

Содержание

1 Компоненты 3

1.1 cmp_add_input_stream 4

1.2 cmp_add_output_stream 5

1.3 cmp_auth 6

1.3.1 Платформы 6

1.3.2 Конфигурация 6

1.3.2.1 Config 6

1.4 cmp_derive 7

1.5 cmp_esp_adc 8

1.6 cmp_esp_gpio 9

1.6.1 Конфигурация 9

1.6.1.1 Config 9

1.7 cmp_esp_mqtt_client 10

1.7.1 Платформы 10

1.8 cmp_esp_nvs 11

1.9 cmp_esp_wifi 12

1.10 cmp_external_fn_process 13

1.11 cmp_http_client 14

1.12 cmp_http_client_wasm 15

1.13 cmp_http_server 16

1.14 cmp_http_server_esp 17

1.15 cmp_influxdb 18

1.16 cmp_inject_periodic 19

1.17 cmp_leptos 20

1.17.1 Платформы 20

1.17.2 Конфигурация 20

1.17.2.1 Config 20

1.17.3 Создание проекта 20

1.17.3.1 .vscode/settings.json 20

1.17.3.2 .zed/settings.json 20

1.17.3.3 Tauri 20

1.17.3.4 Tailwind 20

1.17.3.5 Material Theme 21

1.17.3.6 Iconify 22

1.18 cmp_plc 23

1.19 cmp_telegram 24

1.19.1 Платформы 24

1.19.2 Конфигурация 24

1.19.2.1 Config 24

2 Исполнитель 25

2.1 WASM (Leptos) 26

3 Внешние сервисы 27

3.1 EMQX 28

3.1.1 docker 28

3.2 Go2rtc 29

3.2.1 docker 29

3.2.2 ./config_services/go2rtc/go2rtc.yaml 29

3.3 Grafana 30

3.3.1 docker 30

3.3.2 Файлы конфигурации 30

3.3.2.1 ./config_services/grafana/datasources/ 30

3.3.2.2 ./config_services/grafana/dashboards/ 31

3.4 InfluxDB (v2) 32

3.4.1 docker 32

3.5 InfluxDB (v3) 33

3.6 Loki 34

3.6.1 docker 34

3.7 Portainer 35

3.7.1 docker 35

3.8 Redis 36

3.8.1 docker 36

3.8.2 Файлы конфигурации 36

3.8.2.1 redis.conf 36

3.9 Rust 37

3.9.1 docker (бекенд) 37

3.9.2 docker (cmp_leptos)	37
3.10 Sentryshot	38
3.10.1 docker	38
3.10.2 Файлы конфигурации	38
3.10.2.1 ./sentryshot/configs/sentryshot.toml	38
3.10.2.2 ./sentryshot/configs/monitors/	39
3.11 SurrealDB	40
3.11.1 docker	40
3.12 SystemD	41
3.13 TimescaleDB	42
3.13.1 docker	42
3.13.2 postgresql.conf	42
3.13.3 pg_hba.conf	42
3.13.4 init.sql	42

1 Компоненты

Клиентское подключение:

- [cmp_http_client_wasm](./cmp_http_client_wasm.md)
- [cmp_http_client](./cmp_http_client.md)
- [cmp_modbus_client](./cmp_modbus_client.md)
- [cmp_websocket_client_wasm](./cmp_websocket_client_wasm.md)
- [cmp_websocket_client](./cmp_websocket_client.md)

Сервера:

- [cmp_http_server_esp](./cmp_http_server_esp.md)
- [cmp_http_server](./cmp_http_server.md)
- [cmp_websocket_server](./cmp_websocket_server.md)

Брокеры сообщений:

- [cmp_esp_mqtt_client](./cmp_esp_mqtt_client.md)
- [cmp_mqtt_client](./cmp_mqtt_client.md)
- [cmp_redis_client](./cmp_redis_client.md)

Интерфейс пользователя:

- [cmp_leptos](./cmp_leptos.md)
- [cmp_slint](./cmp_slint.md)

Авторизация:

- [cmp_auth](./cmp_auth.md)

Сохранение данных:

