IsoPrüfi

None

None

None

Table of contents

1. Welcome to IsoPrüfi

We are happy that you are here 🥳🎉

August 24, 2025

DianaTin23, deadmade

2. Contribute & Build

2.1 How do I contribute as a developer?

READ THIS GUIDE BEFORE CONTRIBUTING

Since our project is secured by two pre-commit hocks, it is important to set up the project correctly before contributing.

This is done as followed:

Clone the project

`git clone https://github.com/deadmade/IsoPruefi.git`

Make sure you have installed the following packages globally.

- Python: Needed for MkDocs
- Node Package Manager: Used to install needed dependencies for pre-commit hooks
- .NET 9.0 SDK: Used for our Rest-API
- Docker

After you've cloned the repo make sure to install all needed packages for the hooks via:

`npm i`

and run:

`npm run init`

Now it should be configured 🚀

To get the development environment up and running, follow these steps:

1. Open a terminal, navigate to the `IsoPrüfi` directory, and run:

```
docker compose up
```

1. Once the containers are running, create an admin token for InfluxDB:

```
docker exec -it influxdb influxdb3 create token --admin
```

1. Copy the generated token string.
2. Create a `config.json` file at the following location:

```
IsoPruefi/isopruefi-docker/influx/explorer/config
```

1. Add the following content to `config.json`, replacing `"your-token-here"` with the copied token:

```
{
  "DEFAULT_INFLUX_SERVER": "http://host.docker.internal:8181",
  "DEFAULT_INFLUX_DATABASE": "IsoPrüfi",
  "DEFAULT_API_TOKEN": "your-token-here",
  "DEFAULT_SERVER_NAME": "IsoPrüfi"
}
```

1. Run dotnet user-secrets set "Influx:InfluxDBToken" "" --project isopruefi-backend\MQTT-Receiver-Worker\MQTT-Receiver-Worker.csproj
2. Restart the Containers

2.2 Arduino Set Up

2.2.1 Hardware

- MKR WiFi 1010
- Analog Devices ADT7410 Breakout
- DS3231 RTC
- SD Card Module

2.2.2 Software

`⚠️ Important: Always open the Arduino firmware folder (e.g., code/arduino/) as a PlatformIO Project (via Open Project or Pick a folder in the PlatformIO sidebar). Otherwise, dependencies from platformio.ini might not be detected and you may see false errors in the editor.`

To work on the Arduino/PlatformIO part of the project:

1. Install the PlatformIO Extension in Visual Studio Code
2. Open the folder code/arduino/ (or wherever the firmware is located)
3. Build and upload the firmware using the PlatformIO toolbar or PlatformIO terminal
4. Make sure your board is connected and properly selected in platformio.ini

```
[env:mkrwifi1010]
platform = atmelsam
board = mkrwifi1010
framework = arduino
lib_deps =
    arduino-libraries/WiFiNINA
    adafruit/Adafruit ADT7410 Library
    adafruit/RTClib
    arduino-libraries/ArduinoMqttClient
    greiman/SdFat
    gyverlibs/UnixTime
    bblanchon/ArduinoJson@^7.4.2
```

Tips:

- PlatformIO installs the required libraries automatically on first build
- To run the programm run `pio run -e mkrwifi1010` in the PlatformIO terminal
- To flash the Arudion with new code run `pio run -e mkrwifi1010 --target upload` in the PlatformIO terminal
- The main firmware entry point is located at src/main.cpp
- Use the Serial Monitor (🔌) to debug via USB
- To run the all unit tests run `pio test -e native` in the PlatformIO terminal

Happy Coding 😊

August 31, 2025

DianaTin23, deadmade, deadmade

3. Guidelines / Conventions

3.1 General Formatting Guidelines

- Preserve Settings: Follow the existing project formatting rules.
- Automatic Formatting: Regularly use Rider's automatic formatting function (Ctrl+Alt+L).

3.2 Code Layout

- Line Length: Maximum of 120 characters per line.
- Indented Blocks: Use tabs or 4 spaces for indentation (depending on project settings).
- Blank Lines: Use blank lines to separate logical code blocks.

3.3 Indentation and Spacing

- Indent Blocks: Always use 4 spaces per indentation level.
- Braces: Opening braces on the same line as the statement, closing braces on a new line.
- Operator Spacing: Add spaces around operators like +, -, *, /, =, ==, etc.
- Commas and Semicolons: Add a space after commas and semicolons, not before.

3.4 Naming Conventions

- Classes and Methods: Use PascalCase.
- Variables and Fields: Use camelCase.
- Constants: Use SCREAMING_SNAKE_CASE.

3.5 Documentation Comments

If not obvious, or the method is more than 5 lines, it should be commented.

- Single-line Comments: Use // for single-line comments
- Multi-line Comments: Use /* ... */ for multi-line comments.
- Documentation Comments: Use /// for documentation comments.

3.5.1 Documentation Comments (C#)

We use the xml documentation convention by Microsoft

In short: You can use the following tags structures in your documentation comment to specify properties of the following code:

- `<summary>Your code summary</summary>`
- `<param name="str">Describe parameter.</param>` : Usage may also be nested within summary
- `<code>Use a codeblock within</code>`
- `<example>Put a example here</example>`

There are plenty more tags. You can even reference other doc segments.
If you need other tags, take a look here

3.5.2 Doxygen (C/C++)

For Arduino and C++ code, we use Doxygen style comments:

- `/// Brief description`
- `/** Detailed description */`
- `@param name Description of parameter`
- `@return Description of return value`
- `@code ... @endcode` for code blocks
- `@example ...` for examples

More tags: Doxygen documentation

3.5.3 TypeDoc (TypeScript)

For TypeScript, we use TypeDoc style comments:

- `/** Summary of the function or class */`
- `@param name Description of parameter`
- `@returns Description of return value`
- `@example Example usage`
- `@see Reference to related code or docs`

More tags: TypeDoc tags

3.6 Usings and Namespaces

- Sorting: Sort usings alphabetically and group by system namespace.
- Removing: Remove unused usings.
- Namespace: A file should contain a single namespace.

3.7 Error Handling

- Exceptions: Always catch specific exceptions when possible.

3.8 Unit Tests

Every method should be unit testable and have a unit test for it.

3.9 Commit Messages

3.9.1 How should my commit messages look like?

Our repo follows the Conventional Commits guidelines.

Allowed commit types are specified as following:

- feat -> Introduces a new features
- fix -> Fixes a bug
- docs -> Updates on the docs
- chore -> Updates a grunt task; no-production code change
- style -> Formatting code style (missing semicolon, prettier execution, etc)
- refactor -> Refactoring existing code e.g. renaming a variable, reworking a function
- ci -> CI Tasks e.g. adding a hook
- test -> Adding new tests, refactoring tests, deleting old tests
- revert -> Revert old commits
- perf -> Performance related refactoring, without functional changes

3.10 Branch Naming

Your branche names should follow this style:

[commit-type]/[topic-of-branch-seperated-by-hyphen]

F.e. if you want to introduce a new cool type of button your branch should have the name:

feat/cool-new-button

August 31, 2025

DianaTin23, deadmade, deadmade

4. Code Documentation

4.1 Assembly Rest-API

4.1.1 Namespace Rest_API

- Program

4.1.2 Namespace Rest_API.Controllers

- AuthenticationController
- LocationController
- TemperatureDataController
- TopicController
- UserInfoController

4.1.3 Namespace Rest_API.Helper

- HealthCheck
- StringTools

4.1.4 Namespace Rest_API.Models

- ChangePassword
- JwtToken
- Login
- Register
- Roles
- SensorData
- TemperatureData
- TemperatureDataOverview

4.1.5 Namespace Rest_API.Seeder

- SeedUser

4.1.6 Namespace Rest_API.Services.Auth

- AuthenticationService
- IAuthenticationService

4.1.7 Namespace Rest_API.Services.Temp

- ITempService
- TempService

4.1.8 Namespace Rest_API.Services.Token

- ITokenService
- TokenService

4.1.9 Namespace Rest_API.Services.User

- IUserService
- UserService

September 2, 2025

4.2 Assembly MQTT-Receiver-Worker

4.2.1 Namespace MQTT_Receiver_Worker

- HealthCheck
- NullMetaListConverter
- Program
- Worker

4.2.2 Namespace MQTT_Receiver_Worker.MQTT

- Connection
- MqttHealthCheck
- Receiver

4.2.3 Namespace MQTT_Receiver_Worker.MQTT.Interfaces

- IConnection
- IReceiver

4.2.4 Namespace MQTT_Receiver_Worker.MQTT.Models

- TempSensorMeta
- TempSensorReading

September 2, 2025

4.3 Assembly MQTT-Sender

4.3.1 Namespace MQTT_Sender

- Connection

September 2, 2025

4.4 Assembly Database

4.4.1 Namespace Database.EntityFramework

- ApplicationDbContext

4.4.2 Namespace Database.EntityFramework.Enums

- SensorType

4.4.3 Namespace Database.EntityFramework.Models

- ApiUser
- CoordinateMapping
- TokenInfo
- TopicSetting

4.4.4 Namespace Database.Migrations

- RefactorMigrations

4.4.5 Namespace Database.Repository.CoordinateRepo

- CoordinateRepo
- ICoordinateRepo

4.4.6 Namespace Database.Repository.InfluxRepo

- IInfluxRepo
- InfluxRetryService

4.4.7 Namespace Database.Repository.InfluxRepo.Influx

- InfluxHealthCheck
- InfluxRepo

4.4.8 Namespace Database.Repository.InfluxRepo.InfluxCache

- CachedInfluxHealthCheck
- CachedInfluxRepo

4.4.9 Namespace Database.Repository.SettingsRepo

- ISettingsRepo
- SettingsRepo

4.4.10 Namespace Database.Repository.TokenRepo

- ITokenRepo
- TokenRepo

September 2, 2025

4.5 Assembly UnitTests

4.5.1 Namespace UnitTests.Controllers

- AuthenticationControllerTests
- LocationControllerTests
- TemperatureDataControllerTests
- TopicControllerTests
- UserInfoControllerTests

