ICOweb Application

A frontend/client developed in conjunction with the corresponding ICOapi backend. It is designed to be run in any form of modern browser.

Design / Wireframes: Miro Board

Languages: Vue, TypeScript, (HTML/CSS)

Design Framework: PrimeVue, TailwindCSS

Bundler: Vite

Installation

This repository can be setup manually for Windows and Linux or using the installation script for Linux.

Prerequisites

- Node.js / NPM, from the official NodeJS Website

Manual Installation (Development)

As with any other NPM project, after cloning the repository and ensuring a valid Node installation, run:

`npm install`

Service Installation (Linux)

For Linux, there is an installation script which sets the directory for the actual installation, the directory for the systemd service and the used systemd service name as well as the Node version. The (sensible) defaults are:

```
SERVICE_NAME="icoweb"
INSTALL_DIR="/etc/icoweb"
SERVICE_PATH="/etc/systemd/system"
NODE_VERSION="18.19.0"
FORCE_REINSTALL=false
```

After checking, run the script to install normally:

```
./install
```

Or, if you want to delete existing installations and do a clean reinstall, add the `--force` flag:

```
./install --force
```

Configuration / Environment Variables

This application has two main forms of configuration: environment variables and auto-generated metadata types.

Environment Variables

The application expects a `.env` file in the root directory. It handles the main configuration of the backend application.

> *All variables containing `API_` indicate that there is a counterpart in the API side environment variables. This is to show that changes here most likely need to be propagated to the backend.*

Client/API Connection Settings

These settings determine all forms of client/API communication details.

The main REST API is versioned and does *NOT* use SSL at the moment.

```
VITE_API_PROTOCOL=http
VITE_API_HOSTNAME="0.0.0.0"
VITE_API_PORT=33215
VITE_API_VERSION=v1
```

The WebSocket is for streaming data. It only requires a `VITE_API_WS_PROTOCOL` variable akin to `VITE_API_PROTOCOL` which decided between SSL or not.

```
VITE_API_WS_PROTOCOL=ws
```

Enable/Disable Settings

To make the application configurable, pages and features can be enabled and disabled separately. By default, all pages are shown and no features are enabled - the `.env` file can override that.

```
VITE_APPLICATION_DISABLE_PAGES="Config,Analyze"
VITE_APPLICATION_ENABLE_FEATURES="Cloud,Meta"
```

`VITE_APPLICATION_DISABLE_PAGES` takes the names of the pages to be disabled, which hides them from the menu. `VITE_APPLICATION_ENABLE_FEATURES` takes predefined tags to enable.

- `Cloud` enables the usage of the Trident API data storage
- `Meta` enables the addition of metadata for measurements

Branding Settings

To add branding to the client, a logo with corresponding ALT text can be set.

```
VITE_APPLICATION_EXTRA_LOGO="cirp.png"
VITE_APPLICATION_EXTRA_LOGO_ALT="CIRP"
```

The logo file must be placed in the `public/extra` folder and will be displayed above the IFT logo in the menu.

Type Generation

This repository is based on typescript and relies heavily on the `openapi-ts` package to generate type declarations in `src/client` from the backend directly. To update the type declarations, **ensure the API is running** and run:

`npm run generate-client`

Metadata Parsing

To support the usage of arbitrary metadata when creating measurements, a configuration system has been set up. This system starts as en Excel file in which all metadata fields are defined. This file is then parsed into a YAML file, from which it can be used further.

This repository holds the `.xlsx` master file and the script to generate the `.yaml` file from it. Run the parser with:

```
npm run generate-config
```

This script expects the Excel file to be in the project root and places the parsed YAML file into `public/config`.

After you have parsed the Excel file, run the `generate_metadata.py` file in the `icoapi` repository to create the backend types for the API. Finally, you can run the openAPI command from above to get the types from the backend.