- [cmp_esp_nvs](./cmp_esp_nvs.md)
- [cmp_influxdb](./cmp_influxdb.md)
- [cmp_surrealdb](./cmp_surrealdb.md)
- [cmp_timescaledb](./cmp_timescaledb.md)
- [cmp_webstorage](./cmp_webstorage.md)

Взаимодействие с аппаратной частью

- [cmp_esp_adc](./cmp_esp_adc.md)
- [cmp_esp_gpio](./cmp_esp_gpio.md)
- [cmp_esp_wifi](./cmp_esp_wifi.md)
- [cmp_raspberrypi_gpio](./cmp_raspberrypi_gpio.md)

Логика исполнения

- [cmp_plc](./cmp_plc.md)

Систеная информация

- [cmp_system_info](./cmp_system_info.md)

Служебные компоненты:

- [cmp_add_input_stream](./cmp_add_input_stream.md)
- [cmp_add_output_stream](./cmp_add_output_stream.md)
- [cmp_derive](./cmp_derive.md)
- [cmp_external_fn_process](./cmp_external_fn_process.md)
- [cmp_inject_periodic](./cmp_inject_periodic.md)
- [cmp_logger](./cmp_logger.md)

1.1 cmp_add_input_stream

1.2 cmp_add_output_stream

1.3 cmp_auth

Компонент авторизации пользователей

1.3.1 Платформы

target triple	Поддержка
aarch64-linux-android	-
aarch64-unknown-linux-gnu	+
riscv32imc-esp-espidf	-
x86_64-linux-android	-
x86_64-unknown-linux-gnu	+
wasm32-unknown-unknown	-

1.3.2 Конфигурация

1.3.2.1 Config

secret_key	String	Секретный ключ для валидации токенов
store		Хранилище данных доступа
Локальное сохранение: store: cmp_auth::ConfigStore::Local(vec![cmp_auth::ConfigStoreLocalItem { login: "admin".into(), password: "admin".into(), role: AuthPermissions::Admin, }],		

1.4 cmp_derive

1.5 cmp_esp_adc

1.6 cmp_esp_gpio

Компонент для работы с входами и выходами GPIO микроконтроллера ESP

target triple	Поддержка
aarch64-linux-android	-
aarch64-unknown-linux-gnu	-
riscv32imc-esp-espidf	+
x86_64-linux-android	-
x86_64-unknown-linux-gnu	-
wasm32-unknown-unknown	-

1.6.1 Конфигурация

1.6.1.1 Config

inputs	Конфигурация входов
<pre>inputs: vec![cmp_esp_gpio::ConfigGpioInput { peripherals: peripherals.pins.gpio9.into(), fn_output: value Message::new_custom(Custom::EspBootButton(value)), }],</pre>	
outputs	Конфигурация выходов
<pre>outputs: vec![cmp_esp_gpio::ConfigGpioOutput { peripherals: peripherals.pins.gpio1.into(), fn_input: msg match msg.data { MsgData::Custom(Custom::EspRelay(value)) => Some(value), _ => None, }, is_low_triggered: false, }],</pre>	

1.7 cmp_esp_mqtt_client

Клиент MQTT микроконтроллера ESP32

1.7.1 Платформы

target triple	Поддержка
aarch64-linux-android	-
aarch64-unknown-linux-gnu	-
riscv32imc-esp-espidf	+
x86_64-linux-android	-
x86_64-unknown-linux-gnu	-
wasm32-unknown-unknown	-

1.8 cmp_esp_nvs

1.9 cmp_esp_wifi

1.10 cmp_external_fn_process

1.11 cmp_http_client

1.12 cmp_http_client_wasm

1.13 cmp_http_server

1.14 cmp_http_server_esp

1.15 cmp_influxdb

1.16 cmp_inject_periodic

1.17 cmp_leptos

Компонент для интеграции веб-приложения на основе фреймворка [Leptos](https://leptos.dev).