4.5.2 Namespace UnitTests.MqttReceiver

- ConnectionTests
- NullMetaListConverterTests
- ReceiverTests
- TempSensorReadingTests
- WorkerTests

4.5.3 Namespace UnitTests.Repositories

- InfluxRepoTests
- SettingsRepoTests

4.5.4 Namespace UnitTests.Services

- AuthenticationServiceTests
- TempServiceTests
- TokenServiceTests
- UserServiceTests

September 2, 2025

4.6 Contents pages

- Global contents
- Files
- Structures
- Modules
- Directories

4.7 Index pages

- Global index
- Files
- Structures
- Modules
- Directories

September 2, 2025

4.8 Frontend

isopruefi-frontend v1.0.0

4.8.1 isopruefi-frontend v1.0.0

Modules

- api/api-client
- api/clients
- App
- auth/AuthForm
- auth/SignIn
- auth/SignUp
- components/Navbar
- components/ProtectedRoute
- components/Weather
- main
- pages/AdminPage
- pages/UserPage
- pages/Welcome
- utils/authApi
- utils/config
- utils/tokenHelpers

September 2, 2025

4.8.2 Api

api-client

isopruefi-frontend v1.0.0

isopruefi-frontend / api/api-client

API/API-CLIENT

Enumerations

- SensorType

Classes

- ApiException
- ApiUser
- AuthenticationClient
- ChangePassword
- CoordinateMapping
- IdentityUser
- IdentityUserOfString
- JwtToken
- LocationClient
- Login
- ProblemDetails
- Register
- SensorData
- TemperatureData
- TemperatureDataClient
- TemperatureDataOverview
- TopicClient
- TopicSetting
- UserInfoClient

Interfaces

- FileResponse
- IApiUser
- IChangePassword
- ICoordinateMapping
- IIdentityUser
- IIdentityUserOfString
- IJwtToken
- ILogin
- IProblemDetails
- IRegister
- ISensorData

- ITemperatureData
- ITemperatureDataOverview
- ITopicSetting

September 2, 2025

CLASSES

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ApiException

Class: ApiException

Defined in: api/api-client.ts:1797

Extends

- `Error`

Constructors Constructor

> **new ApiException**(`message` , `status` , `response` , `headers` , `result`): `ApiException`

Defined in: api/api-client.ts:1804

Parameters message

`string`

status

`number`

response

`string`

headers result

`any`

Returns

`ApiException`

Overrides

`Error.constructor`

Properties headers

> **headers**: `object`

Defined in: api/api-client.ts:1801

Index Signature

[`key` : `string`]: `any`

---

isApiException

> `protected` **isApiException**: `boolean` = `true`

Defined in: api/api-client.ts:1814

---

message

> **message**: `string`

Defined in: api/api-client.ts:1798

Overrides

`Error.message`

---

response

> **response**: `string`

Defined in: api/api-client.ts:1800

---

result

> **result**: `any`

Defined in: api/api-client.ts:1802

---

status

> **status**: `number`

Defined in: api/api-client.ts:1799

Methods isApiException()

> `static` **isApiException**(`obj`): `obj is ApiException`

Defined in: api/api-client.ts:1816

Parameters obj

`any`

Returns

`obj is ApiException`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ApiUser

Class: ApiUser

Defined in: api/api-client.ts:1762

Represents an application user in the system

Extends

- `IdentityUser`

Implements

- `IApiUser`

Constructors Constructor

> **new ApiUser**(`data?`): `ApiUser`

Defined in: api/api-client.ts:1764

Parameters data?

`IApiUser`

Returns

`ApiUser`

Overrides

`IdentityUser`.`constructor`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1639

Gets or sets the number of failed login attempts for the current user.

Implementation of

`IApiUser`.`accessFailedCount`

Inherited from

`IdentityUser`.`accessFailedCount`

---

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1627

A random value that must change whenever a user is persisted to the store

Implementation of

`IApiUser`.`concurrencyStamp`

Inherited from

`IdentityUser.concurrencyStamp`

---

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1617

Gets or sets the email address for this user.

Implementation of

`IApiUser.email`

Inherited from

`IdentityUser.email`

---

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1621

Gets or sets a flag indicating if a user has confirmed their email address.

Implementation of

`IApiUser.emailConfirmed`

Inherited from

`IdentityUser.emailConfirmed`

---

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1611

Gets or sets the primary key for this user.

Implementation of

`IApiUser.id`

Inherited from

`IdentityUser.id`

---

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1637

Gets or sets a flag indicating if the user could be locked out.

Implementation of

`IApiUser.lockoutEnabled`

Inherited from

`IdentityUser.lockoutEnabled`

---

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1635

Gets or sets the date and time, in UTC, when any user lockout ends.

Implementation of

`IApiUser.lockoutEnd`

Inherited from

`IdentityUser.lockoutEnd`

---

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1619

Gets or sets the normalized email address for this user.

Implementation of

`IApiUser.normalizedEmail`

Inherited from

`IdentityUser.normalizedEmail`

---

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1615

Gets or sets the normalized user name for this user.

Implementation of

`IApiUser.normalizedUserName`

Inherited from

`IdentityUser.normalizedUserName`

---

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1623

Gets or sets a salted and hashed representation of the password for this user.

Implementation of

`IApiUser.passwordHash`

Inherited from

`IdentityUser.passwordHash`

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1629

Gets or sets a telephone number for the user.

Implementation of

`IApiUser.phoneNumber`

Inherited from

`IdentityUser.phoneNumber`

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1631

Gets or sets a flag indicating if a user has confirmed their telephone address.

Implementation of

`IApiUser.phoneNumberConfirmed`

Inherited from

`IdentityUser.phoneNumberConfirmed`

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1625

A random value that must change whenever a users credentials change (password changed, login removed)

Implementation of

`IApiUser.securityStamp`

Inherited from

`IdentityUser.securityStamp`

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1633

Gets or sets a flag indicating if two factor authentication is enabled for this user.

Implementation of

`IApiUser.twoFactorEnabled`

Inherited from

`IdentityUser.twoFactorEnabled`

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1613

Gets or sets the user name for this user.

Implementation of

`IApiUser.userName`

Inherited from

`IdentityUser.userName`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1768

Parameters _data?

`any`

Returns

`void`

Overrides

`IdentityUser.init`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1779

Parameters data?

`any`

Returns

`any`

Overrides

`IdentityUser.toJSON`

---

fromJS()

> `static` **fromJS**(`data`): `ApiUser`

Defined in: api/api-client.ts:1772

Parameters data

`any`

Returns

`ApiUser`

Overrides

`IdentityUser.fromJS`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / AuthenticationClient

Class: AuthenticationClient

Defined in: api/api-client.ts:11

Constructors Constructor

> **new AuthenticationClient**(`baseUrl?` , `http?`): `AuthenticationClient`

Defined in: api/api-client.ts:16

Parameters baseUrl?

`string`

http? fetch Returns

`AuthenticationClient`

Properties jsonParseReviver

> `protected` **jsonParseReviver**: `undefined` | (`key` , `value`) `=>` `any` `=` `undefined`

Defined in: api/api-client.ts:14

Methods login()

> **login**(`input`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:26

Authenticates a user and returns a JWT token for API access.

Parameters input

`Login`

The login credentials containing username and password.

Returns

`Promise` \< `FileResponse` >

Authentication successful. Returns JWT access token and refresh token.

---

processLogin()

> `protected` **processLogin**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:46

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

---

processRefresh()

> `protected` **processRefresh**(`response`): `Promise` \< `void` >

Defined in: api/api-client.ts:156

Parameters response

`Response`

Returns

`Promise` \< `void` >

processRegister()

> `protected` **processRegister**(`response`): `Promise` \< `void` >

Defined in: api/api-client.ts:92

Parameters response

`Response`

Returns

`Promise` \< `void` >

refresh()

> **refresh**(`token`): `Promise` \< `void` >

Defined in: api/api-client.ts:137

Refreshes an expired JWT access token using a valid refresh token.

