySelectorAll('.signal-item').forEach(item => {
4858         item.addEventListener('dblclick', () => {
4859             const signal = item.dataset.signal;
4860             const textarea = document.getElementById('prop-expression');
4861
4862             // БЫЛО: textarea.value += signal;
4863             // СТАЛО:
4864             insertAtCursor(textarea, signal);
4865         });
4866     });
4867 }
4868
4869 /**
4870 * Сохранить свойства элемента
4871 */
4872 /**
4873 * Сохранить свойства элемента
4874 */
4875
4876 saveElementProperties() {
4877     try {
4878         const modalOverlay = document.getElementById('modal-overlay');
4879         const elemId = modalOverlay.dataset.elementId;
4880         const elemData = AppState.elements[elemId];
4881         const elem = document.getElementById(elemId);
4882         if (!elemData) {
4883             alert('⚠ Элемент не найден – возможно, он был удалён или
переименован.');
4884             console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
4885             this.hideModal('modal-overlay');
4886             return;
4887         }
4888     }
```

```
4889     const elemType = elemData.type;
4890
4891     if (elemType === 'input-signal') {
4892         const name = document.getElementById('prop-name').value || 'Сигнал';
4893         const description = document.getElementById('prop-description').value
4894         || '';
4895         const signalType = document.getElementById('prop-signal-type').value;
4896
4897         const oldSignalType = elemData.props.signalType;
4898         elemData.props.name = name;
4899         elemData.props.description = description;
4900         elemData.props.signalType = signalType;
4901
4901     if (oldSignalType !== signalType) {
4902         AppState.connections = AppState.connections.filter(conn => {
4903             if (conn.fromElement === elemId) {
4904                 const toPortIndex = parseInt(conn.toPort.split('-')[1]);
4905                 const inputType = getInputPortType(conn.toElement,
4906                     toPortIndex);
4907
4908                 return areTypesCompatible(signalType, inputType);
4909             }
4910             return true;
4911         });
4912
4913         const { html } = Elements.createElementHTML(
4914             elemType, elemId, elemData.x, elemData.y, elemData.props,
4915             elemData.width, elemData.height
4916         );
4917         elem.outerHTML = html;
4918
4919         Elements.setupElementHandlers(elemId);
4920         Connections.drawConnections();
4921     } else if (elemType === 'if') {
4922         const operator = document.getElementById('prop-operator').value;
4923         elemData.props.operator = operator;
4924         const symbol = elem.querySelector('.element-symbol');
4925         if (symbol) symbol.textContent = operator;
4926
4927     } else if (elemType === 'const') {
4928         const value = parseFloat(document.getElementById('prop-value').value)
4929         || 0;
4930         elemData.props.value = value;
4931         const symbol = elem.querySelector('.element-symbol');
4932         if (symbol) symbol.textContent = String(value);
4933
4934     } else if (elemType === 'formula') {
4935         const expression = document.getElementById('prop-expression').value;
4936         const inputCount = parseInt(document.getElementById('prop-input-
4937         count').value) || 2;
4938
4939         elemData.props.expression = expression;
4940         elemData.props.inputCount = inputCount;
4941
4942         const symbol = elem.querySelector('.element-symbol');
4943         if (symbol) {
4944             symbol.textContent = expression.length > 12 ? `$
4945             {expression.slice(0, 12)}...` : (expression || 'f(x)');
4946         }
4947
4948         Elements.updateFormulaInputs(elemId, inputCount);
4949         Elements.updateElementSize(elemId); // ← Добавляем это
4950     } else if (elemType === 'and' || elemType === 'or') {
4951         const inputCount = parseInt(document.getElementById('prop-input-
4952         count').value) || 2;
```

```
4947         elemData.props.inputCount = inputCount;
4948
4949         Elements.updateLogicGateInputs(elemId, inputCount);
4950         Elements.updateElementSize(elemId); // ← Добавляем это
4951
4952         const symbol = elem.querySelector('.element-symbol');
4953         if (symbol) {
4954             symbol.textContent = elemType === 'and' ? 'Λ' : 'ν';
4955         }
4956
4957     } else if (elemType === 'output') {
4958         const label = document.getElementById('prop-label').value || 'Выход';
4959         const outputGroup = document.getElementById('prop-output-group').value
4960         || '';
4961
4962         elemData.props.label = label;
4963         elemData.props.outputGroup = outputGroup;
4964
4965         const symbol = elem.querySelector('.element-symbol');
4966         if (symbol) symbol.textContent = label;
4967     }
4968     else if (elemType === 'group') {
4969         const title = document.getElementById('prop-title').value || 'Группа';
4970         elemData.props.title = title;
4971         const titleEl = elem.querySelector('.group-title');
4972         if (titleEl) titleEl.textContent = title;
4973     }
4974     const commentEl = document.getElementById('prop-comment');
4975     if (commentEl) elemData.props.comment = commentEl.value || '';
4976
4977     this.hideModal('modal-overlay');
4978
4979     } catch (error) {
4980         console.error('🔴 Ошибка при сохранении свойств:', error);
4981         alert('Ошибка сохранения: ' + error.message);
4982     },
4983
4984 /**
4985 * Показать свойства проекта
4986 */
4987 showProjectPropertiesModal() {
4988     const content = document.getElementById('project-modal-content');
4989     const project = AppState.project;
4990
4991     // Генерируем HTML для списка выходов только если модуль загружен
4992     let outputsHtml = '';
4993     if (typeof Outputs !== 'undefined' && AppState.outputs) {
4994         const logicalOutputsHtml = AppState.outputs.logical.length > 0
4995             ? AppState.outputs.logical.map(output =>
4996                 <div class="output-item"
4997                     data-element-id="${output.elementId}"
4998                     onmouseenter="Outputs.highlightOutput('${output.elementId}', true)"
4999                     onmouseleave="Outputs.highlightOutput('${output.elementId}', false)"
5000                     onclick="Outputs.navigateToOutput('${output.elementId}')";
5001                     Modal.hideModal('project-modal-overlay');">
5002                         <span class="output-icon">${output.portLabel === 'Да' ? '✅' : '🔴'}</span>
5003                         <span class="output-name">${output[elementName]}</span>
5004                         <span class="output-port">→ ${output.portLabel}</span>
5005                     </div>
5006                 `).join('')
5007             : '<div class="no-outputs">Нет логических выходов</div>';
5008     }
```

```
5007
5008     const numericOutputsHtml = AppState.outputs.numeric.length > 0
5009     ? AppState.outputs.numeric.map(output =>
5010         <div class="output-item numeric"
5011             data-element-id={`${output.elementId}`}
5012             onmouseenter="Outputs.highlightOutput('${output.elementId}', true)"
5013             onmouseleave="Outputs.highlightOutput('${output.elementId}', false)"
5014             onclick="Outputs.navigateToOutput('${output.elementId}')";
5015             Modal.hideModal('project-modal-overlay');">
5016                 <span class="output-icon"></span>
5017                 <span class="output-name">${output[elementName]}</span>
5018                 <span class="output-port">> значение</span>
5019             `).join('')
5020             : '<div class="no-outputs">Нет числовых выходов</div>';
5021
5022     outputsHtml =
5023         <div class="modal-row">
5024             <label>Выходные сигналы схемы:</label>
5025             <div class="outputs-container">
5026                 <div class="outputs-section">
5027                     <div class="outputs-section-title">
5028                         <span class="section-icon"> X</span>
5029                         Логические выходы (${AppState.outputs.logical.length})
5030                     </div>
5031                     <div class="outputs-list">
5032                         ${logicalOutputsHtml}
5033                     </div>
5034                 </div>
5035                 <div class="outputs-section">
5036                     <div class="outputs-section-title">
5037                         <span class="section-icon"> Y</span>
5038                         Числовые выходы (${AppState.outputs.numeric.length})
5039                     </div>
5040                     <div class="outputs-list">
5041                         ${numericOutputsHtml}
5042                     </div>
5043                 </div>
5044             </div>
5045             <div class="outputs-hint">
5046                  Выходами автоматически становятся элементы, чьи выходные
5047             порты не подключены к другим элементам.
5048                 Кликните на выход, чтобы перейти к нему на схеме.
5049             </div>
5050         </div>
5051     ;
5052 }
5053
5054 content.innerHTML =
5055     <div class="modal-row">
5056         <label>Код проекта:</label>
5057         <input type="text" id="project-code" value="${project.code || ''}"
5058             placeholder="Уникальный идентификатор">
5059     </div>
5060
5061     <div class="modal-row">
5062         <label>Тип проекта:</label>
5063         <div class="project-type-selector">
5064             <div class="project-type-btn ${project.type ===
PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
5065                 <div class="type-icon"> Z</div>
5066                 <div class="type-name">Параметр</div>
5067                 <div class="type-desc">Вычисляемое значение</div>
```

```
5066             </div>
5067             <div class="project-type-btn ${project.type ===
5068                 PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
5069                 <div class="type-icon"></div>
5070                 <div class="type-name">Правило</div>
5071                 <div class="type-desc">Логическое условие</div>
5072             </div>
5073         </div>
5074
5075         <div id="parameter-fields" class="conditional-fields ${project.type ===
5076             PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
5077             <div class="modal-row">
5078                 <label>Размерность:</label>
5079                 <input type="text" id="project-dimension" value="$
{project.dimension || ''}" placeholder="Например: м/с, кг, °C">
5080             </div>
5081         </div>
5082
5083         <div id="rule-fields" class="conditional-fields ${project.type ===
5084             PROJECT_TYPE.RULE ? 'visible' : ''}">
5085             <div class="modal-row">
5086                 <label>Возможная причина:</label>
5087                 <textarea id="project-possible-cause" placeholder="Описание
возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
5088             </div>
5089             <div class="modal-row">
5090                 <label>Методические указания:</label>
5091                 <textarea id="project-guidelines" placeholder="Инструкции и
рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
5092             </div>
5093         </div>
5094     `;
5095
5096     // Обработчики переключения типа
5097     content.querySelectorAll('.project-type-btn').forEach(btn => {
5098         btn.addEventListener('click', () => {
5099             content.querySelectorAll('.project-type-btn').forEach(b =>
5100                 b.classList.remove('active'));
5101                 btn.classList.add('active');
5102
5103                 const type = btn.dataset.type;
5104                 document.getElementById('parameter-
fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
5105                 document.getElementById('rule-fields').classList.toggle('visible',
type === PROJECT_TYPE.RULE);
5106             });
5107         });
5108
5109         this.showModal('project-modal-overlay');
5110     },
5111
5112     /**
5113      * Сохранить свойства проекта
5114      */
5115     saveProjectProperties() {
5116         const activeTypeBtn = document.querySelector('.project-type-btn.active');
5117         const type = activeTypeBtn ? activeTypeBtn.dataset.type :
5118             PROJECT_TYPE.PARAMETER;
5119
5120         AppState.project.code = document.getElementById('project-code').value;
5121         AppState.project.type = type;
5122     }
5123 }
```