1.17.1 Платформы

target triple	Поддержка
aarch64-linux-android	-
aarch64-unknown-linux-gnu	-
riscv32imc-esp-espidf	-
x86_64-linux-android	-
x86_64-unknown-linux-gnu	-
wasm32-unknown-unknown	+

1.17.2 Конфигурация

1.17.2.1 Config

body_component	Корневой компонент для монтирования
body_component:	view! { <App/> }
hostname	Имя хоста, на котором развернуто веб-приложение

1.17.3 Создание проекта

1.17.3.1 .vscode/settings.json

```
{
  "rust-analyzer.cargo.target": "wasm32-unknown-unknown"
}
```

1.17.3.2 .zed/settings.json

```
{
  "lsp": {
    "rust-analyzer": {
      "initialization_options": {
        "check": {
          "command": "clippy"
        },
        "cargo": {
          "target": "wasm32-unknown-unknown"
        }
      }
    }
  }
}
```

1.17.3.3 Tauri

```
cargo create-tauri-app --rc
```

```
# добавить поддержку Android
cargo tauri android init
```

1.17.3.4 Tailwind

Установить:

```
npm install -D tailwindcss
npm install -D @tailwindcss/forms
npx tailwindcss init
```

Создать файл tailwind.config.js:

```
/** @type {import('tailwindcss').Config} */

module.exports = {
  content: {
    files: [
      "*.html",
      "./src/**/*.rs",
      "../../rsiot/src/components/cmp_leptos/components/**/*.rs"
    ],
  },
  plugins: [require('@tailwindcss/forms')],
}
```

Создать файл input.css в корне проекта:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Добавить в index.html:

```
<html>
  <head>
    <!-- Подключаем стили Tailwind -->
    <link data-trunk rel="tailwind-css" href="input.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  </head>
</html>
```

1.17.3.5 Material Theme

Создаем тему в [Material Theme Builder](#). Скачиваем набор файлов css, распаковываем в папку material-theme. В начале файла input.css прописываем:

```
/* Material theme */
@import "../material-theme/dark.css";
@import "../material-theme/dark-hc.css";
@import "../material-theme/dark-mc.css";
@import "../material-theme/light.css";
@import "../material-theme/light-hc.css";
@import "../material-theme/light-mc.css";
```