Parameters token

`JwtToken`

The JWT token object containing both the expired access token and valid refresh token.

Returns

`Promise` \< `void` >

Token refresh successful. Returns new access and refresh tokens.

register()

> **register**(`input`): `Promise` \< `void` >

Defined in: api/api-client.ts:73

Registers a new user in the system. Admin access required.

Parameters input

`Register`

The registration data containing username and password for the new user.

Returns

`Promise`\<`void`>

User registered successfully.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ChangePassword

Class: ChangePassword

Defined in: api/api-client.ts:1551

Represents a request to change a user's password.

Implements

- `IChangePassword`

Constructors Constructor

> **new ChangePassword**(`data?`): `ChangePassword`

Defined in: api/api-client.ts:1562

Parameters data?

`IChangePassword`

Returns

`ChangePassword`

Properties currentPassword?

> `optional` **currentPassword**: `string`

Defined in: api/api-client.ts:1557

Gets or sets the current password of the user.

Implementation of

`IChangePassword` . `currentPassword`

---

newPassword?

> `optional` **newPassword**: `string`

Defined in: api/api-client.ts:1560

Gets or sets the new password to be set for the user.

Implementation of

`IChangePassword` . `newPassword`

---

userId?

> `optional` **userId**: `string`

Defined in: api/api-client.ts:1554

Gets or sets the unique identifier of the user whose password is to be changed.

Implementation of

`IChangePassword` . `userId`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1571

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1586

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `ChangePassword`

Defined in: api/api-client.ts:1579

Parameters data

`any`

Returns

`ChangePassword`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / CoordinateMapping

Class: CoordinateMapping

Defined in: api/api-client.ts:1460

Stores geographic coordinates associated with postalcodes, including the time the mapping was used.

Implements

- `ICoordinateMapping`

Constructors Constructor

> **new CoordinateMapping**(`data?`): `CoordinateMapping`

Defined in: api/api-client.ts:1480

Parameters data?

`ICoordinateMapping`

Returns

`CoordinateMapping`

Properties lastUsed?

> `optional` **lastUsed**: `Date`

Defined in: api/api-client.ts:1475

Gets or sets the time the postalcode was last entered by the user.

Implementation of

`ICoordinateMapping`.`lastUsed`

---

latitude?

> `optional` **latitude**: `number`

Defined in: api/api-client.ts:1469

Gets or sets the latitude for the location.

Implementation of

`ICoordinateMapping`.`latitude`

---

location?

> `optional` **location**: `string`

Defined in: api/api-client.ts:1466

Gets or sets the name of the location.

Implementation of

`ICoordinateMapping`.`location`

---

lockedUntil?

> `optional` **lockedUntil**: `Date`

Defined in: api/api-client.ts:1478

Gets or sets the time until which the entry is locked.

Implementation of

`ICoordinateMapping`.`lockedUntil`

---

longitude?

> `optional` **longitude**: `number`

Defined in: api/api-client.ts:1472

Gets or sets the longitude of the location.

Implementation of

`ICoordinateMapping`.`longitude`

---

postalCode?

> `optional` **postalCode**: `number`

Defined in: api/api-client.ts:1463

Gets or sets the postalcode which is also the uniqe identifier.

Implementation of

`ICoordinateMapping`.`postalCode`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1489

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1507

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `CoordinateMapping`

Defined in: api/api-client.ts:1500

Parameters data

`any`

Returns

`CoordinateMapping`

September 2, 2025

isopruefi-frontend / api/api-client / IdentityUser

Class: IdentityUser

Defined in: api/api-client.ts:1733

The default implementation of IdentityUser`1 which uses a string as a primary key.

Extends

- `IdentityUserOfString`

Extended by

- `ApiUser`

Implements

- `IIdentityUser`

Constructors Constructor

> **new IdentityUser**(`data?`): `IdentityUser`

Defined in: api/api-client.ts:1735

Parameters data?

`IIdentityUser`

Returns

`IdentityUser`

Overrides

`IdentityUserOfString . constructor`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1639

Gets or sets the number of failed login attempts for the current user.

Implementation of

`IIdentityUser . accessFailedCount`

Inherited from

`IdentityUserOfString . accessFailedCount`

---

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1627

A random value that must change whenever a user is persisted to the store

Implementation of

`IIdentityUser.concurrencyStamp`

Inherited from

`IdentityUserOfString.concurrencyStamp`

---

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1617

Gets or sets the email address for this user.

Implementation of

`IIdentityUser.email`

Inherited from

`IdentityUserOfString.email`

---

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1621

Gets or sets a flag indicating if a user has confirmed their email address.

Implementation of

`IIdentityUser.emailConfirmed`

Inherited from

`IdentityUserOfString.emailConfirmed`

---

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1611

Gets or sets the primary key for this user.

Implementation of

`IIdentityUser.id`

Inherited from

`IdentityUserOfString.id`

---

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1637

Gets or sets a flag indicating if the user could be locked out.

Implementation of

`IIdentityUser.lockoutEnabled`

Inherited from

`IdentityUserOfString.lockoutEnabled`

---

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1635

Gets or sets the date and time, in UTC, when any user lockout ends.

Implementation of

`IIdentityUser.lockoutEnd`

Inherited from

`IdentityUserOfString.lockoutEnd`

---

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1619

Gets or sets the normalized email address for this user.

Implementation of

`IIdentityUser.normalizedEmail`

Inherited from

`IdentityUserOfString.normalizedEmail`

---

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1615

Gets or sets the normalized user name for this user.

Implementation of

`IIdentityUser.normalizedUserName`

Inherited from

`IdentityUserOfString.normalizedUserName`

---

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1623

Gets or sets a salted and hashed representation of the password for this user.

Implementation of

`IIdentityUser.passwordHash`

Inherited from

`IdentityUserOfString.passwordHash`

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1629

Gets or sets a telephone number for the user.

Implementation of

`IIdentityUser.phoneNumber`

Inherited from

`IdentityUserOfString.phoneNumber`

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1631

Gets or sets a flag indicating if a user has confirmed their telephone address.

Implementation of

`IIdentityUser.phoneNumberConfirmed`

Inherited from

`IdentityUserOfString.phoneNumberConfirmed`

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1625

A random value that must change whenever a users credentials change (password changed, login removed)

Implementation of

`IIdentityUser.securityStamp`

Inherited from

`IdentityUserOfString.securityStamp`

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1633

Gets or sets a flag indicating if two factor authentication is enabled for this user.

Implementation of

`IIdentityUser.twoFactorEnabled`

Inherited from

`IdentityUserOfString.twoFactorEnabled`

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1613

Gets or sets the user name for this user.

Implementation of

`IIdentityUser.userName`

Inherited from

`IdentityUserOfString.userName`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1739

Parameters _data?

`any`

Returns

`void`

Overrides

`IdentityUserOfString.init`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1750

Parameters data?

`any`

Returns

`any`

Overrides

`IdentityUserOfString.toJSON`

---

fromJS()

> `static` **fromJS**(`data`): `IdentityUser`

Defined in: api/api-client.ts:1743

Parameters data

`any`

Returns

`IdentityUser`

Overrides

`IdentityUserOfString`.`fromJS`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IdentityUserOfString

Class: IdentityUserOfString

Defined in: api/api-client.ts:1609

Represents a user in the identity system

Extended by

- `IdentityUser`

Implements

- `IIdentityUserOfString`

Constructors Constructor

> **new IdentityUserOfString**(`data?`): `IdentityUserOfString`

Defined in: api/api-client.ts:1641

Parameters data?

`IIdentityUserOfString`

Returns

`IdentityUserOfString`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1639

Gets or sets the number of failed login attempts for the current user.

Implementation of

`IIdentityUserOfString` . `accessFailedCount`

---

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1627

A random value that must change whenever a user is persisted to the store

Implementation of

`IIdentityUserOfString` . `concurrencyStamp`

---

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1617

Gets or sets the email address for this user.

Implementation of

`IIdentityUserOfString.email`

---

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1621

Gets or sets a flag indicating if a user has confirmed their email address.

Implementation of

`IIdentityUserOfString.emailConfirmed`

---

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1611

Gets or sets the primary key for this user.

Implementation of

`IIdentityUserOfString.id`

---

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1637

Gets or sets a flag indicating if the user could be locked out.

Implementation of

`IIdentityUserOfString.lockoutEnabled`

---

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1635

Gets or sets the date and time, in UTC, when any user lockout ends.

Implementation of

`IIdentityUserOfString.lockoutEnd`

---

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1619

Gets or sets the normalized email address for this user.

Implementation of

`IIdentityUserOfString.normalizedEmail`

---

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1615

Gets or sets the normalized user name for this user.