```
5121     if (type === PROJECT_TYPE.PARAMETER) {
5122         AppState.project.dimension = document.getElementById('project-
5123 dimension').value;
5124         AppState.project.possibleCause = '';
5125         AppState.project.guidelines = '';
5126     } else {
5127         AppState.project.dimension = '';
5128         AppState.project.possibleCause = document.getElementById('project-
5129 possible-cause').value;
5130         AppState.project.guidelines = document.getElementById('project-
5131 guidelines').value;
5132     }
5133 }
5134
5135 output.js:
5136 /**
5137 * Модуль управления выходными сигналами
5138 */
5140
5141 const Outputs = {
5142     /**
5143     * Обновление статуса выходных элементов
5144     * Вызывается при каждом изменении схемы
5145     */
5146     updateOutputStatus() {
5147         this.clearAllOutputHighlights();
5148         AppState.outputs.logical = [];
5149         AppState.outputs.numeric = [];
5150         updateFrameChildren();
5151
5152         // Обработка элементов-выходов
5153         Object.values(AppState.elements).forEach(elem => {
5154             if (!elem || elem.type !== 'output') return;
5155
5156             // Проверяем, к чему подключен вход этого выхода
5157             const inputConns = AppState.connections.filter(c =>
5158                 c.toElement === elem.id && c.toPort === 'in-0'
5159             );
5160
5161             // Каждое соединение к выходу – это отдельный выход
5162             inputConns.forEach((conn, index) => {
5163                 const fromElem = AppState.elements[conn.fromElement];
5164                 if (!fromElem) return;
5165
5166                 const outputType = conn.signalType;
5167                 const outputInfo = {
5168                     id: `${elem.id}_conn_${index}`,
5169                     elementId: elem.id,
5170                     sourceElementId: conn.fromElement,
5171                     sourcePort: conn.fromPort,
5172                     portIndex: 0,
5173                     portId: 'in-0',
5174                     type: outputType,
5175                     label: elem.props?.label || 'Выход',
5176                     elementType: 'output',
5177                     elementName: elem.props?.label || 'Выход',
5178                     name: elem.props?.label || 'Выход'
5179                 };
5180
5181                 if (outputType === SIGNAL_TYPE.LOGIC) {
5182                     AppState.outputs.logical.push(outputInfo);
5183                 }
5184             });
5185         });
5186     }
5187 }
```

```
5183         } else if (outputType === SIGNAL_TYPE.NUMBER) {
5184             AppState.outputs.numeric.push(outputInfo);
5185         }
5186
5187         // Подсветим входной порт
5188         this.highlightOutputPort(elem.id, 0, outputType);
5189     });
5190
5191     this.updateOutputCounter();
5192 },
5193
5194 /**
5195 * Очистка всех выделений выходов
5196 */
5197 clearAllOutputHighlights() {
5198     document.querySelectorAll('.port.output-active').forEach(port => {
5199         port.classList.remove('output-active');
5200     });
5201
5202     document.querySelectorAll('.element.has-output').forEach(elem => {
5203         elem.classList.remove('has-output');
5204     });
5205
5206     document.querySelectorAll('.element.output-ambiguous').forEach(el =>
5207 el.classList.remove('output-ambiguous'));
5208     document.querySelectorAll('.element.output-missing').forEach(el =>
5209 el.classList.remove('output-missing'));
5210 }
5211
5212 /**
5213 * Выделение выходного порта
5214 */
5215 highlightOutputPort(elemId, portIndex, portType) {
5216     const elem = document.getElementById(elemId);
5217     if (!elem) return;
5218
5219     const port = elem.querySelector(`.port.output[data-port="out-${portIndex}]`);
5220     if (port) {
5221         port.classList.add('output-active');
5222     }
5223
5224     // Добавляем класс элементу (даёт общий визуал)
5225     elem.classList.add('has-output');
5226 },
5227
5228 /**
5229 * Обновление счётчика выходов в меню
5230 */
5231 updateOutputCounter() {
5232     const counter = document.getElementById('output-counter');
5233     if (counter) {
5234         const total = AppState.outputs.logical.length +
5235             AppState.outputs.numeric.length;
5236         counter.textContent = total;
5237         counter.style.display = total > 0 ? 'inline-block' : 'none';
5238     }
5239
5240 /**
5241 * Получить все выходы для сохранения в проект
5242 */
5243 getOutputsForSave() {
5244     // Сохраняем информацию о frame/inner для рамок
5245     return {
```

```
5245     logical: AppState.outputs.logical.map(o => ({
5246         id: o.id,
5247         elementId: o.elementId,
5248         frameId: o.frameId || null,
5249         innerElementId: o.innerElementId || null,
5250         portIndex: o.portIndex ?? o.innerPortIndex ?? null,
5251         portLabel: o.label
5252     })),
5253     numeric: AppState.outputs.numeric.map(o => ({
5254         id: o.id,
5255         elementId: o.elementId,
5256         frameId: o.frameId || null,
5257         innerElementId: o.innerElementId || null,
5258         portIndex: o.portIndex ?? o.innerPortIndex ?? null,
5259         portLabel: o.label
5260     }))
5261 },
5262 },
5263 /**
5264 * Подсветить конкретный выход (при наведении в списке)
5265 */
5266 highlightOutput(elementId, highlight = true) {
5267     const elem = document.getElementById(elementId);
5268     if (elem) {
5269         if (highlight) {
5270             elem.classList.add('output-highlighted');
5271         } else {
5272             elem.classList.remove('output-highlighted');
5273         }
5274     }
5275 },
5276 /**
5277 * Перейти к элементу выхода на схеме (elementId – фокусируемый элемент; для рамок
5278 это id рамки)
5279 */
5280 navigateToOutput(elementId) {
5281     const elemData = AppState.elements[elementId];
5282     if (!elemData) return;
5283
5284     // Центрируем viewport на элементе
5285     const container = document.getElementById('workspace-container');
5286     const rect = container.getBoundingClientRect();
5287
5288     const centerX = elemData.x + elemData.width / 2;
5289     const centerY = elemData.y + elemData.height / 2;
5290
5291     AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
5292     AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;
5293
5294     Viewport.updateTransform();
5295
5296     // Выделяем элемент
5297     Elements.selectElement(elementId);
5298
5299     // Временная подсветка
5300     this.highlightOutput(elementId, true);
5301     setTimeout(() => this.highlightOutput(elementId, false), 2000);
5302 }
5303 }
5304 };
5305
5306 project.js:
5307 /**
5308 */
```

```
5309 * Модуль управления проектом (сохранение, загрузка)
5310 */
5311
5312 // --- миграция id: '-' -> '_' с обновлением всех ссылок ---
5313 function migrateIdsDashToUnderscore() {
5314     const map = {};
5315
5316     // 1) собрать map старых id → новых
5317     Object.values(AppState.elements).forEach(el => {
5318         if (typeof el.id === 'string' && el.id.includes('-')) {
5319             map[el.id] = el.id.replace(/-/g, '_');
5320         }
5321     });
5322
5323     if (!Object.keys(map).length) return;
5324
5325     // 2) DOM id + data-element
5326     Object.entries(map).forEach(([oldId, newId]) => {
5327         const dom = document.getElementById(oldId);
5328         if (dom) dom.id = newId;
5329
5330         if (dom) {
5331             dom.querySelectorAll('[data-element]').forEach(p => {
5332                 if (p.dataset.element === oldId) p.dataset.element = newId;
5333             });
5334         }
5335     });
5336
5337     // 3) AppState.elements ключи
5338     Object.entries(map).forEach(([oldId, newId]) => {
5339         const el = AppState.elements[oldId];
5340         if (!el) return;
5341         el.id = newId;
5342         AppState.elements[newId] = el;
5343         delete AppState.elements[oldId];
5344     });
5345
5346     // 4) connections
5347     AppState.connections.forEach(c => {
5348         if (map[c.fromElement]) c.fromElement = map[c.fromElement];
5349         if (map[c.toElement]) c.toElement = map[c.toElement];
5350     });
5351
5352     // 5) формулы
5353     const escapeRegex = s => s.replace(/[^+?^${}()|[\]\\\]/g, '\\$&');
5354     Object.values(AppState.elements).forEach(el => {
5355         if (el.type === 'formula' && el.props?.expression) {
5356             let expr = el.props.expression;
5357             Object.entries(map).forEach(([oldId, newId]) => {
5358                 const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
5359 z0-9_])`, 'g');
5360                 expr = expr.replace(re, (m, p1) => ` ${p1}${newId}`);
5361             });
5362             el.props.expression = expr;
5363         }
5364     });
5365
5366     // 6) selected + modal
5367     if (map[AppState.selectedElement]) AppState.selectedElement =
5368         map[AppState.selectedElement];
5369     const modal = document.getElementById('modal-overlay');
5370     if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
5371         map[modal.dataset.elementId];
5372 }
```