Прописать секцию theme в tailwind.config.json:

```
module.exports = {
  content: {
    files: [...],
  },
  plugins: [...],
  theme: {
    extend: {
      colors: {
        primary: "var(--md-sys-color-primary)",
        "surface-tint": "var(--md-sys-color-surface-tint)",
        "on-primary": "var(--md-sys-color-on-primary)",
        "primary-container": "var(--md-sys-color-primary-container)",
        "on-primary-container": "var(--md-sys-color-on-primary-container)",
        secondary: "var(--md-sys-color-secondary)",
        "on-secondary": "var(--md-sys-color-on-secondary)",
        "secondary-container": "var(--md-sys-color-secondary-container)",
        "on-secondary-container": "var(--md-sys-color-on-secondary-container)",
        tertiary: "var(--md-sys-color-tertiary)",
        "on-tertiary": "var(--md-sys-color-on-tertiary)",
        "tertiary-container": "var(--md-sys-color-tertiary-container)",
        "on-tertiary-container": "var(--md-sys-color-on-tertiary-container)",
        error: "var(--md-sys-color-error)",
        "on-error": "var(--md-sys-color-on-error)",
        "error-container": "var(--md-sys-color-error-container)",
        "on-error-container": "var(--md-sys-color-on-error-container)",
        background: "var(--md-sys-color-background)",
        "on-background": "var(--md-sys-color-on-background)",
        surface: "var(--md-sys-color-surface)",
        "on-surface": "var(--md-sys-color-on-surface)",
        "surface-variant": "var(--md-sys-color-surface-variant)",
        "on-surface-variant": "var(--md-sys-color-on-surface-variant)",
        outline: "var(--md-sys-color-outline)",
        "outline-variant": "var(--md-sys-color-outline-variant)",
        shadow: "var(--md-sys-color-shadow)",
        scrim: "var(--md-sys-color-scrim)",
        "inverse-surface": "var(--md-sys-color-inverse-surface)",
        "inverse-on-surface": "var(--md-sys-color-inverse-on-surface)",
        "inverse-primary": "var(--md-sys-color-inverse-primary)",
        "primary-fixed": "var(--md-sys-color-primary-fixed)",
        "on-primary-fixed": "var(--md-sys-color-on-primary-fixed)",
        "primary-fixed-dim": "var(--md-sys-color-primary-fixed-dim)",
        "on-primary-fixed-variant": "var(--md-sys-color-on-primary-fixed-variant)",
        "secondary-fixed": "var(--md-sys-color-secondary-fixed)",
        "on-secondary-fixed": "var(--md-sys-color-on-secondary-fixed)",
        "secondary-fixed-dim": "var(--md-sys-color-secondary-fixed-dim)",
        "on-secondary-fixed-variant": "var(--md-sys-color-on-secondary-fixed-variant)",
        "tertiary-fixed": "var(--md-sys-color-tertiary-fixed)",
        "on-tertiary-fixed": "var(--md-sys-color-on-tertiary-fixed)",
        "tertiary-fixed-dim": "var(--md-sys-color-tertiary-fixed-dim)",
        "on-tertiary-fixed-variant": "var(--md-sys-color-on-tertiary-fixed-variant)",
        "surface-dim": "var(--md-sys-color-surface-dim)",
        "surface-bright": "var(--md-sys-color-surface-bright)",
        "surface-container-lowest": "var(--md-sys-color-surface-container-lowest)",
        "surface-container-low": "var(--md-sys-color-surface-container-low)",
        "surface-container": "var(--md-sys-color-surface-container)",
        "surface-container-high": "var(--md-sys-color-surface-container-high)",
        "surface-container-highest": "var(--md-sys-color-surface-container-highest)",
```

```

    "green-color": "var(--md-extended-color-green-color)",
    "green-on-color": "var(--md-extended-color-green-on-color)",
    "green-color-container": "var(--md-extended-color-green-color-container)",
    "green-on-color-container": "var(--md-extended-color-green-on-color-container)",
    "yellow-color": "var(--md-extended-color-yellow-color)",
    "yellow-on-color": "var(--md-extended-color-yellow-on-color)",
    "yellow-color-container": "var(--md-extended-color-yellow-color-container)",
    "yellow-on-color-container": "var(--md-extended-color-yellow-on-color-container)",
  },
},
},
}

```

Для выбора темы применяем класс к элементу `html.body`:

```
<body class="dark"></body>
```

Допустимые классы:

- dark-high-contrast
- dark-medium-contrast
- dark
- light-high-contrast
- light-medium-contrast
- light

Добавить в файл `input.css`:

```

:root {
  --md-ref-typeface-brand: "Roboto";
  --md-ref-typeface-plain: system-ui;
}

```

Material theme builder почему-то не экспортирует настройки шрифтов. Когда пофиксят - пересмотреть.

1.17.3.6 Iconify

```

npm i -D @iconify/tailwind
npm i -D @iconify/json

```

Добавить в файл `tailwind.config.js`:

```

const { addIconSelectors } = require("@iconify/tailwind");

module.exports = {
  plugins: [addIconSelectors(["mdi", "material-symbols"])],
}

```

Добавить в параметры `addIconSelectors` семейства иконок.

Далее в проекте иконки можно вставлять:

```
<span class="iconify material-symbols--menu-rounded h-5 w-5"></span>
```

1.18 cmp_plc

Шаблон конфигурации ПЛК:

```
use std::time::Duration;

use rsiot::{components::cmp_plc, message::Message};

pub fn config() -> cmp_plc::Config<Custom, fb_main::I, fb_main::Q, fb_main::S> {
    cmp_plc::Config {
        fn_cycle_init,
        fn_input,
        fn_output,
        fb_main: fb_main::FB::new(),
        period: Duration::from_millis(200),
        retention: None,
    }
}

fn fn_cycle_init(input: &mut fb_main::I) {}

fn fn_input(input: &mut fb_main::I, msg: &Message<Custom>) {}

fn fn_output(output: &fb_main::Q) -> Vec<Message<Custom>> {
    let msgs = vec![];

    msgs.into_iter()
        .map(|m| Message::new_custom(Custom::ExampleGroup(m)))
        .collect()
}
```

1.19 cmp_telegram

Компонент для рассылки сообщений через телеграм.