Implementation of

`IIdentityUserOfString.normalizedUserName`

---

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1623

Gets or sets a salted and hashed representation of the password for this user.

Implementation of

`IIdentityUserOfString.passwordHash`

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1629

Gets or sets a telephone number for the user.

Implementation of

`IIdentityUserOfString.phoneNumber`

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1631

Gets or sets a flag indicating if a user has confirmed their telephone address.

Implementation of

`IIdentityUserOfString.phoneNumberConfirmed`

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1625

A random value that must change whenever a users credentials change (password changed, login removed)

Implementation of

`IIdentityUserOfString.securityStamp`

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1633

Gets or sets a flag indicating if two factor authentication is enabled for this user.

Implementation of

`IIdentityUserOfString.twoFactorEnabled`

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1613

Gets or sets the user name for this user.

Implementation of

`IIdentityUserOfString.userName`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1650

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1677

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `IdentityUserOfString`

Defined in: api/api-client.ts:1670

Parameters data

`any`

Returns

`IdentityUserOfString`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / JwtToken

Class: JwtToken

Defined in: api/api-client.ts:1112

Represents a JWT token and its associated refresh token and metadata.

Implements

- `IJwtToken`

Constructors Constructor

> **new JwtToken**(`data?`): `JwtToken`

Defined in: api/api-client.ts:1129

Parameters data?

`IJwtToken`

Returns

`JwtToken`

Properties createdDate?

> `optional` **createdDate**: `Date`

Defined in: api/api-client.ts:1124

Gets or sets the creation date and time of the JWT token.

Implementation of

`IJwtToken`.`createdDate`

---

expiryDate?

> `optional` **expiryDate**: `Date`

Defined in: api/api-client.ts:1121

Gets or sets the expiry date and time of the JWT token.

Implementation of

`IJwtToken`.`expiryDate`

---

refreshToken?

> `optional` **refreshToken**: `string`

Defined in: api/api-client.ts:1118

Gets or sets the refresh token string.

Implementation of

`IJwtToken`.`refreshToken`

---

roles?

> `optional` **roles**: `string` []

Defined in: api/api-client.ts:1127

Gets or sets the user roles associated with the JWT token.

Implementation of

`IJwtToken` . `roles`

---

token?

> `optional` **token**: `string`

Defined in: api/api-client.ts:1115

Gets or sets the JWT access token string.

Implementation of

`IJwtToken` . `token`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1138

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1159

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `JwtToken`

Defined in: api/api-client.ts:1152

Parameters data

`any`

Returns

`JwtToken`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / LocationClient

Class: LocationClient

Defined in: api/api-client.ts:190

Constructors Constructor

> **new LocationClient**(`baseUrl?` , `http?`): `LocationClient`

Defined in: api/api-client.ts:195

Parameters baseUrl?

`string`

http? fetch Returns

`LocationClient`

Properties jsonParseReviver

> `protected` **jsonParseReviver**: `undefined` | (`key` , `value`) => `any` = `undefined`

Defined in: api/api-client.ts:193

Methods getAllPostalcodes()

> **getAllPostalcodes**(): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:204

Retrieves all saved locations.

Returns

`Promise` \< `FileResponse` >

A list of all postalcodes; otherwise, NotFound.

---

insertLocation()

> **insertLocation**(`postalcode?`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:247

Checks for existence of location and if necessary inserts new location.

Parameters postalcode?

`number`

(optional) Defines the location.

Returns

`Promise` \< `FileResponse` >

Ok if successful; otherwise, an error response.

---

processGetAllPostalcodes()

> `protected` **processGetAllPostalcodes**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:220

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

---

processInsertLocation()

> `protected` **processInsertLocation**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:267

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

---

processRemovePostalcode()

> `protected` **processRemovePostalcode**(`response`): `Promise` \< `void` >

Defined in: api/api-client.ts:308

Parameters response

`Response`

Returns

`Promise` \< `void` >

---

removePostalcode()

> **removePostalcode**(`postalCode?`): `Promise` \< `void` >

Defined in: api/api-client.ts:289

Parameters postalCode?

`number`

Returns

`Promise` \< `void` >

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / Login

Class: Login

Defined in: api/api-client.ts:948

Represents the login credentials for a user.

Implements

- `ILogin`

Constructors Constructor

> **new Login**(`data?`): `Login`

Defined in: api/api-client.ts:956

Parameters data?

`ILogin`

Returns

`Login`

Properties password

> **password**: `string`

Defined in: api/api-client.ts:954

Gets or sets the password of the user.

Implementation of

`ILogin`.`password`

---

userName

> **userName**: `string`

Defined in: api/api-client.ts:951

Gets or sets the username of the user.

Implementation of

`ILogin`.`userName`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:965

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:979

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `Login`

Defined in: api/api-client.ts:972

Parameters data

`any`

Returns

`Login`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ProblemDetails

Class: ProblemDetails

Defined in: api/api-client.ts:997

Implements

- `IProblemDetails`

Indexable

[`key` : `string`]: `any`

Constructors Constructor

> **new ProblemDetails**(`data?`): `ProblemDetails`

Defined in: api/api-client.ts:1006

Parameters data?

`IProblemDetails`

Returns

`ProblemDetails`

Properties detail?

> `optional` **detail**: `string`

Defined in: api/api-client.ts:1001

Implementation of

`IProblemDetails` . `detail`

---

instance?

> `optional` **instance**: `string`

Defined in: api/api-client.ts:1002

Implementation of

`IProblemDetails` . `instance`

---

status?

> `optional` **status**: `number`

Defined in: api/api-client.ts:1000

Implementation of

`IProblemDetails` . `status`

---

title?

> `optional` **title**: `string`

Defined in: api/api-client.ts:999

Implementation of

`IProblemDetails`.`title`

---

type?

> `optional` **type**: `string`

Defined in: api/api-client.ts:998

Implementation of

`IProblemDetails`.`type`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1015

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1036

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `ProblemDetails`

Defined in: api/api-client.ts:1029

Parameters data

`any`

Returns

`ProblemDetails`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / Register

Class: Register

Defined in: api/api-client.ts:1062

Represents the registration credentials for a new user.

Implements

- `IRegister`

Constructors Constructor

> **new Register**(`data?`): `Register`

Defined in: api/api-client.ts:1070

Parameters data?

`IRegister`

Returns

`Register`

Properties password

> **password**: `string`

Defined in: api/api-client.ts:1068

Gets or sets the password for the new user.

Implementation of

`IRegister` . `password`

---

userName

> **userName**: `string`

Defined in: api/api-client.ts:1065

Gets or sets the username for the new user.

Implementation of

`IRegister` . `userName`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1079

Parameters _data?

`any`

Returns

`void`

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1093

Parameters data?

`any`

Returns

`any`

fromJS()

> `static` **fromJS**(`data`): `Register`

Defined in: api/api-client.ts:1086

Parameters data

`any`

Returns

`Register`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / SensorData

Class: SensorData

Defined in: api/api-client.ts:1255

Implements

- `ISensorData`

Constructors Constructor

> **new SensorData**(`data?`): `SensorData`

Defined in: api/api-client.ts:1260

Parameters data?

`ISensorData`

Returns

`SensorData`

Properties location?

> `optional` **location**: `string`

Defined in: api/api-client.ts:1257

Implementation of

`ISensorData` . `location`

---

sensorName?

> `optional` **sensorName**: `string`

Defined in: api/api-client.ts:1256

Implementation of

`ISensorData` . `sensorName`

---

temperatureDatas?

> `optional` **temperatureDatas**: `TemperatureData` []

Defined in: api/api-client.ts:1258

Implementation of

`ISensorData` . `temperatureDatas`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1269

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1288

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `SensorData`

Defined in: api/api-client.ts:1281

Parameters data

`any`

Returns

`SensorData`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / TemperatureData

Class: TemperatureData

Defined in: api/api-client.ts:1308

Represents a single temperature data point with timestamp and value.

Implements

- `ITemperatureData`

Constructors Constructor

> **new TemperatureData**(`data?`): `TemperatureData`

Defined in: api/api-client.ts:1317

Parameters data?

`ITemperatureData`

Returns

`TemperatureData`

Properties plausibility?

> `optional` **plausibility**: `string`

Defined in: api/api-client.ts:1315

Implementation of

`ITemperatureData` . `plausibility`

---

temperature?

> `optional` **temperature**: `number`

Defined in: api/api-client.ts:1314

Gets or sets the temperature value.

Implementation of

`ITemperatureData` . `temperature`

---

timestamp?

> `optional` **timestamp**: `Date`

Defined in: api/api-client.ts:1311

Gets or sets the timestamp of the temperature measurement.

Implementation of

`ITemperatureData` . `timestamp`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1326

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1341

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `TemperatureData`

Defined in: api/api-client.ts:1334

Parameters data

`any`

Returns

`TemperatureData`

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/api-client / TemperatureDataClient

Class: TemperatureDataClient

Defined in: api/api-client.ts:324

Constructors Constructor

> **new TemperatureDataClient**(`baseUrl?` , `http?`): `TemperatureDataClient`

Defined in: api/api-client.ts:329

Parameters baseUrl?