```
5371 const Project = {
5372     /**
5373      * Инициализация
5374      */
5375      /**
5376      * Инициализация
5377      */
5378     init() {
5379         document.getElementById('btn-new').addEventListener('click', () =>
5380             this.newProject());
5381         document.getElementById('btn-save').addEventListener('click', () =>
5382             this.saveProject());
5383         document.getElementById('btn-load').addEventListener('click', () =>
5384             this.openProjectListModal());
5385         document.getElementById('btn-project-settings').addEventListener('click', () => {
5386             Modal.showProjectPropertiesModal();
5387         });
5388
5389         // Работа с модалкой выбора проекта
5390         this.projectList = [];
5391         this.filteredProjectList = [];
5392         this.selectedProjectFilename = null;
5393
5394         document.getElementById('project-cancel').addEventListener('click', () =>
5395             this.closeProjectListModal());
5396         document.getElementById('project-refresh').addEventListener('click', () =>
5397             this.refreshProjectList());
5398
5399         document.getElementById('project-load').addEventListener('click', () => {
5400             if (this.selectedProjectFilename) {
5401                 this.loadProjectFromList(this.selectedProjectFilename);
5402             }
5403         });
5404
5405         /**
5406          * Новый проект
5407          */
5408         newProject() {
5409             if (Object.keys(AppState.elements).length > 0) {
5410                 if (!confirm('Создать новый проект? Несохранённые изменения будут
5411 потерянны.')) {
5412                     return;
5413                 }
5414             }
5415             document.getElementById('workspace').innerHTML = '';
5416             document.getElementById('connections-svg').innerHTML = '';
5417
5418             setState();
5419             Viewport.updateTransform();
5420         },
5421
5422         /**
5423          * Запрос имени файла и загрузка с сервера
5424          */
5425         async loadProjectPrompt() {
5426             const filename = window.prompt(
5427                 "Введите имя файла проекта для загрузки (с сервера). Пример:
5428                 scheme_logic.json",
5429                 AppState.project.code ? `${AppState.project.code}`_$
```

```
5429     {AppState.project.type}.json` : "scheme_type.json"
5430     );
5431
5432     if (!filename) return; // Отмена
5433
5434     try {
5435         // Используем обертку из Settings.js для запроса к /api/project/load
5436         const data = await Settings.loadProject(filename);
5437
5438         // Если загрузка успешна, вызываем основную функцию обработки данных
5439         this._processLoadedData(data);
5440         alert(`Проект "${filename}" успешно загружен с сервера.`);
5441
5442     } catch (error) {
5443         console.error('Ошибка загрузки проекта:', error);
5444         alert(`Ошибка загрузки проекта: ${error.message}`);
5445     }
5446
5447 /**
5448 * Сохранение проекта
5449 */
5450     async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
5451     // 1. Проверяем свойства проекта
5452     if (!AppState.project.code) {
5453         Modal.showProjectPropertiesModal();
5454         alert('Пожалуйста, укажите код проекта перед сохранением.');
5455         return;
5456     }
5457
5458     // Обновляем размеры рамок перед сохранением
5459     updateFrameChildren();
5460     // ✅ нормализуем, даже если проект был открыт до фикса
5461     migrateIdsDashToUnderscore();
5462
5463     // ✅ подчистим связи прямо перед сохранением
5464     const exists = (id) => !!AppState.elements[id];
5465     AppState.connections = (AppState.connections || [])
5466         .map(c => {
5467             ...c,
5468             fromElement: exists(c.fromElement) ? c.fromElement :
5469             c.fromElement.replace(/-/g, '_'),
5470             toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g,
5471             '_')
5472         })
5473         .filter(c => exists(c.fromElement) && exists(c.toElement))
5474         .filter((c, idx, arr) => {
5475             const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5476             return arr.findIndex(x =>
5477                 `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5478             ) === idx;
5479         });
5480     // ✅ 1. Генерируем код заранее
5481     let generatedCode = '';
5482     if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
5483     {
5484         try {
5485             generatedCode = CodeGen.generate() || '';
5486         } catch (err) {
5487             console.error('Code generation failed:', err);
5488         }
5489     }
5490
5491     // 2. Сборка объекта проекта
5492     const project = {
```

```
5490     version: '1.0',
5491     project: AppState.project,
5492     elements: AppState.elements,
5493     connections: AppState.connections,
5494     counter: AppState.elementCounter,
5495     viewport: {
5496         zoom: AppState.viewport.zoom,
5497         panX: AppState.viewport.panX,
5498         panY: AppState.viewport.panY
5499     },
5500     code: generatedCode
5501 };
5502
5503     const filename = `${AppState.project.code} || 'scheme'}_${$`{AppState.project.type}`}.json`;
5504
5505     // 3. Сохранение на сервер
5506     try {
5507         await Settings.saveProject(filename, project);
5508         alert(`Проект успешно сохранен на сервере как: ${filename}`);
5509     } catch (error) {
5510         console.error('Ошибка сохранения проекта:', error);
5511         alert(`Ошибка сохранения проекта: ${error.message}`);
5512     }
5513 },
5514
5515     async showProjectList() {
5516         try {
5517             const result = await Settings.listProjects(); // нужно реализовать в
settings.js
5518             const list = result.projects || [];
5519
5520             if (list.length === 0) {
5521                 alert('Проекты в папке не найдены.');
5522                 return;
5523             }
5524
5525             const choice = window.prompt(
5526                 'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code} ||
p.filename} - ${p.description}`).join('\n') +
5527                     '\n\nВведите номер проекта для загрузки:',
5528                     '1'
5529             );
5530             const index = parseInt(choice, 10) - 1;
5531             if (isNaN(index) || !list[index]) return;
5532
5533             await this.loadProjectByFilename(list[index].filename);
5534         } catch (error) {
5535             console.error(error);
5536             alert('Не удалось получить список проектов: ' + error.message);
5537         }
5538     },
5539
5540     async loadProjectByFilename(filename) {
5541         try {
5542             const data = await Settings.loadProject(filename);
5543             this._processLoadedData(data);
5544             alert(`Проект "${filename}" загружен.`);
5545         } catch (error) {
5546             console.error(error);
5547             alert('Ошибка загрузки проекта: ' + error.message);
5548         }
5549     },
5550
5551     openProjectListModal() {
```

```
5552     const modal = document.getElementById('modal-project-list');
5553     modal.classList.remove('hidden');
5554     document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
5555     this.refreshProjectList();
5556   },
5557
5558   closeProjectListModal() {
5559     const modal = document.getElementById('modal-project-list');
5560     modal.classList.add('hidden');
5561     document.body.classList.remove('modal-open');
5562   },
5563
5564   async refreshProjectList() {
5565     const tbody = document.getElementById('project-list-body');
5566     tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка...</td></tr>`;
5567     try {
5568       const result = await Settings.listProjects();
5569       this.projectList = result.projects || [];
5570       this.filteredProjectList = [...this.projectList];
5571       this.renderProjectList();
5572     } catch (err) {
5573       console.error(err);
5574       tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: ${err.message}</td></tr>`;
5575     }
5576   },
5577
5578   renderProjectList() {
5579     const tbody = document.getElementById('project-list-body');
5580     const loadBtn = document.getElementById('project-load');
5581     loadBtn.disabled = true;
5582     this.selectedProjectFilename = null;
5583
5584     if (!this.filteredProjectList.length) {
5585       tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
5586       return;
5587     }
5588
5589     tbody.innerHTML = '';
5590     this.filteredProjectList.forEach((item) => {
5591       const tr = document.createElement('tr');
5592       tr.innerHTML =
5593         `<td>${item.filename}</td>
5594         <td>${item.code} || ''</td>
5595         <td>${item.description} || ''</td>
5596         <td>${item.type} || ''</td>
5597       `;
5598       tr.addEventListener('click', () => {
5599         this.highlightRow(tr);
5600         this.selectedProjectFilename = item.filename;
5601         loadBtn.disabled = false;
5602       });
5603       tr.addEventListener('dblclick', () => {
5604         this.highlightRow(tr);
5605         this.selectedProjectFilename = item.filename;
5606         loadBtn.disabled = false;
5607         this.loadProjectFromList(item.filename);
5608       });
5609       tbody.appendChild(tr);
5610     });
5611   },
5612 }
```

```
5613 highlightRow(row) {
5614     const tbody = row.parentElement;
5615     [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
5616     row.classList.add('selected');
5617 },
5618
5619
5620 // Фильтр по поисковой строке
5621 filterProjectList(query) {
5622     const q = (query || '').trim().toLowerCase();
5623     if (!q) {
5624         this.filteredProjectList = [...this.projectList];
5625     } else {
5626         this.filteredProjectList = this.projectList.filter((item) => {
5627             return [
5628                 item.filename,
5629                 item.code,
5630                 item.description,
5631                 item.type
5632             ].some((field) => (field || '').toLowerCase().includes(q));
5633         });
5634     }
5635     this.renderProjectList();
5636 },
5637
5638 async loadProjectFromList(filename) {
5639     try {
5640         const data = await Settings.loadProject(filename);
5641         this._processLoadedData(data);
5642         this.closeProjectListModal();
5643         alert(`Проект "${filename}" успешно загружен.`);
5644     } catch (error) {
5645         console.error(error);
5646         alert('Ошибка загрузки проекта: ' + error.message);
5647     }
5648 },
5649
5650
5651
5652
5653
5654
5655 /**
5656 * Загрузка проекта
5657 */
5658 _processLoadedData(data) {
5659     try {
5660         document.getElementById('workspace').innerHTML = '';
5661         document.getElementById('connections-svg').innerHTML = '';
5662         resetState();
5663
5664         if (data.project) {
5665             AppState.project = { ...AppState.project, ...data.project };
5666         }
5667
5668         AppState.elementCounter = data.counter || 0;
5669
5670         if (data.viewport) {
5671             AppState.viewport.zoom = data.viewport.zoom || 1;
5672             AppState.viewport.panX = data.viewport.panX || 0;
5673             AppState.viewport.panY = data.viewport.panY || 0;
5674         }
5675
5676         const elements = data.elements || {};
5677         Object.values(elements)
```