1.19.1 Платформы

target triple	Поддержка
aarch64-linux-android	-
aarch64-unknown-linux-gnu	+
riscv32imc-esp-espidf	-
x86_64-linux-android	-
x86_64-unknown-linux-gnu	+
wasm32-unknown-unknown	-

1.19.2 Конфигурация

1.19.2.1 Config

bot_token	String	Токен бота. Определяется при создании бота через BotFather
bot_token: "token".into(),		
chat_id	i64	Идентификатор чата, в который бот будет отправлять сообщения. Определить идентификатор можно разными способами. Один из способов - через телеграм бот usinfbot . Нужно переслать сообщение из канала в данный бот, в ответе будет идентификатор канала.
chat_id: -1002220119164,		

2 Исполнитель

2.1 WASM (Leptos)

```
use leptos::*;
use tokio::task::LocalSet;

use crate::components::cmp_leptos;
fn main() -> anyhow::Result<()> {
    #[component]
    fn App() -> impl IntoView {
        view! {}
    }

    console_error_panic_hook::set_once();

    configure_logging("info").unwrap();

    // cmp_leptos -----
    let config_leptos = cmp_leptos::Config {
        body_component: || view! { <App/> },
        hostname: "localhost".into(),
    };

    // config_executor -----
    let config_executor = ComponentExecutorConfig {
        buffer_size: 100,
        service: Services::frontend,
        fn_auth: |msg, _| Some(msg),
    };

    // executor -----
    let context = LocalSet::new();
    context.spawn_local(async move {
        ComponentExecutor::<Custom>::new(config_executor)
            .add_cmp(cmp_leptos::Cmp::new(config_leptos))
            .wait_result()
            .await?;
        Ok(()) as anyhow::Result<()>
    });
    spawn_local(context);
    Ok(())
}
```

3 Внешние сервисы

Конфигурация различных внешних сервисов.

3.1 EMQX

MQTT-брокер

3.1.1 docker

```
services:
  emqx:
    container_name: emqx
    healthcheck:
      test: ["CMD", "/opt/emqx/bin/emqx", "ctl", "status"]
      interval: 5s
      timeout: 25s
      retries: 5
    hostname: emqx
    image: emqx:5.7.2 # https://hub.docker.com/_/emqx
    networks:
      - network_internal
    ports:
      - 1883:1883
      - 8083:8083
      - 8084:8084
      - 8883:8883
      - 18083:18083
    profiles:
      - dev
      - target
    volumes:
      - emqx_volume:/opt/emqx/data

networks:
  network_internal:

volumes:
  emqx_volume:
    name: emqx_volume
```

3.2 Go2rtc

Сервис конвертирования видеопотока с видеокамеры.

3.2.1 docker

```
services:
  go2rtc:
    container_name: go2rtc
    hostname: go2rtc
    image: alexxit/go2rtc
    network_mode: host
    privileged: true
    restart: unless-stopped
    profiles:
      - target
      - dev
    volumes:
      - "./config_services/go2rtc:/config"
```

3.2.2 ./config_services/go2rtc/go2rtc.yaml

```
streams:
  tapo: rtsp://administrator:Admin123!@10.0.6.3:554/stream1

api:
  origin: "*"
  listen: ":8003"
```

3.3 Grafana

3.3.1 docker

```
services:
  grafana:
    container_name: grafana
    hostname: grafana
    image: grafana/grafana:10.2.3 # https://hub.docker.com/r/grafana/grafana/tags
    environment:
      - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
      - GF_AUTH_ANONYMOUS_ENABLED=true
      - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
      - GF_SECURITY_ALLOW_EMBEDDING=true
      # настройки источника - TimescaleDB
      - TIMESCALEDB_HOST=timescaledb
      - TIMESCALEDB_PORT=5432
      - TIMESCALEDB_DB_DATA=db_data
      # настройки источника - логгер loki
      - LOKI_HOST=loki
      - LOKI_PORT=3100
      # настройки источника - InfluxDB
      - INFLUXDB_HOST=influxdb
      - INFLUXDB_PORT=8086
      - INFLUXDB_ORG=org
      - INFLUXDB_BUCKET=bucket
      - INFLUXDB_TOKEN=token
    ports:
      - "3000:3000"
    profiles:
      - dev
      - target
    volumes:
      - ./config_services/grafana/datasources:/etc/grafana/provisioning/datasources
      - ./config_services/grafana/dashboards:/etc/grafana/provisioning/dashboards
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    networks:
      - network_internal
```

```
networks:
  network_internal:
```