`string`

http? fetch Returns

`TemperatureDataClient`

Properties jsonParseReviver

> `protected` **jsonParseReviver**: `undefined` | (`key` , `value`) => `any` = `undefined`

Defined in: api/api-client.ts:327

Methods getTemperature()

> **getTemperature**(`start?` , `end?` , `place?` , `isFahrenheit?`): `Promise` \< `TemperatureDataOverview` >

Defined in: api/api-client.ts:342

Retrieves comprehensive temperature data for a specified time range and location.

Parameters start?

`Date`

(optional) Start date and time for the data range (ISO 8601 format).

end?

`Date`

(optional) End date and time for the data range (ISO 8601 format).

place?

`string`

(optional) Location name for external weather data (e.g., "Berlin", "Munich").

isFahrenheit?

`boolean`

(optional) Optional. If true, converts all temperatures to Fahrenheit. Default is false (Celsius).

Returns

`Promise` \< `TemperatureDataOverview` >

Successfully retrieved temperature data. Returns comprehensive temperature overview.

processGetTemperature()

> `protected` **processGetTemperature**(`response`): `Promise` \< `TemperatureDataOverview` >

Defined in: api/api-client.ts:374

Parameters response

`Response`

Returns

`Promise` \< `TemperatureDataOverview` >

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / TemperatureDataOverview

Class: TemperatureDataOverview

Defined in: api/api-client.ts:1194

Represents an overview of temperature data for different locations.

Implements

- `ITemperatureDataOverview`

Constructors Constructor

> **new TemperatureDataOverview**(`data?`): `TemperatureDataOverview`

Defined in: api/api-client.ts:1200

Parameters data?

`ITemperatureDataOverview`

Returns

`TemperatureDataOverview`

Properties sensorData?

> `optional` **sensorData**: `SensorData` []

Defined in: api/api-client.ts:1195

Implementation of

`ITemperatureDataOverview` . `sensorData`

---

temperatureOutside?

> `optional` **temperatureOutside**: `TemperatureData` []

Defined in: api/api-client.ts:1198

Gets or sets the list of temperature data for the outside location.

Implementation of

`ITemperatureDataOverview` . `temperatureOutside`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1209

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1231

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `TemperatureDataOverview`

Defined in: api/api-client.ts:1224

Parameters data

`any`

Returns

`TemperatureDataOverview`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / TopicClient

Class: TopicClient

Defined in: api/api-client.ts:418

Constructors Constructor

> **new TopicClient**(`baseUrl?` , `http?`): `TopicClient`

Defined in: api/api-client.ts:423

Parameters baseUrl?

`string`

http? fetch Returns

`TopicClient`

Properties jsonParseReviver

> `protected` **jsonParseReviver**: `undefined` | (`key` , `value`) `=>` `any` `=` `undefined`

Defined in: api/api-client.ts:421

Methods createTopic()

> **createTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/api-client.ts:555

Creates a new MQTT topic configuration for sensor monitoring.

Parameters topicSetting

`TopicSetting`

The complete topic setting configuration to create.

Returns

`Promise` \< `any` >

Topic setting created successfully. Returns the new topic ID.

---

deleteTopic()

> **deleteTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/api-client.ts:683

Parameters topicSetting

`TopicSetting`

Returns

`Promise` \< `any` >

---

getAllSensorTypes()

> **getAllSensorTypes**(): `Promise` \< `string` []>

Defined in: api/api-client.ts:491

Returns

`Promise` \< `string` []>

---

getAllTopics()

> **getAllTopics**(): `Promise` \< `TopicSetting` []>

Defined in: api/api-client.ts:432

Retrieves all configured MQTT topic settings from the system.

Returns

`Promise` \< `TopicSetting` []>

Successfully retrieved all topic settings.

---

processCreateTopic()

> `protected` **processCreateTopic**(`response`): `Promise` \< `any` >

Defined in: api/api-client.ts:575

Parameters response

`Response`

Returns

`Promise` \< `any` >

---

processDeleteTopic()

> `protected` **processDeleteTopic**(`response`): `Promise` \< `any` >

Defined in: api/api-client.ts:703

Parameters response

`Response`

Returns

`Promise` \< `any` >

---

processGetAllSensorTypes()

> `protected` **processGetAllSensorTypes**(`response`): `Promise` \< `string` []>

Defined in: api/api-client.ts:507

Parameters response

`Response`

Returns

`Promise` \< `string` []>

processGetAllTopics()

> `protected` **processGetAllTopics**(`response`): `Promise` \< `TopicSetting` []>

Defined in: api/api-client.ts:448

Parameters response

`Response`

Returns

`Promise` \< `TopicSetting` []>

processUpdateTopic()

> `protected` **processUpdateTopic**(`response`): `Promise` \< `any` >

Defined in: api/api-client.ts:639

Parameters response

`Response`

Returns

`Promise` \< `any` >

updateTopic()

> **updateTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/api-client.ts:619

Parameters topicSetting

`TopicSetting`

Returns

`Promise` \< `any` >

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / TopicSetting

Class: TopicSetting

Defined in: api/api-client.ts:1362

Represents the settings for a specific MQTT topic, including default path, group, and sensor information.

Implements

- `ITopicSetting`

Constructors Constructor

> **new TopicSetting**(`data?`): `TopicSetting`

Defined in: api/api-client.ts:1387

Parameters data?

`ITopicSetting`

Returns

`TopicSetting`

Properties coordinateMapping?

> `optional` **coordinateMapping**: `CoordinateMapping`

Defined in: api/api-client.ts:1367

Implementation of

`ITopicSetting` . `coordinateMapping`

---

coordinateMappingId?

> `optional` **coordinateMappingId**: `number`

Defined in: api/api-client.ts:1366

Implementation of

`ITopicSetting` . `coordinateMappingId`

---

defaultTopicPath?

> `optional` **defaultTopicPath**: `string`

Defined in: api/api-client.ts:1370

Gets or sets the default MQTT topic path for this setting.

Implementation of

`ITopicSetting` . `defaultTopicPath`

---

groupId?

> `optional` **groupId**: `number`

Defined in: api/api-client.ts:1373

Gets or sets the group identifier associated with this topic setting.

Implementation of

`ITopicSetting.groupId`

hasRecovery?

> `optional` **hasRecovery**: `boolean`

Defined in: api/api-client.ts:1385

Gets or sets a value indicating whether this topic setting has recovery enabled.

Implementation of

`ITopicSetting.hasRecovery`

sensorLocation?

> `optional` **sensorLocation**: `string`

Defined in: api/api-client.ts:1382

Gets or sets the location of the sensor.

Implementation of

`ITopicSetting.sensorLocation`

sensorName?

> `optional` **sensorName**: `string`

Defined in: api/api-client.ts:1379

Gets or sets the name of the sensor.

Implementation of

`ITopicSetting.sensorName`

sensorTypeEnum?

> `optional` **sensorTypeEnum**: `SensorType`

Defined in: api/api-client.ts:1376

Gets or sets the type of sensor (e.g., temperature, humidity).

Implementation of

`ITopicSetting.sensorTypeEnum`

topicSettingId?

> `optional` **topicSettingId**: `number`

Defined in: api/api-client.ts:1365

Gets or sets the unique identifier for the TopicSetting entity.

Implementation of

`ITopicSetting`.`topicSettingId`

Methods init()

> **init**(`_data?`): `void`

Defined in: api/api-client.ts:1396

Parameters _data?

`any`

Returns

`void`

---

toJSON()

> **toJSON**(`data?`): `any`

Defined in: api/api-client.ts:1417

Parameters data?

`any`

Returns

`any`

---

fromJS()

> `static` **fromJS**(`data`): `TopicSetting`

Defined in: api/api-client.ts:1410

Parameters data

`any`

Returns

`TopicSetting`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / UserInfoClient

Class: UserInfoClient

Defined in: api/api-client.ts:748

Constructors Constructor

> **new UserInfoClient**(`baseUrl?` , `http?`): `UserInfoClient`

Defined in: api/api-client.ts:753

Parameters baseUrl?

`string`

http? fetch Returns

`UserInfoClient`

Properties jsonParseReviver

> `protected` **jsonParseReviver**: `undefined` | (`key` , `value`) => `any` = `undefined`

Defined in: api/api-client.ts:751

Methods changePassword()

> **changePassword**(`input`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:810

Changes the password for a user.

Parameters input

`ChangePassword`

The change password request containing user ID, current password, and new password.

Returns

`Promise` \< `FileResponse` >

Ok if successful; otherwise, an error response.

---

changeUser()

> **changeUser**(`user`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:857

Updates user information.

Parameters user

`ApiUser`

The user object with updated information.

Returns

`Promise` \< `FileResponse` >

Ok if successful; otherwise, an error response.