```
5678     .filter(e => e.type === 'output-frame')
5679     .forEach(elemData => {
5680       Elements.addElement(
5681         elemData.type,
5682         elemData.x,
5683         elemData.y,
5684         elemData.props,
5685         elemData.id,
5686         elemData.width,
5687         elemData.height
5688       );
5689     });
5690 
5691   Object.values(elements)
5692     .filter(e => e.type !== 'output-frame')
5693     .forEach(elemData => {
5694       Elements.addElement(
5695         elemData.type,
5696         elemData.x,
5697         elemData.y,
5698         elemData.props,
5699         elemData.id,
5700         elemData.width,
5701         elemData.height
5702       );
5703     });
5704 
5705   AppState.connections = data.connections || [];
5706 
5707   //  ВСТАВЬ ЭТОТ БЛОК СРАЗУ ЗДЕСЬ (до вычисления счётчика)
5708   // Object.values(AppState.elements).forEach(e => {
5709   //   if (typeof e.id === 'string') {
5710   //     e.id = e.id.replace(/-/g, '_');
5711   //   }
5712   //   if (e.props?.name) {
5713   //     e.props.name = e.props.name.replace(/-/g, '_');
5714   //   }
5715   // });
5716   //  конец добавленной секции
5717   //  Миграция id: '-' -> '_'
5718   migrateIdsDashToUnderscore();
5719   //  очистка соединений: удалить битые и дубликаты
5720   const exists = (id) => !AppState.elements[id];
5721 
5722   AppState.connections = (AppState.connections || [])
5723     // оставить только те, где оба конца реально существуют
5724     .filter(c => exists(c.fromElement) && exists(c.toElement))
5725     // убрать дубликаты
5726     .filter((c, idx, arr) => {
5727       const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5728       return arr.findIndex(x =>
5729         `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5730       ) === idx;
5731     });
5732 
5733 
5734   // корректно восстанавливаем счётчик
5735   const counterFromFile = Number(data.counter);
5736   AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
5737 
5738   const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
5739     if (!el?.id) return max;
5740     const match = String(el.id).match(/_(\d+)$/); // теперь хвост по
5741     подчёркиванию
5742     const num = match ? parseInt(match[1], 10) : NaN;
```

```
5742         return Number.isFinite(num) ? Math.max(max, num) : max;
5743     }, 0);
5744
5745     AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
5746
5747     Viewport.updateTransform();
5748     Connections.drawConnections();
5749     updateFrameChildren();
5750
5751 } catch (e) {
5752     alert('Ошибка обработки данных проекта: ' + e.message);
5753     console.error(e);
5754 }
5755 }
5756 };
5757
5758 settings.js:
5759
5760 const Settings = {
5761     config: null,
5762     templates: null,
5763
5764     async init() {
5765         // тянем настройки (не обязательно, но полезно)
5766         try {
5767             const r = await fetch('/api/settings');
5768             if (r.ok) this.config = await r.json();
5769         } catch (e) {
5770             console.warn('Settings load failed:', e);
5771         }
5772         try {
5773             const t = await this.fetchFormulaTemplates();
5774             this.templates = t.templates || [];
5775         } catch (e) {
5776             this.templates = [];
5777         }
5778     },
5779
5780     getTemplatesMap() {
5781         const map = {};
5782         (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
5783         return map;
5784     },
5785
5786     async fetchSignals(mask, limit = 50) {
5787         const url = `/api/signals?q=${encodeURIComponent(mask || '')}&limit=${encodeURIComponent(limit)}`;
5788         const r = await fetch(url);
5789         if (!r.ok) throw new Error('Failed to fetch signals');
5790         return await r.json(); // {items, total}
5791     },
5792     // ... в объекте Settings
5793
5794     async saveProject(filename, projectData) {
5795         if (!filename.endsWith('.json')) {
5796             filename += '.json';
5797         }
5798         const r = await fetch('/api/project/save', {
5799             method: 'POST',
5800             headers: { 'Content-Type': 'application/json' },
5801             body: JSON.stringify({
5802                 filename: filename,
5803                 content: projectData
5804             })
5805         });
5806     }
5807 }
```

```
5806     if (!r.ok) throw new Error('Failed to save project');
5807     return r.json();
5808 },
5809
5810 async listProjects() {
5811     const r = await fetch('/api/project/list');
5812     if (!r.ok) throw new Error('Failed to list projects');
5813     return r.json();
5814 },
5815
5816 async fetchFormulaTemplates() {
5817     const r = await fetch('/api/formula-templates');
5818     if (!r.ok) throw new Error('Failed to fetch formula templates');
5819     return await r.json(); // {templates:[...]}
5820 },
5821
5822 async loadProject(filename) {
5823     if (!filename.endsWith('.json')) {
5824         filename += '.json';
5825     }
5826     const r = await fetch(`/api/project/load/${encodeURIComponent(filename)}`);
5827     if (!r.ok) {
5828         if (r.status === 404) {
5829             throw new Error(`Project "${filename}" not found (404)`);
5830         }
5831         throw new Error('Failed to load project');
5832     }
5833     return r.json();
5834 }
5835
5836 // ...
5837 };
5838
5839 state.js:
5840 /**
5841 * Глобальное состояние приложения
5842 */
5843
5844 const AppState = {
5845     // Элементы схемы
5846     elements: {},
5847     connections: [],
5848     elementCounter: 0,
5849
5850     // Выделение
5851     selectedElement: null,
5852
5853     // Перетаскивание
5854     draggingElement: null,
5855     dragOffset: { x: 0, y: 0 },
5856     isDraggingFromPalette: false,
5857     dragPreview: null,
5858     dragType: null,
5859
5860     // Соединения
5861     connectingFrom: null,
5862     connectingFromType: null,
5863     tempLine: null,
5864
5865     // Resize
5866     resizing: null,
5867
5868     // Viewport (масштабирование и перемещение)
5869     viewport: {
5870         zoom: 1,
```

```
5871     panX: 0,
5872     panY: 0,
5873     isPanning: false,
5874     lastMouseX: 0,
5875     lastMouseY: 0
5876   },
5877
5878   // Свойства проекта
5879   project: {
5880     code: '',
5881     type: PROJECT_TYPE.PARAMETER,
5882     // Для параметра
5883     dimension: '',
5884     // Для правила
5885     possibleCause: '',
5886     guidelines: ''
5887   },
5888
5889   // Выходные сигналы (автоматически определяются)
5890   outputs: {
5891     logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
5892     numeric: []    // Числовые выходы (формулы)
5893   }
5894 };
5895
5896 /**
5897  * Сброс состояния
5898 */
5899 function resetState() {
5900   AppState.elements = {};
5901   AppState.connections = [];
5902   AppState.elementCounter = 0;
5903   AppState.selectedElement = null;
5904   AppState.draggingElement = null;
5905   AppState.connectingFrom = null;
5906   AppState.tempLine = null;
5907   AppState.resizing = null;
5908
5909   AppState.viewport = {
5910     zoom: 1,
5911     panX: 0,
5912     panY: 0,
5913     isPanning: false,
5914     lastMouseX: 0,
5915     lastMouseY: 0
5916   };
5917
5918   AppState.project = {
5919     code: '',
5920     type: PROJECT_TYPE.PARAMETER,
5921     dimension: '',
5922     possibleCause: '',
5923     guidelines: ''
5924   };
5925
5926   AppState.outputs = {
5927     logical: [],
5928     numeric: []
5929   };
5930 }
5931
5932 utils.js:
5933 /**
5934  * Вспомогательные функции
5935 */
```

```
5936
5937 /**
5938 * Генерация уникального ID
5939 */
5940 function generateId() {
5941     AppState.elementCounter++;
5942     return `elem_${AppState.elementCounter}`;
5943 }
5944
5945 function getInputPortType(elementId, portIdentifier) {
5946     const element = AppState.elements[elementId];
5947     if (!element) return SIGNAL_TYPE.ANY;
5948
5949     const config = ELEMENT_TYPES[element.type];
5950     if (!config) return SIGNAL_TYPE.ANY;
5951
5952     let portIndex = portIdentifier;
5953
5954     // Обработка технического порта условия
5955     if (typeof portIdentifier === 'string') {
5956         if (portIdentifier === 'cond-0' && config.hasConditionPort) {
5957             return config.conditionPortType || SIGNAL_TYPE.LOGIC;
5958         }
5959
5960         if (portIdentifier.startsWith('in-')) {
5961             portIndex = parseInt(portIdentifier.split('-')[1], 10);
5962         }
5963     }
5964
5965     if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5966         portIndex = 0;
5967     }
5968
5969     // Динамические входы для AND/OR берут тип из конфига
5970     if ((element.type === 'and' || element.type === 'or')) {
5971         return SIGNAL_TYPE.LOGIC; // Логические элементы всегда ожидают LOGIC на
5972         // входе
5973     }
5974
5975     if (element.type === 'formula') {
5976         return SIGNAL_TYPE.ANY;
5977     }
5978
5979     const types = config.inputTypes || [];
5980     if (types.length === 0) return SIGNAL_TYPE.ANY;
5981
5982     if (portIndex < types.length) {
5983         return types[portIndex] || SIGNAL_TYPE.ANY;
5984     }
5985
5986     return types[types.length - 1] || SIGNAL_TYPE.ANY;
5987 }
5988
5989 function getOutputPortType(elementId, portIdentifier) {
5990     const element = AppState.elements[elementId];
5991     if (!element) return SIGNAL_TYPE.ANY;
5992
5993     const config = ELEMENT_TYPES[element.type];
5994     if (!config) return SIGNAL_TYPE.ANY;
5995
5996     let portIndex = portIdentifier;
5997
5998     if (typeof portIdentifier === 'string') {
5999         if (portIdentifier.startsWith('out-')) {
6000             portIndex = parseInt(portIdentifier.split('-')[1], 10);
6001         }
6002     }
6003
6004     if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
6005         portIndex = 0;
6006     }
6007
6008     if (portIndex > config.outputTypes.length - 1) {
6009         return SIGNAL_TYPE.ANY;
6010     }
6011
6012     return config.outputTypes[portIndex] || SIGNAL_TYPE.ANY;
6013 }
```