3.3.2 Файлы конфигурации

3.3.2.1 ./config_services/grafana/datasources/

В папке хранятся файлы для настройки источников данных.

influxdb.yaml:

```
apiVersion: 1

datasources:
  - name: InfluxDB
    type: influxdb
    access: proxy
    url: http://${INFLUXDB_HOST}:${INFLUXDB_PORT}
    jsonData:
      version: Flux
      organization: ${INFLUXDB_ORG}
      defaultBucket: ${INFLUXDB_BUCKET}
      tlsSkipVerify: true
    secureJsonData:
      token: ${INFLUXDB_TOKEN}
```

loki.yaml:

```
apiVersion: 1

datasources:
  - name: loki
    type: loki
    access: proxy
    orgId: 1
    url: http://${LOKI_HOST}:${LOKI_PORT}
    basicAuth: false
    isDefault: true
    version: 1
    editable: false
```

timescaledb.yaml:

```
apiVersion: 1
```

```
datasources:
- name: timescaledb
  type: postgres
  url: ${TIMESCALEDB_HOST}:${TIMESCALEDB_PORT}
  user: postgres
  secureJsonData:
    password: "postgres"
  jsonData:
    database: ${TIMESCALEDB_DB_DATA}
    sslmode: "disable" # disable/require/verify-ca/verify-full
    maxOpenConns: 100 # Grafana v5.4+
    maxIdleConns: 100 # Grafana v5.4+
    maxIdleConnsAuto: true # Grafana v9.5.1+
    connMaxLifetime: 14400 # Grafana v5.4+
    postgresVersion: 1500 # 903=9.3, 904=9.4, 905=9.5, 906=9.6, 1000=10
    timescaledb: true
  editable: false
```

3.3.2.2 ./config_services/grafana/dashboards/

В папке хранятся все дашборды. Структура папок переносится в структуру дашбоардов. В корне папки нужно разместить файл config.yaml:

```
apiVersion: 1

providers:
- name: dashboards
  type: file
  updateIntervalSeconds: 5
  options:
    path: /etc/grafana/provisioning/dashboards
    foldersFromFilesStructure: true
```

3.4 InfluxDB (v2)

3.4.1 docker

```
services:
  influxdb:
    container_name: influxdb
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=Admin123!
      - DOCKER_INFLUXDB_INIT_ORG=org
      - DOCKER_INFLUXDB_INIT_BUCKET=bucket
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=token
    hostname: influxdb
    image: influxdb:2.7.6 # https://hub.docker.com/_/influxdb
    networks:
      - network_internal
    ports:
      - "8086:8086"
    volumes:
      - influxdb_data:/var/lib/influxdb2
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro

networks:
  network_internal:

volumes:
  influxdb_data:
    name: influxdb_data
# TODO - healthcheck
```


3.5 InfluxDB (v3)

3.6 Loki

Для проверки запуска можно открыть в браузере:

- <http://localhost:3100/metrics>
- <http://localhost:3100/ready>

3.6.1 docker

services:

loki:

command: -config.file=/etc/loki/local-config.yaml

container_name: loki

healthcheck:

test: wget --spider http://localhost:3100/ready

interval: 10s

timeout: 20s

retries: 15

hostname: loki

image: grafana/loki:2.9.2 # https://hub.docker.com/r/grafana/loki/tags?page=1&name=2.

networks:

- network_internal

ports:

- "\${LOKI_PORT}:3100"

profiles:

- dev

- target

volumes:

- loki_data:/loki

- /etc/timezone:/etc/timezone:ro

- /etc/localtime:/etc/localtime:ro

volumes:

loki_data:

name: loki_data

networks:

network_internal:

3.7 Portainer

3.7.1 docker

```
services:
  portainer:
    container_name: portainer
    hostname: portainer
    image: portainer/portainer-ce:latest
    ports:
      - "${PORTAINER_PORT}:9000"
    profiles:
      - target
    restart: always
    volumes:
      - portainer_data_volume:/data
      - /var/run/docker.sock:/var/run/docker.sock
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro

volumes:
  portainer_data_volume:
    name: portainer_data_volume
```

3.8 Redis

3.8.1 docker

```
services:
  redis:
    container_name: redis
    healthcheck:
      test: redis-cli --raw incr ping
      interval: 5s
      timeout: 5s
      retries: 5
    hostname: redis
    image: redis/redis-stack:latest
    networks:
      - network_internal
    ports:
      - "${REDIS_PORT}:6379" # порт Redis
      - "${REDIS_PORT_UI}:8001" # порт UI
    volumes:
      - redis_data:/data # для сохранения данных
      - ./services/redis/redis.conf:/redis-stack.conf # путь к файлу конфигурации
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro

networks:
  network_internal:

volumes:
  redis_data:
    name: redis_data
```

3.8.2 Файлы конфигурации

3.8.2.1 redis.conf

Для сохранения сообщений при перезапуске:

```
appendonly yes
```

3.9 Rust

Запуск программ на rust в контейнерах docker

3.9.1 docker (бекенд)

docker-compose.yml:

```
services:
  backend:
    command: ./backend
    container_name: backend
    depends_on:
      redis:
        condition: service_healthy
        restart: true
      loki:
        condition: service_healthy
        restart: true
    hostname: backend
    image: ubuntu:noble
    networks:
      - network_internal
    environment:
      - RUST_LOG=info
    profiles:
      - target
    volumes:
      - ./backend:/backend
      - ../.env:/.env
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro

networks:
  network_internal:
```

3.9.2 docker (cmp_leptos)

docker-compose.yml:

```
services:
  frontend:
    container_name: frontend
    hostname: frontend
    image: nginx
    networks:
      - network_internal
    ports:
      - "8000:80"
    profiles:
      - target
    volumes:
      - ./frontend/dist:/usr/share/nginx/html
      - ./frontend/nginx.conf:/etc/nginx/conf.d/default.conf

networks:
  network_internal:
# TODO - healthcheck

nginx.conf:

server {
    listen 80;
    listen [::]:80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html =404;
    }
}
```

3.10 Sentryshot

Сохранение потока с видеокамеры. [Ссылка на репозиторий](#).

3.10.1 docker

```
services:
  sentryshot:
    shm_size: 500m
    image: codeberg.org/sentryshot/sentryshot:v0.2.17
    ports:
      - 2020:2020
    environment:
      - TZ=Europe/Minsk
    profiles:
      - target
    volumes:
      - ./config_services/sentryshot/configs:/app/configs
      - ./config_services/sentryshot/storage:/app/storage
```

Проверить версию - <https://codeberg.org/SentryShot/sentryshot/releases>.

3.10.2 Файлы конфигурации

3.10.2.1 ./sentryshot/configs/sentryshot.toml

Проверить max_disk_usage.

```
# Port app will be served on.
port = 2020

# Directory where recordings will be stored.
storage_dir = "/app/storage"

# Directory where configs will be stored.
config_dir = "/app/configs"

# Directory where the plugins are located.
plugin_dir = "/app/plugins"

# Maximum allowed storage space in GigaBytes.
# Recordings are delete automatically before this limit is exceeded.
max_disk_usage = 100

# PLUGINS

# Authentication. One must be enabled.

# Basic Auth.
[[plugin]]
name = "auth_basic"
enable = false

# No authentication.
[[plugin]]
name = "auth_none"
enable = true

# Motion detection.
# Documentation ./plugins/motion/README.md
[[plugin]]
name = "motion"
enable = false

# TFlite object detection.
# Enabling will generate a `tflite.toml` file.
[[plugin]]
name = "tflite"
enable = false

# Thumbnail downscaling.
# Downscale video thumbnails to improve page load times and data usage.
[[plugin]]
name = "thumb_scale"
enable = false
```