---

deleteUser()

> **deleteUser**(`userId?`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:904

Deletes a user by their unique identifier.

Parameters userId?

`string`

(optional) The unique identifier of the user to delete.

Returns

`Promise` \< `FileResponse` >

Ok if successful; otherwise, an error response.

---

getUserById()

> **getUserById**(`userId?`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:763

Retrieves a user by their unique identifier.

Parameters userId?

`string`

(optional) The unique identifier of the user.

Returns

`Promise` \< `FileResponse` >

The user information if found; otherwise, NotFound.

---

processChangePassword()

> `protected` **processChangePassword**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:830

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

---

processChangeUser()

> `protected` **processChangeUser**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:877

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

processDeleteUser()

> `protected` **processDeleteUser**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:924

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

processGetUserById()

> `protected` **processGetUserById**(`response`): `Promise` \< `FileResponse` >

Defined in: api/api-client.ts:783

Parameters response

`Response`

Returns

`Promise` \< `FileResponse` >

September 2, 2025

ENUMERATIONS

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / SensorType

Enumeration: SensorType

Defined in: api/api-client.ts:1541

Enumeration Members Co2

> **Co2**: `4`

Defined in: api/api-client.ts:1546

Hum

> **Hum**: `2`

Defined in: api/api-client.ts:1544

Ikea

> **Ikea**: `3`

Defined in: api/api-client.ts:1545

Mic

> **Mic**: `5`

Defined in: api/api-client.ts:1547

Spl

> **Spl**: `1`

Defined in: api/api-client.ts:1543

Temp

> **Temp**: `0`

Defined in: api/api-client.ts:1542

September 2, 2025

INTERFACES

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / FileResponse

Interface: FileResponse

Defined in: api/api-client.ts:1790

Properties data

> **data**: `Blob`

Defined in: api/api-client.ts:1791

---

fileName?

> `optional` **fileName**: `string`

Defined in: api/api-client.ts:1793

---

headers?

> `optional` **headers**: `object`

Defined in: api/api-client.ts:1794

Index Signature

[`name` : `string`]: `any`

---

status

> **status**: `number`

Defined in: api/api-client.ts:1792

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IApiUser

Interface: IApiUser

Defined in: api/api-client.ts:1787

Represents an application user in the system

Extends

- `IIdentityUser`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1729

Gets or sets the number of failed login attempts for the current user.

Inherited from

`IIdentityUser`.`accessFailedCount`

---

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1717

A random value that must change whenever a user is persisted to the store

Inherited from

`IIdentityUser`.`concurrencyStamp`

---

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1707

Gets or sets the email address for this user.

Inherited from

`IIdentityUser`.`email`

---

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1711

Gets or sets a flag indicating if a user has confirmed their email address.

Inherited from

`IIdentityUser`.`emailConfirmed`

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1701

Gets or sets the primary key for this user.

Inherited from

`IIdentityUser`.`id`

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1727

Gets or sets a flag indicating if the user could be locked out.

Inherited from

`IIdentityUser`.`lockoutEnabled`

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1725

Gets or sets the date and time, in UTC, when any user lockout ends.

Inherited from

`IIdentityUser`.`lockoutEnd`

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1709

Gets or sets the normalized email address for this user.

Inherited from

`IIdentityUser`.`normalizedEmail`

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1705

Gets or sets the normalized user name for this user.

Inherited from

`IIdentityUser`.`normalizedUserName`

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1713

Gets or sets a salted and hashed representation of the password for this user.

Inherited from

`IIdentityUser`.`passwordHash`

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1719

Gets or sets a telephone number for the user.

Inherited from

`IIdentityUser`.`phoneNumber`

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1721

Gets or sets a flag indicating if a user has confirmed their telephone address.

Inherited from

`IIdentityUser`.`phoneNumberConfirmed`

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1715

A random value that must change whenever a users credentials change (password changed, login removed)

Inherited from

`IIdentityUser`.`securityStamp`

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1723

Gets or sets a flag indicating if two factor authentication is enabled for this user.

Inherited from

`IIdentityUser`.`twoFactorEnabled`

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1703

Gets or sets the user name for this user.

Inherited from

`IIdentityUser`.`userName`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IChangePassword

Interface: IChangePassword

Defined in: api/api-client.ts:1596

Represents a request to change a user's password.

Properties currentPassword?

> `optional` **currentPassword**: `string`

Defined in: api/api-client.ts:1602

Gets or sets the current password of the user.

---

newPassword?

> `optional` **newPassword**: `string`

Defined in: api/api-client.ts:1605

Gets or sets the new password to be set for the user.

---

userId?

> `optional` **userId**: `string`

Defined in: api/api-client.ts:1599

Gets or sets the unique identifier of the user whose password is to be changed.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ICoordinateMapping

Interface: ICoordinateMapping

Defined in: api/api-client.ts:1520

Stores geographic coordinates associated with postalcodes, including the time the mapping was used.

Properties lastUsed?

> `optional` **lastUsed**: `Date`

Defined in: api/api-client.ts:1535

Gets or sets the time the postalcode was last entered by the user.

---

latitude?

> `optional` **latitude**: `number`

Defined in: api/api-client.ts:1529

Gets or sets the latitude for the location.

---

location?

> `optional` **location**: `string`

Defined in: api/api-client.ts:1526

Gets or sets the name of the location.

---

lockedUntil?

> `optional` **lockedUntil**: `Date`

Defined in: api/api-client.ts:1538

Gets or sets the time until which the entry is locked.

---

longitude?

> `optional` **longitude**: `number`

Defined in: api/api-client.ts:1532

Gets or sets the longitude of the location.

---

postalCode?

> `optional` **postalCode**: `number`

Defined in: api/api-client.ts:1523

Gets or sets the postalcode which is also the uniqe identifier.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/api-client / IIdentityUser

Interface: IIdentityUser

Defined in: api/api-client.ts:1758

The default implementation of IdentityUser`1 which uses a string as a primary key.

Extends

- `IIdentityUserOfString`

Extended by

- `IApiUser`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1729

Gets or sets the number of failed login attempts for the current user.

Inherited from

`IIdentityUserOfString`.`accessFailedCount`

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1717

A random value that must change whenever a user is persisted to the store

Inherited from

`IIdentityUserOfString`.`concurrencyStamp`

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1707

Gets or sets the email address for this user.

Inherited from

`IIdentityUserOfString`.`email`

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1711

Gets or sets a flag indicating if a user has confirmed their email address.

Inherited from

`IIdentityUserOfString.emailConfirmed`

---

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1701

Gets or sets the primary key for this user.

Inherited from

`IIdentityUserOfString.id`

---

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1727

Gets or sets a flag indicating if the user could be locked out.

Inherited from

`IIdentityUserOfString.lockoutEnabled`

---

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1725

Gets or sets the date and time, in UTC, when any user lockout ends.

Inherited from

`IIdentityUserOfString.lockoutEnd`

---

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1709

Gets or sets the normalized email address for this user.

Inherited from

`IIdentityUserOfString.normalizedEmail`

---

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1705

Gets or sets the normalized user name for this user.

Inherited from

`IIdentityUserOfString.normalizedUserName`

---

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1713

Gets or sets a salted and hashed representation of the password for this user.

Inherited from

`IIdentityUserOfString.passwordHash`

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1719

Gets or sets a telephone number for the user.

Inherited from

`IIdentityUserOfString.phoneNumber`

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1721

Gets or sets a flag indicating if a user has confirmed their telephone address.

Inherited from

`IIdentityUserOfString.phoneNumberConfirmed`

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1715

A random value that must change whenever a users credentials change (password changed, login removed)

Inherited from

`IIdentityUserOfString.securityStamp`

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1723

Gets or sets a flag indicating if two factor authentication is enabled for this user.

Inherited from

`IIdentityUserOfString.twoFactorEnabled`

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1703

Gets or sets the user name for this user.

Inherited from

`IIdentityUserOfString.userName`

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IIdentityUserOfString

Interface: IIdentityUserOfString

Defined in: api/api-client.ts:1699

Represents a user in the identity system

Extended by

- `IIdentityUser`

Properties accessFailedCount?

> `optional` **accessFailedCount**: `number`

Defined in: api/api-client.ts:1729

Gets or sets the number of failed login attempts for the current user.

---

concurrencyStamp?

> `optional` **concurrencyStamp**: `string`

Defined in: api/api-client.ts:1717

A random value that must change whenever a user is persisted to the store

---

email?

> `optional` **email**: `string`

Defined in: api/api-client.ts:1707

Gets or sets the email address for this user.

---

emailConfirmed?

> `optional` **emailConfirmed**: `boolean`

Defined in: api/api-client.ts:1711

Gets or sets a flag indicating if a user has confirmed their email address.

---

id?

> `optional` **id**: `string`

Defined in: api/api-client.ts:1701

Gets or sets the primary key for this user.

---

lockoutEnabled?