```
6000         }
6001     }
6002
6003     if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
6004         portIndex = 0;
6005     }
6006
6007     const types = config.outputTypes || [];
6008     if (types.length === 0) return SIGNAL_TYPE.ANY;
6009
6010     if (portIndex < types.length) {
6011         return types[portIndex] || SIGNAL_TYPE.ANY;
6012     }
6013
6014     return types[types.length - 1] || SIGNAL_TYPE.ANY;
6015 }
6016 /**
6017 * Проверка совместимости типов сигналов
6018 *
6019 * Новая логика:
6020 * - ANY совместим со всем
6021 * - TRUE совместим с LOGIC, TRUE, ANY
6022 * - FALSE совместим с LOGIC, FALSE, ANY
6023 * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
6024 * - NUMERIC совместим с NUMERIC, ANY
6025 */
6026 function areTypesCompatible(outputType, inputType) {
6027     // Если один из типов ANY - совместимы
6028     if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
6029         return true;
6030     }
6031
6032     // Если типы одинаковые - совместимы
6033     if (outputType === inputType) {
6034         return true;
6035     }
6036
6037     // TRUE/FALSE совместимы с LOGIC
6038     if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
6039         inputType === SIGNAL_TYPE.LOGIC) {
6040         return true;
6041     }
6042
6043     // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
6044     if (outputType === SIGNAL_TYPE.LOGIC &&
6045         (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
6046         return true;
6047     }
6048
6049     return false;
6050 }
6051 /**
6052 * Проверка, находится ли элемент внутри рамки
6053 */
6054 function isInsideFrame(elemId, frameId) {
6055     const elem = AppState.elements[elemId];
6056     const frame = AppState.elements[frameId];
6057
6058     if (!elem || !frame || frame.type !== 'output-frame') return false;
6059
6060     const elemCenterX = elem.x + elem.width / 2;
6061     const elemCenterY = elem.y + elem.height / 2;
6062
6063     return elemCenterX > frame.x &&
```

```
6065         elemCenterX < frame.x + frame.width &&
6066         elemCenterY > frame.y &&
6067         elemCenterY < frame.y + frame.height;
6068     }
6069
6070     /**
6071      * Обновить принадлежность элементов к рамкам
6072     */
6073     function updateFrameChildren() {
6074         // Сначала очистим children у рамок и parentFrame у всех элементов
6075         Object.values(AppState.elements).forEach(elem => {
6076             if (elem.type === 'output-frame') {
6077                 elem.children = [];
6078             } else {
6079                 // удаляем parentFrame по умолчанию (пересчитаем ниже)
6080                 if (elem.parentFrame) delete elem.parentFrame;
6081             }
6082         });
6083
6084         // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
6085         // попадает
6086         Object.values(AppState.elements).forEach(elem => {
6087             if (!elem || elem.type === 'output-frame') return;
6088
6089             Object.values(AppState.elements).forEach(frame => {
6090                 if (!frame || frame.type !== 'output-frame') return;
6091
6092                 if (isInsideFrame(elem.id, frame.id)) {
6093                     // добавляем в массив детей рамки
6094                     frame.children.push(elem.id);
6095                     // отмечаем у элемента родительскую рамку
6096                     if (AppState.elements[frame.id]) {
6097                         AppState.elements[frame.id].parentFrame = frame.id;
6098                     }
6099                 });
6100             });
6101         }
6102
6103     /**
6104      * Преобразование координат экрана в координаты холста
6105     */
6106     function screenToCanvas(screenX, screenY) {
6107         const container = document.getElementById('workspace-container');
6108         const rect = container.getBoundingClientRect();
6109
6110         const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
6111         const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;
6112
6113         return { x, y };
6114     }
6115
6116     /**
6117      * Преобразование координат холста в координаты экрана
6118     */
6119     function canvasToScreen(canvasX, canvasY) {
6120         const container = document.getElementById('workspace-container');
6121         const rect = container.getBoundingClientRect();
6122
6123         const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
6124         const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;
6125
6126         return { x, y };
6127     }
6128 }
```

```
6129  /**
6130   * Проверка, является ли порт выходным (не подключен к другим элементам)
6131   */
6132  function isOutputPort(elemId, portIndex) {
6133    const portKey = `out-${portIndex}`;
6134
6135    // Проверяем, есть ли соединения от этого порта
6136    const hasConnection = AppState.connections.some(conn =>
6137      conn.fromElement === elemId && conn.fromPort === portKey
6138    );
6139
6140    return !hasConnection;
6141  }
6142
6143  /**
6144   * Получить информацию о выходном порте
6145   */
6146  function getOutputPortInfo(elemId, portIndex) {
6147    const elem = AppState.elements[elemId];
6148    if (!elem) return null;
6149
6150    const config = ELEMENT_TYPES[elem.type];
6151    if (!config) return null;
6152
6153    return {
6154      elementId: elemId,
6155      elementType: elem.type,
6156      elementName: config.name,
6157      portIndex: portIndex,
6158      portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
6159      portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
6160      // Дополнительная информация для идентификации
6161      displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$`}`.trim()
6162    };
6163  }
6164
6165  function splitArgsTopLevel(argStr) {
6166    const out = [];
6167    let cur = '';
6168    let depth = 0;
6169    for (let i = 0; i < argStr.length; i++) {
6170      const ch = argStr[i];
6171      if (ch === '(') depth++;
6172      if (ch === ')') depth--;
6173      if (ch === ',' && depth === 0) {
6174        out.push(cur.trim());
6175        cur = '';
6176      } else {
6177        cur += ch;
6178      }
6179    }
6180    if (cur.trim()) out.push(cur.trim());
6181    return out;
6182  }
6183
6184  function expandFormulaTemplates(expr, templatesMap) {
6185    if (!expr) return expr;
6186    if (!templatesMap) return expr;
6187
6188    // несколько проходов на случай вложенных шаблонов
6189    for (let pass = 0; pass < 10; pass++) {
6190      let changed = false;
6191
6192      expr = expr.replace(/([A-Za-z_]\w*)\s*\((([^()])|\\(([^()]*\\))*\\))/g, (match, name) =>
```

```
6193     {
6194         const tpl = templatesMap[name];
6195         if (!tpl) return match;
6196
6197         // вытащим аргументы вручную: name(....)
6198         const open = match.indexOf('(');
6199         const close = match.lastIndexOf(')');
6200         const inside = match.slice(open + 1, close);
6201
6202         const args = splitArgsTopLevel(inside);
6203         const formal = tpl.args || [];
6204         let body = String(tpl.body || '0');
6205
6206         // если количество аргументов не совпало – не трогаем (лучше так, чем сломать)
6207         if (args.length !== formal.length) return match;
6208
6209         formal.forEach((f, i) => {
6210             const re = new RegExp(`\\b${f}\\b`, 'g');
6211             body = body.replace(re, `(${args[i]})`);
6212         });
6213
6214         changed = true;
6215         return `(${body})`;
6216     );
6217
6218     if (!changed) break;
6219 }
6220
6221 return expr;
6222 }
6223
6224 viewport.js:
6225 /**
6226 * Модуль управления viewport (масштабирование и перемещение)
6227 */
6228 const Viewport = {
6229     /**
6230     * Инициализация viewport
6231     */
6232     init() {
6233         this.setupZoomControls();
6234         this.setupPanning();
6235         this.setupMouseWheel();
6236         this.setupMinimap();
6237         this.setupCursorPosition();
6238         this.updateTransform();
6239         const container = document.getElementById('workspace-container');
6240         const rect = container.getBoundingClientRect();
6241         AppState.viewport.panX = 100; // немного отступить от левого края
6242         AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
6243         AppState.viewport.zoom;
6244         this.updateTransform();
6245     },
6246
6247     /**
6248     * Настройка кнопок масштабирования
6249     */
6250     setupZoomControls() {
6251         document.getElementById('btn-zoom-in').addEventListener('click', () => {
6252             this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
6253         });
6254
6255         document.getElementById('btn-zoom-out').addEventListener('click', () => {
6256             this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
6257         });
6258     }
6259 }
```

```
6256
6257
6258     document.getElementById('btn-zoom-reset').addEventListener('click', () => {
6259         this.setZoom(1);
6260         this.setPan(0, 0);
6261     });
6262
6263     document.getElementById('btn-zoom-fit').addEventListener('click', () => {
6264         this.fitToContent();
6265     });
6266 },
6267
6268 /**
6269 * Настройка перемещения (пан)
6270 */
6271 setupPanning() {
6272     const container = document.getElementById('workspace-container');
6273
6274     container.addEventListener('mousedown', (e) => {
6275         // Средняя кнопка мыши или пробел + левая кнопка
6276         if (e.button === 1 || (e.button === 0 && e.target === container)) {
6277             e.preventDefault();
6278             AppState.viewport.isPanning = true;
6279             AppState.viewport.lastMouseX = e.clientX;
6280             AppState.viewport.lastMouseY = e.clientY;
6281             container.style.cursor = 'grabbing';
6282         }
6283     });
6284
6285     document.addEventListener('mousemove', (e) => {
6286         if (AppState.viewport.isPanning) {
6287             const dx = e.clientX - AppState.viewport.lastMouseX;
6288             const dy = e.clientY - AppState.viewport.lastMouseY;
6289
6290             this.setPan(
6291                 AppState.viewport.panX + dx,
6292                 AppState.viewport.panY + dy
6293             );
6294
6295             AppState.viewport.lastMouseX = e.clientX;
6296             AppState.viewport.lastMouseY = e.clientY;
6297         }
6298     });
6299
6300     document.addEventListener('mouseup', (e) => {
6301         if (AppState.viewport.isPanning) {
6302             AppState.viewport.isPanning = false;
6303             document.getElementById('workspace-container').style.cursor = '';
6304         }
6305     });
6306
6307 // Клавиша пробел для режима перемещения
6308 document.addEventListener('keydown', (e) => {
6309     if (e.code === 'Space' && !e.repeat) {
6310         document.getElementById('workspace-container').style.cursor = 'grab';
6311     }
6312 });
6313
6314 document.addEventListener('keyup', (e) => {
6315     if (e.code === 'Space') {
6316         document.getElementById('workspace-container').style.cursor = '';
6317     }
6318 });
6319 },
6320 }
```