3.10.2.2 ./sentryshot/configs/monitors/

В папке хранятся файлы конфигурации для каждой камеры. Пример файла для камеры RTSP:

```
{
  "alwaysRecord": true,
  "enable": true,
  "id": "tapo",
  "name": "tapo",
  "source": "rtsp",
  "sourcertsp": {
    "mainStream": "rtsp://administrator:Admin123!@192.168.31.3:554/stream1",
    "protocol": "tcp"
  },
  "videoLength": 15
}
```

3.11 SurrealDB

3.11.1 docker

```
services:
  surrealdb:
    command: start --user root --pass root file:/data/database.db
    container_name: surrealdb
    hostname: surrealdb
    image: surrealdb/surrealdb:latest
    networks:
      - network_internal
    ports:
      - "${SURREALDB_PORT}:8000"
    user: root
    volumes:
      - surrealdb_data:/data

networks:
  network_internal:

volumes:
  surrealdb_data:
    name: surrealdb_data
# TODO - healthcheck
```


3.12 SystemD

Пример создания файла для автозапуска сервисов с помощью SystemD

Файл _PROJECT_.service:

```
[Unit]
Description=PROJECT_DESC
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/home/user/PROJECT_FOLDER
ExecStart=/home/user/.cargo/bin/nu scripts/target-start.nu
ExecStop=/home/user/.cargo/bin/nu scripts/target-stop.nu
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
```

Установить сервис на целевой машине:

```
sudo mv _PROJECT_.service /etc/systemd/system
sudo systemctl daemon-reload
sudo systemctl enable _PROJECT_
sudo systemctl start _PROJECT_
```

3.13 TimescaleDB

3.13.1 docker

```
services:
  timescaledb:
    command: postgres
    -c config_file=/etc/postgresql/postgresql.conf
    -c hba_file=/etc/postgresql/pg_hba.conf
    container_name: timescaledb
    healthcheck:
      test: pg_isready -d db_prod
      interval: 30s
      timeout: 60s
      retries: 5
      start_period: 80s
    hostname: timescaledb
    image: timescale/timescaledb:2.12.2-pg15
    networks:
      - network_internal
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - "5432:5432"
    profiles:
      - dev
      - target
    volumes:
      - ./timescaledb/postgresql.conf:/etc/postgresql/postgresql.conf
      - ./timescaledb/pg_hba.conf:/etc/postgresql/pg_hba.conf
      - ./timescaledb/init.sql:/docker-entrypoint-initdb.d/init.sql
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro

networks:
  network_internal:
```

3.13.2 postgresql.conf

```
listen_addresses = '*'
max_locks_per_transaction = 10000
```

3.13.3 pg_hba.conf

```
local all all trust
host all all 0.0.0.0/0 trust
```

3.13.4 init.sql

```
CREATE DATABASE db_conf;
CREATE DATABASE db_data;

\c db_data
CREATE EXTENSION IF NOT EXISTS timescaledb;

-- enum agg_type
CREATE TYPE agg_type AS ENUM (
  'curr',
  'first',
  'inc',
  'sum',
  'mean',
  'min',
  'max'
);

-- table raw
CREATE TABLE raw (
  ts          TIMESTAMPTZ      NOT NULL,
  entity      TEXT             NOT NULL,
  attr        TEXT             NOT NULL,
  value       DOUBLE PRECISION NULL,
  agg         AGG_TYPE         NOT NULL,
  aggtss      TIMESTAMPTZ      NULL,
  aggnext     AGG_TYPE[]       NULL,
  UNIQUE (ts, entity, attr, agg)
);
SELECT create_hypertable(
  'raw', 'ts',
  chunk_time_interval => INTERVAL '24 hours'
);
ALTER TABLE raw SET (
```

```
        timescaledb.compress,  
        timescaledb.compress_segmentby='entity, attr, agg'  
    );  
SELECT add_compression_policy('raw', INTERVAL '100000 hours');  
  
-- agg_30min  
CREATE TABLE agg_30min (LIKE raw);  
  
-- create databases for test  
CREATE DATABASE db_data_test WITH TEMPLATE db_data;  
CREATE DATABASE db_conf_test WITH TEMPLATE db_conf;
```