> `optional` **lockoutEnabled**: `boolean`

Defined in: api/api-client.ts:1727

Gets or sets a flag indicating if the user could be locked out.

---

lockoutEnd?

> `optional` **lockoutEnd**: `Date`

Defined in: api/api-client.ts:1725

Gets or sets the date and time, in UTC, when any user lockout ends.

---

normalizedEmail?

> `optional` **normalizedEmail**: `string`

Defined in: api/api-client.ts:1709

Gets or sets the normalized email address for this user.

---

normalizedUserName?

> `optional` **normalizedUserName**: `string`

Defined in: api/api-client.ts:1705

Gets or sets the normalized user name for this user.

---

passwordHash?

> `optional` **passwordHash**: `string`

Defined in: api/api-client.ts:1713

Gets or sets a salted and hashed representation of the password for this user.

---

phoneNumber?

> `optional` **phoneNumber**: `string`

Defined in: api/api-client.ts:1719

Gets or sets a telephone number for the user.

---

phoneNumberConfirmed?

> `optional` **phoneNumberConfirmed**: `boolean`

Defined in: api/api-client.ts:1721

Gets or sets a flag indicating if a user has confirmed their telephone address.

---

securityStamp?

> `optional` **securityStamp**: `string`

Defined in: api/api-client.ts:1715

A random value that must change whenever a users credentials change (password changed, login removed)

---

twoFactorEnabled?

> `optional` **twoFactorEnabled**: `boolean`

Defined in: api/api-client.ts:1723

Gets or sets a flag indicating if two factor authentication is enabled for this user.

---

userName?

> `optional` **userName**: `string`

Defined in: api/api-client.ts:1703

Gets or sets the user name for this user.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IJwtToken

Interface: IJwtToken

Defined in: api/api-client.ts:1175

Represents a JWT token and its associated refresh token and metadata.

Properties

createdDate?

> `optional` **createdDate**: `Date`

Defined in: api/api-client.ts:1187

Gets or sets the creation date and time of the JWT token.

---

expiryDate?

> `optional` **expiryDate**: `Date`

Defined in: api/api-client.ts:1184

Gets or sets the expiry date and time of the JWT token.

---

refreshToken?

> `optional` **refreshToken**: `string`

Defined in: api/api-client.ts:1181

Gets or sets the refresh token string.

---

roles?

> `optional` **roles**: `string`[]

Defined in: api/api-client.ts:1190

Gets or sets the user roles associated with the JWT token.

---

token?

> `optional` **token**: `string`

Defined in: api/api-client.ts:1178

Gets or sets the JWT access token string.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ILogin

Interface: ILogin

Defined in: api/api-client.ts:988

Represents the login credentials for a user.

Properties password

> **password**: `string`

Defined in: api/api-client.ts:994

Gets or sets the password of the user.

---

userName

> **userName**: `string`

Defined in: api/api-client.ts:991

Gets or sets the username of the user.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IProblemDetails

Interface: IProblemDetails

Defined in: api/api-client.ts:1051

Indexable

[`key` : `string`]: `any`

Properties detail?

> `optional` **detail**: `string`

Defined in: api/api-client.ts:1055

---

instance?

> `optional` **instance**: `string`

Defined in: api/api-client.ts:1056

---

status?

> `optional` **status**: `number`

Defined in: api/api-client.ts:1054

---

title?

> `optional` **title**: `string`

Defined in: api/api-client.ts:1053

---

type?

> `optional` **type**: `string`

Defined in: api/api-client.ts:1052

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / IRegister

Interface: IRegister

Defined in: api/api-client.ts:1102

Represents the registration credentials for a new user.

Properties password

> **password**: `string`

Defined in: api/api-client.ts:1108

Gets or sets the password for the new user.

---

userName

> **userName**: `string`

Defined in: api/api-client.ts:1105

Gets or sets the username for the new user.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/api-client / ISensorData

Interface: ISensorData

Defined in: api/api-client.ts:1301

Properties location?

> `optional` **location**: `string`

Defined in: api/api-client.ts:1303

sensorName?

> `optional` **sensorName**: `string`

Defined in: api/api-client.ts:1302

temperatureDatas?

> `optional` **temperatureDatas**: `TemperatureData`[]

Defined in: api/api-client.ts:1304

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ITemperatureData

Interface: ITemperatureData

Defined in: api/api-client.ts:1351

Represents a single temperature data point with timestamp and value.

Properties plausibility?

> `optional` **plausibility**: `string`

Defined in: api/api-client.ts:1358

---

temperature?

> `optional` **temperature**: `number`

Defined in: api/api-client.ts:1357

Gets or sets the temperature value.

---

timestamp?

> `optional` **timestamp**: `Date`

Defined in: api/api-client.ts:1354

Gets or sets the timestamp of the temperature measurement.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/api-client / ITemperatureDataOverview

Interface: ITemperatureDataOverview

Defined in: api/api-client.ts:1248

Represents an overview of temperature data for different locations.

Properties sensorData?

> `optional` **sensorData**: `SensorData` []

Defined in: api/api-client.ts:1249

---

temperatureOutside?

> `optional` **temperatureOutside**: `TemperatureData` []

Defined in: api/api-client.ts:1252

Gets or sets the list of temperature data for the outside location.

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/api-client / ITopicSetting

Interface: ITopicSetting

Defined in: api/api-client.ts:1433

Represents the settings for a specific MQTT topic, including default path, group, and sensor information.

Properties coordinateMapping?

> `optional` **coordinateMapping**: `CoordinateMapping`

Defined in: api/api-client.ts:1438

---

coordinateMappingId?

> `optional` **coordinateMappingId**: `number`

Defined in: api/api-client.ts:1437

---

defaultTopicPath?

> `optional` **defaultTopicPath**: `string`

Defined in: api/api-client.ts:1441

Gets or sets the default MQTT topic path for this setting.

---

groupId?

> `optional` **groupId**: `number`

Defined in: api/api-client.ts:1444

Gets or sets the group identifier associated with this topic setting.

---

hasRecovery?

> `optional` **hasRecovery**: `boolean`

Defined in: api/api-client.ts:1456

Gets or sets a value indicating whether this topic setting has recovery enabled.

---

sensorLocation?

> `optional` **sensorLocation**: `string`

Defined in: api/api-client.ts:1453

Gets or sets the location of the sensor.

---

sensorName?

> `optional` **sensorName**: `string`

Defined in: api/api-client.ts:1450

Gets or sets the name of the sensor.

---

sensorTypeEnum?

> `optional` **sensorTypeEnum**: `SensorType`

Defined in: api/api-client.ts:1447

Gets or sets the type of sensor (e.g., temperature, humidity).

---

topicSettingId?

> `optional` **topicSettingId**: `number`

Defined in: api/api-client.ts:1436

Gets or sets the unique identifier for the TopicSetting entity.

September 2, 2025

clients

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients

API/CLIENTS

Type Aliases

- PostalLocation

Variables

- authClient
- locationClient
- tempClient
- topicClient

Functions

- addPostalLocation
- createTopic
- deleteTopic
- fetchPostalLocations
- getAllTopics
- getStoredLocationName
- removePostalLocation
- updateTopic

References

ApiException

Re-exports ApiException

TopicSetting

Re-exports TopicSetting

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / addPostalLocation

Function: addPostalLocation()

> **addPostalLocation**(`postalCode`): `Promise` \< `FileResponse` >

Defined in: api/clients.ts:155

Adds a new postal code location to the system.

Parameters postalCode

`number`

The postal code to add

Returns

`Promise` \< `FileResponse` >

Promise resolving to the API response

Throws

When the request fails or postal code already exists

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/clients / createTopic

Function: createTopic()

> **createTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/clients.ts:187

Creates a new MQTT topic configuration in the system.

Parameters topicSetting

`TopicSetting`

The complete topic setting configuration to create

Returns

`Promise` \< `any` >

Promise resolving to the newly created topic with assigned ID

Throws

When validation fails, topic already exists, or access is denied

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / deleteTopic

Function: deleteTopic()

> **deleteTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/clients.ts:209

Removes an MQTT topic configuration from the system.

Parameters topicSetting

`TopicSetting`

The topic setting to delete (requires topicSettingId)

Returns

`Promise` \< `any` >

Promise resolving to void on successful deletion

Throws

When topic doesn't exist or access is denied

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / fetchPostalLocations

Function: fetchPostalLocations()

> **fetchPostalLocations**(): `Promise`\< `object` []>

Defined in: api/clients.ts:71

Fetches all postal code locations from the API and normalizes the response format.
Handles multiple possible response formats from the backend API.

Returns

`Promise`\< `object` []>

Promise resolving to an array of PostalLocation objects

Throws

When API request fails

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / getAllTopics

Function: getAllTopics()

> **getAllTopics**(): `Promise`\< `TopicSetting` []>

Defined in: api/clients.ts:176

Retrieves all MQTT topic settings from the system.