```
6321     /**
6322      * Настройка масштабирования колесом мыши
6323      */
6324     setupMouseWheel() {
6325         const container = document.getElementById('workspace-container');
6326
6327         container.addEventListener('wheel', (e) => {
6328             e.preventDefault();
6329
6330             const rect = container.getBoundingClientRect();
6331             const mouseX = e.clientX - rect.left;
6332             const mouseY = e.clientY - rect.top;
6333
6334             // Позиция мыши на холсте до масштабирования
6335             const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
6336                 AppState.viewport.zoom;
6337             const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
6338                 AppState.viewport.zoom;
6339
6340             // Новый масштаб
6341             const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
6342                 VIEWPORT_CONFIG.zoomStep;
6343             const newZoom = Math.max(
6344                 VIEWPORT_CONFIG.minZoom,
6345                 Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
6346             );
6347
6348             // Корректируем pan, чтобы точка под курсором осталась на месте
6349             const newPanX = mouseX - canvasPosBeforeX * newZoom;
6350             const newPanY = mouseY - canvasPosBeforeY * newZoom;
6351
6352             AppState.viewport.zoom = newZoom;
6353             AppState.viewport.panX = newPanX;
6354             AppState.viewport.panY = newPanY;
6355
6356             this.updateTransform();
6357         }, { passive: false });
6358     },
6359
6360     /**
6361      * Установить масштаб
6362      */
6363     setZoom(zoom) {
6364         const container = document.getElementById('workspace-container');
6365         const rect = container.getBoundingClientRect();
6366
6367         // Центр экрана
6368         const centerX = rect.width / 2;
6369         const centerY = rect.height / 2;
6370
6371         // Позиция центра на холсте
6372         const canvasCenterX = (centerX - AppState.viewport.panX) /
6373             AppState.viewport.zoom;
6374         const canvasCenterY = (centerY - AppState.viewport.panY) /
6375             AppState.viewport.zoom;
6376
6377         // Новый масштаб
6378         const newZoom = Math.max(
6379             VIEWPORT_CONFIG.minZoom,
6380             Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
6381         );
6382
6383         // Корректируем pan
6384         AppState.viewport.panX = centerX - canvasCenterX * newZoom;
6385         AppState.viewport.panY = centerY - canvasCenterY * newZoom;
```

```
6381     AppState.viewport.zoom = newZoom;
6382
6383     this.updateTransform();
6384 },
6385
6386 /**
6387 * Установить смещение
6388 */
6389 setPan(x, y) {
6390     AppState.viewport.panX = x;
6391     AppState.viewport.panY = y;
6392     this.updateTransform();
6393 },
6394
6395 /**
6396 * Вписать содержимое в экран
6397 */
6398 fitToContent() {
6399     const elements = Object.values(AppState.elements);
6400     if (elements.length === 0) {
6401         this.setZoom(1);
6402         this.setPan(0, 0);
6403         return;
6404     }
6405
6406     // Находим границы содержимого
6407     let minX = Infinity, minY = Infinity;
6408     let maxX = -Infinity, maxY = -Infinity;
6409
6410     elements.forEach(elem => {
6411         minX = Math.min(minX, elem.x);
6412         minY = Math.min(minY, elem.y);
6413         maxX = Math.max(maxX, elem.x + elem.width);
6414         maxY = Math.max(maxY, elem.y + elem.height);
6415     });
6416
6417     const contentWidth = maxX - minX;
6418     const contentHeight = maxY - minY;
6419
6420     const container = document.getElementById('workspace-container');
6421     const rect = container.getBoundingClientRect();
6422
6423     const padding = 50;
6424     const availableWidth = rect.width - padding * 2;
6425     const availableHeight = rect.height - padding * 2;
6426
6427     const zoomX = availableWidth / contentWidth;
6428     const zoomY = availableHeight / contentHeight;
6429     const newZoom = Math.min(zoomX, zoomY, 1);
6430
6431     AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
6432     AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
6433 (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
6434     AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
6435 (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
6436
6437     this.updateTransform();
6438 },
6439
6440 /**
6441 * Обновить трансформацию
6442 */
6443 updateTransform() {
6444     const workspace = document.getElementById('workspace');
6445     const svg = document.getElementById('connections-svg');
```

```
6444
6445     const transform = `translate(${AppState.viewport.panX}px, ${
6446     AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
6446
6447     workspace.style.transform = transform;
6448     svg.style.transform = transform;
6449
6450     // Обновляем отображение масштаба
6451     document.getElementById('zoom-level').textContent = `${${
6452     Math.round(AppState.viewport.zoom * 100)}%}`;
6452
6453     // Обновляем мини-карту
6454     this.updateMinimap();
6455 },
6456
6457 /**
6458 * Настройка мини-карты
6459 */
6460 setupMinimap() {
6461     const minimap = document.getElementById('minimap');
6462     const canvas = document.getElementById('minimap-canvas');
6463
6464     canvas.width = MINIMAP_CONFIG.width;
6465     canvas.height = MINIMAP_CONFIG.height;
6466
6467     // Клик по мини-карте для перемещения
6468     minimap.addEventListener('click', (e) => {
6469         const rect = minimap.getBoundingClientRect();
6470         const x = e.clientX - rect.left;
6471         const y = e.clientY - rect.top;
6472
6473         this.navigateToMinimapPosition(x, y);
6474     });
6475 },
6476
6477 /**
6478 * Обновить мини-карту
6479 */
6480 updateMinimap() {
6481     const canvas = document.getElementById('minimap-canvas');
6482     const ctx = canvas.getContext('2d');
6483     const viewportEl = document.getElementById('minimap-viewport');
6484
6485     // Очищаем
6486     ctx.fillStyle = '#0a0ala';
6487     ctx.fillRect(0, 0, canvas.width, canvas.height);
6488
6489     // Масштаб мини-карты
6490     const scale = Math.min(
6491         canvas.width / VIEWPORT_CONFIG.canvasWidth,
6492         canvas.height / VIEWPORT_CONFIG.canvasHeight
6493     );
6494
6495     // Рисуем элементы
6496     Object.values(AppState.elements).forEach(elem => {
6497         const x = elem.x * scale;
6498         const y = elem.y * scale;
6499         const w = Math.max(elem.width * scale, 2);
6500         const h = Math.max(elem.height * scale, 2);
6501
6502         ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
6503         ctx.fillRect(x, y, w, h);
6504     });
6505
6506     // Рисуем viewport
```

```
6507     const container = document.getElementById('workspace-container');
6508     const rect = container.getBoundingClientRect();
6509
6510     const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
6511     const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
6512     const vpW = (rect.width / AppState.viewport.zoom) * scale;
6513     const vpH = (rect.height / AppState.viewport.zoom) * scale;
6514
6515     viewportEl.style.left = `${vpX}px`;
6516     viewportEl.style.top = `${vpY}px`;
6517     viewportEl.style.width = `${vpW}px`;
6518     viewportEl.style.height = `${vpH}px`;
6519 },
6520
6521 /**
6522 * Перейти к позиции на мини-карте
6523 */
6524 navigateToMinimapPosition(minimapX, minimapY) {
6525     const canvas = document.getElementById('minimap-canvas');
6526     const container = document.getElementById('workspace-container');
6527     const rect = container.getBoundingClientRect();
6528
6529     const scale = Math.min(
6530         canvas.width / VIEWPORT_CONFIG.canvasWidth,
6531         canvas.height / VIEWPORT_CONFIG.canvasHeight
6532     );
6533
6534     const canvasX = minimapX / scale;
6535     const canvasY = minimapY / scale;
6536
6537     // Центрируем viewport на этой точке
6538     AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
6539     AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
6540
6541     this.updateTransform();
6542 },
6543
6544 /**
6545 * Отслеживание позиции курсора
6546 */
6547 setupCursorPosition() {
6548     const container = document.getElementById('workspace-container');
6549
6550     container.addEventListener('mousemove', (e) => {
6551         const pos = screenToCanvas(e.clientX, e.clientY);
6552         document.getElementById('cursor-pos').textContent =
6553             `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)} `;
6554     });
6555 }
6556 };
6557
6558 styles.css:
6559 *
6560 * {
6561     margin: 0;
6562     padding: 0;
6563     box-sizing: border-box;
6564 }
6565
6566 body {
6567     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
6568     background: #lala2e;
6569     color: #eee;
6570     overflow: hidden;
6571 }
```

```
6572
6573 #app {
6574     display: flex;
6575     flex-direction: column;
6576     height: 100vh;
6577 }
6578
6579 /* ===== МЕНЮ ===== */
6580 #menu {
6581     background: #16213e;
6582     padding: 10px 20px;
6583     display: flex;
6584     gap: 10px;
6585     align-items: center;
6586     border-bottom: 2px solid #0f3460;
6587     z-index: 100;
6588     flex-wrap: wrap;
6589 }
6590
6591 .menu-btn {
6592     background: #0f3460;
6593     color: #eee;
6594     border: none;
6595     padding: 8px 16px;
6596     border-radius: 5px;
6597     cursor: pointer;
6598     transition: background 0.3s;
6599     font-size: 13px;
6600 }
6601
6602 .menu-btn:hover {
6603     background: #e94560;
6604 }
6605
6606 .menu-separator {
6607     width: 1px;
6608     height: 30px;
6609     background: #0f3460;
6610     margin: 0 10px;
6611 }
6612
6613 .zoom-controls {
6614     display: flex;
6615     align-items: center;
6616     gap: 8px;
6617     background: #0a0a1a;
6618     padding: 5px 10px;
6619     border-radius: 5px;
6620 }
6621
6622 .zoom-btn {
6623     width: 30px;
6624     height: 30px;
6625     padding: 0;
6626     font-size: 18px;
6627     font-weight: bold;
6628 }
6629
6630 #zoom-level {
6631     min-width: 50px;
6632     text-align: center;
6633     font-size: 12px;
6634     color: #aaa;
6635 }
6636
```