> *This may seem convoluted, but this way the metadata settings are stored in the client project (xlsx, yaml) where they semantically belong, but the types still all come from the openAPI specification from the backend to provide consistency.*

Run

To run the client locally, use:

```
npm run start
```

Note: The browser must have CORS disabled. When using Chrome, run with the flags

`--disable-web-security --user-data-dir="<Some Writable Folder>"` to achieve this.

Known Issues

A Running list of issues and who reported them. This will not track every single issue, but rather complete requests/complaints.

2025-04-14 by TT

Per screen recording, on Firefox, using RevPi.

- [x] STU & STH MAC do not autofill
- [x] ADC drawer styling issues (Firefox?): Closing button overlaps inputs
- [x] Streaming gets choppy after ~10s (might be recording issue)
- [x] IFT Value is not shown after "stop" button was called
- [x] Starting a new measurement does not clear the graph screen
- [x] Starting a new measurement and connecting the stream shows two confused graphs
- [x] Setting metadata values to 0 should be allowed

Planned Features

This is the list of features to be implemented.

Concerning "Analyze" Tab

- [x] **Create Analyze Tab**: Separate tab where files can be loaded in and analyzed
- [x] **Graph: Window selection**: Make a window selection for zooming in
- [ ] **Graph: Better crosshair for point inspection**
- [ ] **Graph: FFT**: Make FFT available for dataset
- [ ] **Dataloss**: Display dataloss in analyze tab
  - [ ] Make "intelligent fill" (fills lost values with past values) create a new file
- [x] **Performance**: Large file sizes?
  - [x] Loading indicator
- [ ] **Scales**: Make the graph scales represent the holder's actual unit and range

Notes on crosshair and window selection: Both are enabled by the plugin chartjs-plugin-crosshair and would be perfectly suitable. However, when using the plugin, upon zooming the datasets disappear. This seems to be connectect to the `type: linear` property in the chart options, as without this, it works - but with an unformated scale.

Concerning "Measure" Tab

- [x] **Graph: Zoom only X**
- [x] ~~**Graph**: Fixed time shows whole timeframe at once with fixed number of points~~ Keep the scrolling behavior
- [x] **Acquisition Time**: Determine best way to set acquisition time
  - Note: according to JG, just seconds input is fine.
- [x] **Performance**: Maximum length of acquisition?
  - Note: a single axis measurement of 400MB was possible, so no worries there.
- [x] **Scales**: Make the graph scales represent the holder's actual unit and range

Concerning General Features

- [x] **Dataloss**: make dataloss visible in measurement tab
- [x] **Renaming**: For security reasons, make `rename` into prompt (no accidental renames of other STHs)
- [x] **Export/Import config**: Put all relevant settings into config file and make it importable/exportable
- [x] **Storage**: Check storage capacity and warn accordingly

Concerning "Config" Tab

- [x] **Tool Holder**: Split into default and custom holder templates
  - [x] Make default holders hardcoded
  - [ ] Make default holders with picture
  - [ ] **Sketches**: Create and show informational sketches

Feature Request List

This list is a loose collection of feedback given by people involved in the STH project.

Concerning Sensor Data

- [ ] **Raw data option**: create the option to look at the raw data without any conversion to @g_0 or scaling to sensor range
- [ ] **Reference Voltage**: Use first channel reference voltage for all channels
- [ ] **Linear transformation**: Create option to assign a linear transformation per sensor to a holder

Concerning General Features

- [ ] **Hardware**: Firmware update of STH / STU via GUI
- [x] **Metadata**: Add ability to add metadata to STU / STH.
  - e.g. Machine, part, machine data, trial number, ...
  - make this available in the measurement tab
- [ ] **MQTT**: Provide interface so a certain MQTT topic can be subscribed to and added to the measurement stream data

Concerning "Measure" Tab

- [ ] **KPIs**: Add support for more KPIs (IFT value is considered a KPI)
- [ ] **Rule Engine**: Create rule engine rules in GUI and export them
- [ ] **Threshold**: Add ability to set threshold and get 0/1 if passed