```
6637 /* ===== ОСНОВНАЯ ОБЛАСТЬ ===== */
6638 #main {
6639     display: flex;
6640     flex: 1;
6641     overflow: hidden;
6642 }
6643
6644 /* ===== ПАЛИТРА ===== */
6645 #palette {
6646     width: 200px;
6647     background: #16213e;
6648     padding: 15px;
6649     border-right: 2px solid #0f3460;
6650     overflow-y: auto;
6651     z-index: 10;
6652     flex-shrink: 0;
6653 }
6654
6655 #palette h3 {
6656     margin-bottom: 15px;
6657     color: #e94560;
6658     text-align: center;
6659     font-size: 14px;
6660 }
6661
6662 .palette-section {
6663     margin-bottom: 15px;
6664 }
6665
6666 .palette-section-title {
6667     font-size: 11px;
6668     color: #888;
6669     margin-bottom: 8px;
6670     padding-bottom: 3px;
6671     border-bottom: 1px solid #333;
6672 }
6673
6674 .palette-item {
6675     background: #0f3460;
6676     padding: 8px;
6677     margin-bottom: 6px;
6678     border-radius: 8px;
6679     cursor: grab;
6680     text-align: center;
6681     transition: all 0.3s;
6682     border: 2px solid transparent;
6683     user-select: none;
6684 }
6685
6686 .palette-item:hover {
6687     border-color: #e94560;
6688     transform: scale(1.02);
6689 }
6690
6691 .palette-item:active {
6692     cursor: grabbing;
6693 }
6694
6695 .palette-item svg {
6696     width: 50px;
6697     height: 32px;
6698     margin-bottom: 2px;
6699     pointer-events: none;
6700 }
6701
```

```
6702 .palette-item-name {  
6703     font-size: 10px;  
6704     color: #aaa;  
6705     pointer-events: none;  
6706 }  
6707  
6708 .type-legend {  
6709     margin-top: 15px;  
6710     padding-top: 10px;  
6711     border-top: 1px solid #333;  
6712     font-size: 10px;  
6713 }  
6714  
6715 .type-legend-item {  
6716     display: flex;  
6717     align-items: center;  
6718     gap: 8px;  
6719     margin-bottom: 5px;  
6720 }  
6721  
6722 .type-legend-dot {  
6723     width: 12px;  
6724     height: 12px;  
6725     border-radius: 50%;  
6726     border: 2px solid #fff;  
6727 }  
6728 .type-legend-dot.logic { background: #a855f7; }  
6729 .type-legend-dot.number { background: #3b82f6; }  
6730  
6731 /* ===== РАБОЧАЯ ОБЛАСТЬ ===== */  
6732 #workspace-container {  
6733     flex: 1;  
6734     position: relative;  
6735     overflow: hidden;  
6736     background-color: #0a0a1a;  
6737     background-image:  
6738         linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),  
6739         linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);  
6740     background-size: 25px 25px;  
6741 }  
6742  
6743 #workspace {  
6744     position: absolute;  
6745     transform-origin: 0 0;  
6746     width: 5000px;  
6747     height: 5000px;  
6748 }  
6749  
6750 #connections-svg {  
6751     position: absolute;  
6752     transform-origin: 0 0;  
6753     pointer-events: none;  
6754     z-index: 5;  
6755     width: 5000px;  
6756     height: 5000px;  
6757 }  
6758  
6759 #connections-svg path {  
6760     pointer-events: stroke;  
6761 }  
6762  
6763 /* ===== ЭЛЕМЕНТЫ ===== */  
6764 .element {  
6765     position: absolute;  
6766     background: #0f3460;
```

```
6767     border: 2px solid #4a90d9;
6768     border-radius: 8px;
6769     cursor: move;
6770     user-select: none;
6771     z-index: 10;
6772     display: flex;
6773     flex-direction: column;
6774 }
6775
6776 .element.selected {
6777     border-color: #e94560;
6778     box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
6779 }
6780
6781 .element-header {
6782     background: #4a90d9;
6783     padding: 5px 10px;
6784     border-radius: 5px 5px 0 0;
6785     font-size: 11px;
6786     font-weight: bold;
6787     text-align: center;
6788     white-space: nowrap;
6789     overflow: hidden;
6790     text-overflow: ellipsis;
6791 }
6792
6793 .element-body {
6794     padding: 10px;
6795     display: flex;
6796     justify-content: space-between;
6797     align-items: center;
6798     flex: 1;
6799     gap: 8px;
6800 }
6801
6802 .element-symbol {
6803     font-size: 16px;
6804     font-weight: bold;
6805     flex: 1;
6806     text-align: center;
6807     padding: 0 5px;
6808     word-break: break-all;
6809     color: #eee;
6810 }
6811
6812 /* ===== ПОРТЫ ===== */
6813 .ports-left, .ports-right {
6814     display: flex;
6815     flex-direction: column;
6816     justify-content: space-around;
6817     gap: 10px;
6818     height: 100%;
6819 }
6820
6821 .port {
6822     width: 14px;
6823     height: 14px;
6824     border-radius: 50%;
6825     border: 2px solid #fff;
6826     cursor: crosshair;
6827     transition: all 0.2s;
6828     position: relative;
6829     flex-shrink: 0;
6830 }
6831
```

```
6832 .port:hover { transform: scale(1.3); }
6833 .port.input { margin-left: -8px; }
6834 .port.output { margin-right: -8px; }
6835 .port.connected { background: #4ade80; }
6836
6837 /* Типы портов */
6838 .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
6839 .port.logic-port:hover { background: #c084fc; }
6840 .port.logic-port.connected { background: #7c3aed; }
6841
6842 .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
6843 .port.number-port:hover { background: #60a5fa; }
6844 .port.number-port.connected { background: #2563eb; }
6845
6846 .port.any-port { background: #6b7280; border-color: #d1d5db; }
6847 .port.any-port:hover { background: #9ca3af; }
6848 .port.any-port.connected { background: #4b5563; }
6849
6850 .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !important; }
6851 .port.output.no-port { background: #f87171 !important; border-color: #fecaca !important; }
6852
6853 .port.incompatible { opacity: 0.3; cursor: not-allowed; }
6854 .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
6855
6856 /* ===== RESIZE HANDLES ===== */
6857 .resize-handle {
6858     position: absolute;
6859     width: 12px;
6860     height: 12px;
6861     background: #e94560;
6862     border: 1px solid #fff;
6863     border-radius: 3px;
6864     z-index: 20;
6865     opacity: 0;
6866     transition: opacity 0.2s;
6867 }
6868 .element.selected .resize-handle { opacity: 0.8; }
6869 .resize-handle:hover { opacity: 1; }
6870 .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
6871 .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor: ew-resize; }
6872 .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%); cursor: ns-resize; }
6873
6874
6875 /* ===== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ===== */
6876 .element.input-signal {
6877     background: transparent;
6878     border: none;
6879 }
6880
6881 .element.input-signal .element-header {
6882     display: none; /* У трапеции нет заголовка */
6883 }
6884
6885 .element.input-signal .element-body {
6886     padding: 0;
6887     background: #0f3460;
6888     border: 2px solid #4a90d9;
6889     clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
6890     display: flex;
6891     justify-content: space-between;
6892     align-items: center;
```

```
6893     padding-left: 15px;
6894     padding-right: 25px;
6895 }
6896
6897 .element.input-signal .element-symbol {
6898     text-align: left;
6899     color: #eee;
6900 }
6901
6902 .element.input-signal.selected .element-body {
6903     border-color: #e94560;
6904 }
6905
6906 /* ===== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ===== */
6907 .element.output {
6908     background: rgba(16, 185, 129, 0.1);
6909     border: 2px dashed #10b981;
6910 }
6911
6912 .element.output .element-header {
6913     display: none; /* У выхода нет заголовка */
6914 }
6915
6916 .element.output .element-body {
6917     padding-left: 20px;
6918 }
6919
6920 .element.output .element-symbol {
6921     color: #10b981;
6922     font-size: 14px;
6923 }
6924
6925 .element.output.selected {
6926     border-color: #e94560;
6927     border-style: dashed;
6928 }
6929
6930
6931 /* Formula condition port */
6932 /* Универсальный стиль для технического порта (сверху) */
6933 .element.has-condition-port {
6934     margin-top: 30px; /* Даем место порту над элементом */
6935 }
6936
6937 .condition-port-wrapper {
6938     position: absolute;
6939     top: -28px;
6940     left: 50%;
6941     transform: translateX(-50%);
6942     display: flex;
6943     flex-direction: column;
6944     align-items: center;
6945     gap: 4px;
6946     pointer-events: none;
6947     z-index: 21;
6948 }
6949
6950 .condition-port-label {
6951     font-size: 10px;
6952     color: #f59e0b;
6953     font-weight: 600;
6954     white-space: nowrap;
6955 }
6956
6957 .port.condition-port {
```

```
6958     pointer-events: auto;
6959     width: 16px;
6960     height: 16px;
6961     border-radius: 50%;
6962     border: 2px solid #f59e0b;
6963     background: #fff7ed;
6964     margin: 0; /* Сбрасываем лишние отступы */
6965 }
6966 .element.formula .condition-port:hover { background: #fde68a; }
6967
6968
6969 /* ===== СОЕДИНЕНИЯ ===== */
6970 .connection {
6971     fill: none !important; /* ← добавляем !important */
6972     stroke: #4a90d9;
6973     stroke-width: 2.5;
6974 }
6975 .connection:hover {
6976     stroke: #e94560;
6977     stroke-width: 4;
6978 }
6979
6980 .connection.logic-conn { stroke: #a855f7; }
6981 .connection.numeric-conn { stroke: #3b82f6; }
6982 .connection.any-conn { stroke: #6b7280; }
6983 .connection.true-conn { stroke: #4ade80; }
6984 .connection.false-conn { stroke: #f87171; }
6985
6986 .connection.yes-conn { stroke: #4ade80; }
6987 .connection.no-conn { stroke: #f87171; }
6988
6989 .temp-connection {
6990     fill: none !important; /* ← добавляем !important */
6991     stroke: #e94560;
6992     stroke-width: 2;
6993     stroke-dasharray: 5, 5;
6994 }
6995 .temp-connection.invalid { stroke: #ef4444; }
6996
6997 /* ===== ПРОЧЕЕ ===== */
6998 .drag-preview {
6999     position: fixed;
7000     pointer-events: none;
7001     opacity: 0.8;
7002     z-index: 1000;
7003     background: #0f3460;
7004     border: 2px solid #e94560;
7005     border-radius: 8px;
7006     padding: 10px 15px;
7007     color: #fff;
7008     font-size: 12px;
7009 }
7010
7011 #minimap {
7012     position: absolute;
7013     bottom: 20px;
7014     right: 20px;
7015     width: 200px;
7016     height: 150px;
7017     background: #16213e;
7018     border: 2px solid #0f3460;
7019     border-radius: 8px;
7020     overflow: hidden;
7021     z-index: 50;
7022 }
```

```
7023  
7024 #minimap-canvas { width: 100%; height: 100%; }  
7025 #minimap-viewport {  
7026     position: absolute;  
7027     border: 2px solid #e94560;  
7028     background: rgba(233, 69, 96, 0.2);  
7029     pointer-events: none;  
7030 }  
7031  
7032 #viewport-info {  
7033     position: absolute;  
7034     bottom: 20px;  
7035     left: 20px;  
7036     background: rgba(22, 33, 62, 0.9);  
7037     padding: 8px 12px;  
7038     border-radius: 5px;  
7039     font-size: 11px;  
7040     color: #888;  
7041     z-index: 50;  
7042     display: flex;  
7043     gap: 15px;  
7044 }  
7045 #selection-info { color: #e94560; }  
7046  
7047 #modal-overlay, .modal-overlay-class {  
7048     display: none;  
7049     position: fixed;  
7050     top: 0; left: 0;  
7051     width: 100%; height: 100%;  
7052     background: rgba(0, 0, 0, 0.7);  
7053     z-index: 1000;  
7054     justify-content: center;  
7055     align-items: center;  
7056 }  
7057  
7058 #modal, .modal-class {  
7059     background: #16213e;  
7060     border-radius: 10px;  
7061     padding: 20px;  
7062     min-width: 400px;  
7063     max-width: 600px;  
7064     max-height: 80vh;  
7065     overflow-y: auto;  
7066     border: 2px solid #0f3460;  
7067 }  
7068  
7069 #modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }  
7070 .modal-row { margin-bottom: 15px; }  
7071 .modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }  
7072 .modal-row input, .modal-row select, .modal-row textarea {  
7073     width: 100%;  
7074     padding: 10px;  
7075     background: #0f3460;  
7076     border: 1px solid #4a90d9;  
7077     border-radius: 5px;  
7078     color: #eee;  
7079     font-size: 14px;  
7080 }  
7081 .modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline:  
    none; border-color: #e94560; }  
7082 .modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }  
7083 .signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-  
    radius: 5px; padding: 5px; margin-top: 5px; }  
7084 .signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size:  
    12px; }
```

```
7085 .signal-item:hover { background: #4a90d9; }
7086 .modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
7087 .modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer; font-size: 14px; transition: background 0.3s; }
7088 .modal-btn.save { background: #4ade80; color: #000; }
7089 .modal-btn.save:hover { background: #22c55e; }
7090 .modal-btn.cancel { background: #6b7280; color: #fff; }
7091 .modal-btn.cancel:hover { background: #4b5563; }
7092
7093 #context-menu {
7094     display: none;
7095     position: fixed;
7096     background: #16213e;
7097     border: 1px solid #0f3460;
7098     border-radius: 5px;
7099     padding: 5px 0;
7100     z-index: 1001;
7101     min-width: 150px;
7102     box-shadow: 0 5px 20px rgba(0,0,0,0.3);
7103 }
7104 .context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition: background 0.2s; }
7105 .context-item:hover { background: #0f3460; }
7106
7107 #file-input { display: none; }
7108
7109 .project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
7110 .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; }
7111 .project-type-btn:hover { border-color: #e94560; }
7112 .project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
7113 .project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
7114 .project-type-btn .type-name { font-weight: bold; }
7115 .project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }
7116
7117 .conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-radius: 8px; margin-top: 10px; }
7118 .conditional-fields.visible { display: block; }
7119
7120 ::-webkit-scrollbar { width: 8px; height: 8px; }
7121 ::-webkit-scrollbar-track { background: #0a0a1a; }
7122 ::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
7123 ::-webkit-scrollbar-thumb:hover { background: #e94560; }
7124
7125 /* Стили для выходов */
7126 .output-btn { position: relative; }
7127 .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; }
7128 .output-counter:empty, .output-counter[style*="display: none"] { display: none; }
7129 .element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
7130 .element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important; border-color: #fbbf24 !important; }
7131 .port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation: pulse-output 1.5s infinite; }
7132 @keyframes pulse-output {
7133     0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
7134     50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
7135 }
7136
7137 .outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-height: 250px; overflow-y: auto; }
7138 .outputs-section { margin-bottom: 15px; }
```

```
7139 .outputs-section:last-child { margin-bottom: 0; }
7140 .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; }
7141 .outputs-section-title .section-icon { font-size: 16px; }
7142 .outputs-list { display: flex; flex-direction: column; gap: 5px; }
7143 .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; }
7144 .output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981;
    transform: translateX(5px); }
7145 .output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59,
    130, 246, 0.3); }
7146 .output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color:
    #3b82f6; }
7147 .output-icon { font-size: 14px; }
7148 .output-name { font-weight: bold; color: #eee; }
7149 .output-port { color: #888; font-size: 12px; margin-left: auto; }
7150 .no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
7151 .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; }
7152 .element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-
    color: rgba(240, 80, 80, 0.8) !important; }
7153 .element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-
    color: rgba(250, 200, 30, 0.8) !important; }
7154 /* TRUE/FALSE порты (для сепаратора) */
7155 .port.true-port {
    background: #4ade80 !important;
    border-color: #bbf7d0 !important;
}
7156 .port.true-port:hover {
    background: #22c55e !important;
}
7157 .port.true-port.connected {
    background: #16a34a !important;
}
7158 .
7159 .port.false-port {
    background: #f87171 !important;
    border-color: #fecaca !important;
}
7160 .port.false-port:hover {
    background: #ef4444 !important;
}
7161 .port.false-port.connected {
    background: #dc2626 !important;
}
7162 .
7163 .
7164 .
7165 .
7166 .
7167 .
7168 .
7169 .
7170 .
7171 .
7172 .
7173 .
7174 .
7175 .
7176 .
7177 /* Сепаратор стиль */
7178 .element.separator {
    background: #0f3460;
    border: 2px solid #f59e0b;
}
7179 .
7180 .
7181 .
7182 .
7183 .
7184 .
7185 .
7186 .
7187 .
7188 /* === Выделение рамкой === */
7189 #selection-rect {
    position: absolute;
    border: 1px dashed #e94560;
    background: rgba(233, 69, 96, 0.1);
    pointer-events: none;
}
```

```
7194     display: none;
7195     z-index: 200;
7196 }
7197
7198 /* === Кастомный элемент “Группа” === */
7199 .element.group {
7200     background: rgba(107, 114, 128, 0.12);
7201     border: 2px dashed #6b7280;
7202     border-radius: 8px;
7203     position: absolute;
7204     z-index: 1;           /* ниже обычных элементов (у них z-index: 10) */
7205 }
7206
7207 .element.group .group-title {
7208     pointer-events: auto;
7209 }
7210
7211 .group-title {
7212     position: absolute;
7213     top: -20px;
7214     left: 5px;
7215     font-size: 11px;
7216     color: #ccc;
7217     background: #16213e;
7218     padding: 2px 6px;
7219     border-radius: 4px;
7220     pointer-events: auto; /* можно кликнуть для выбора */
7221 }
7222
7223 .modal.hidden { display: none; }
7224 .modal { position: fixed; inset: 0; display: flex; align-items: center; justify-content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
7225 .modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px; max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow: hidden; }
7226 .modal__content--wide { width: 800px; }
7227 .modal__title { margin: 0; }
7228
7229 .project-list__toolbar { display: flex; gap: 12px; }
7230 .project-list__toolbar input { flex: 1; padding: 6px 10px; }
7231 .project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd; border-radius: 6px; }
7232 .project-list__table { width: 100%; border-collapse: collapse; }
7233 .project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom: 1px solid #eee; }
7234 .project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
7235 .project-list__table tbody tr:hover { background: #f0f6ff; }
7236 .project-list__empty { text-align: center; color: #888; padding: 16px; }
7237 .modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
7238 .project-list__table th,
7239 .project-list__table td {
7240     color: #111;           /* насыщенный чёрный текст */
7241     padding: 8px 12px;
7242     border-bottom: 1px solid #eee;
7243 }
7244 .modal__content--wide {
7245     width: 860px;
7246     max-height: 90vh;      /* занимает 90% экрана */
7247 }
7248
7249 .project-list__table-container {
7250     flex: 1;
7251     overflow: auto;
7252     border: 1px solid #ddd;
7253     border-radius: 6px;
```

```
7254     max-height: 60vh;      /* много строк */
7255 }
7256
7257 .element-comment {
7258     padding: 6px 10px 10px;
7259     font-size: 11px;
7260     color: #cbd5e1;
7261     opacity: 0.9;
7262     border-top: 1px solid rgba(255,255,255,0.08);
7263     white-space: pre-wrap;
7264     word-break: break-word;
7265 }
7266
7267 .element-comment:empty { display: none; }
7268
```