Returns

`Promise`\< `TopicSetting` []>

Promise resolving to an array of TopicSetting objects

Throws

When the request fails or access is denied

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / getStoredLocationName

Function: getStoredLocationName()

> **getStoredLocationName**(`displayLocationName`): `string`

Defined in: api/clients.ts:144

Retrieves the backend-stored location name for a display location name.
Used internally to map user-friendly display names to backend identifiers.

Parameters displayLocationName

`string`

The display name shown to users

Returns

`string`

The corresponding stored location name for API calls

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / removePostalLocation

Function: removePostalLocation()

> **removePostalLocation**(`postalCode`): `Promise` \< `void` >

Defined in: api/clients.ts:166

Removes a postal code location from the system.

Parameters postalCode

`number`

The postal code to remove

Returns

`Promise` \< `void` >

Promise resolving to void on successful removal

Throws

When the request fails or postal code doesn't exist

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / updateTopic

Function: updateTopic()

> **updateTopic**(`topicSetting`): `Promise` \< `any` >

Defined in: api/clients.ts:198

Updates an existing MQTT topic configuration.

Parameters topicSetting

`TopicSetting`

The topic setting with updated values (must include topicSettingId)

Returns

`Promise` \< `any` >

Promise resolving to the updated topic setting

Throws

When validation fails, topic doesn't exist, or access is denied

September 2, 2025

TYPE-ALIASES

isopruefi-frontend v1.0.0

---

isopruefi-frontend / api/clients / PostalLocation

Type Alias: PostalLocation

> **PostalLocation** = `object`

Defined in: api/clients.ts:217

Represents a postal code location with its associated name.
Used for location-based temperature data queries.

Properties locationName

> **locationName**: `string`

Defined in: api/clients.ts:221

The human-readable location name

---

postalCode

> **postalCode**: `number`

Defined in: api/clients.ts:219

The postal code number

September 2, 2025

VARIABLES

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / authClient

Variable: authClient

> `const` **authClient**: `AuthenticationClient`

Defined in: api/clients.ts:42

Pre-configured authentication client with automatic JWT token handling.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / locationClient

Variable: locationClient

> `const` **locationClient**: `LocationClient`

Defined in: api/clients.ts:52

Pre-configured location client with automatic JWT token handling.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / tempClient

Variable: tempClient

> `const` **tempClient**: `TemperatureDataClient`

Defined in: api/clients.ts:47

Pre-configured temperature data client with automatic JWT token handling.

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / api/clients / topicClient

Variable: topicClient

> `const` **topicClient**: `TopicClient`

Defined in: api/clients.ts:57

Pre-configured MQTT topic client with automatic JWT token handling.

September 2, 2025

4.8.3 Authentication

AuthForm

isopruefi-frontend v1.0.0

---

isopruefi-frontend / auth/AuthForm

AUTH/AUTHFORM

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / auth/AuthForm / default

Function: default()

> **default**(`props`): `Element`

Defined in: auth/AuthForm.tsx:31

Authentication form component for sign in and sign up.
Handles user input, authentication API calls, error display, and navigation.
On successful login, sets the global authentication state and navigates to the appropriate page.
On registration, shows a success message and navigates to the sign in page.

Parameters props

`AuthFormProps`

Component props.

Returns

`Element`

The rendered authentication form.

September 2, 2025

SignIn

isopruefi-frontend v1.0.0

---

isopruefi-frontend / auth/SignIn

AUTH/SIGNIN

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

---

isopruefi-frontend / auth/SignIn / default

Function: default()

> **default**(): `Element`

Defined in: auth/SignIn.tsx:3

Returns

`Element`

September 2, 2025

SignUp

isopruefi-frontend v1.0.0

---

isopruefi-frontend / auth/SignUp

AUTH/SIGNUP

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / auth/SignUp / default

Function: default()

> **default**(): `Element`

Defined in: auth/SignUp.tsx:3

Returns

`Element`

September 2, 2025

4.8.4 Components

Navbar

isopruefi-frontend v1.0.0

isopruefi-frontend / components/Navbar

COMPONENTS/NAVBAR

September 2, 2025

ProtectedRoute

isopruefi-frontend v1.0.0

---

isopruefi-frontend / components/ProtectedRoute

COMPONENTS/PROTECTEDROUTE

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / components/ProtectedRoute / default

Function: default()

> **default**(`props`): `Element`

Defined in: components/ProtectedRoute.tsx:15

ProtectedRoute component for role-based route protection.
Checks authentication and user role before rendering child routes.
- If authentication is not ready, shows a loading indicator.
- If user is not authenticated, redirects to the public welcome page.
- If user role is not allowed, redirects to their default page.

Parameters props

Component props.

allowed

(`"admin"` | `"user"`)[]

Array of allowed roles for the route.

Returns

`Element`

The rendered protected route or a redirect.

September 2, 2025

Weather

isopruefi-frontend v1.0.0

---

isopruefi-frontend / components/Weather

COMPONENTS/WEATHER

Type Aliases

- WeatherEntry

Functions

- TempChart

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

---

isopruefi-frontend / components/Weather / TempChart

Function: TempChart()

> **TempChart**(`props`): `Element`

Defined in: components/Weather.tsx:36

Displays a temperature chart and sensor tiles for a given location.
Fetches weather and sensor data from the backend, supports filtering by time range,
and allows switching between Celsius and Fahrenheit.

Parameters props

`TempChartProps` = `{}`

Component props.

Returns

`Element`

The rendered temperature chart and sensor tiles.

September 2, 2025

TYPE-ALIASES

isopruefi-frontend v1.0.0

---

isopruefi-frontend / components/Weather / WeatherEntry

Type Alias: WeatherEntry

> **WeatherEntry** = `object`

Defined in: components/Weather.tsx:13

Represents a single weather data entry for a specific timestamp.

Indexable

[`key` : `string`]: `undefined` | `string` | `number`

Properties t

> **t**: `number`

Defined in: components/Weather.tsx:15

Epoch time in milliseconds.

---

tempOutside?

> `optional` **tempOutside**: `number`

Defined in: components/Weather.tsx:17

Outside temperature value.

---

timestamp

> **timestamp**: `string`

Defined in: components/Weather.tsx:14

ISO formatted timestamp of the data point.

September 2, 2025

4.8.5 Pages

AdminPage

isopruefi-frontend v1.0.0

isopruefi-frontend / pages/AdminPage

PAGES/ADMINPAGE

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / pages/AdminPage / default

Function: default()

> **default**(): `Element`

Defined in: pages/AdminPage.tsx:18

AdminPage component for managing locations, topics, and viewing weather data.
Provides controls for selecting location and temperature units, displays a weather chart,
and includes management sections for locations and topics.
Also provides a logout button to clear authentication and redirect to sign-in.

Returns

`Element`

The rendered admin page.

September 2, 2025

UserPage

isopruefi-frontend v1.0.0

isopruefi-frontend / pages/UserPage

PAGES/USERPAGE

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

---

isopruefi-frontend / pages/UserPage / default

Function: default()

> **default**(): `Element`

Defined in: pages/UserPage.tsx:15

UserPage component for viewing weather data and managing user preferences.
Provides controls for selecting location and temperature units, displays a weather chart,
and includes a logout button to clear authentication and redirect to the welcome page.

Returns

`Element`

The rendered user page.

September 2, 2025

Welcome

isopruefi-frontend v1.0.0

---

isopruefi-frontend / pages/Welcome

PAGES/WELCOME

Functions

- default

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / pages/Welcome / default

Function: default()

> **default**(): `Element`

Defined in: pages/Welcome.tsx:27

Welcome page component that serves as the application's landing screen.

Features:
- Split-screen layout with logo and navigation
- Branded design with IsoPrüfi styling
- Navigation links to authentication pages
- Responsive design with centered content
- Consistent color scheme and typography

Returns

`Element`

JSX element containing the welcome page layout

Example

```
// Used in routing configuration
<Route path="/" element={<Welcome />} />
```

September 2, 2025

4.8.6 Utils

authApi

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/authApi

UTILS/AUTHAPI

Type Aliases

- LoginResult

Functions

- login
- refreshToken
- register

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/authApi / login

Function: login()

> **login**(`userName` , `password`): `Promise` \< `LoginResult` >

Defined in: utils/authApi.ts:69

Authenticates a user with username and password credentials.

Parameters userName

`string`

The user's login username

password

`string`

The user's password

Returns

`Promise` \< `LoginResult` >

Promise resolving to login tokens

Throws

When credentials are invalid or server error occurs

Example

```
try {
  const result = await login('user@example.com', 'password123');
  saveToken(result.token, result.refreshToken);
} catch (error) {
  console.error('Login failed:', error);
}
```

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/authApi / refreshToken

Function: refreshToken()

> **refreshToken**(`token` , `refreshToken`): `Promise` \< `LoginResult` >

Defined in: utils/authApi.ts:120

Refreshes an expired access token using a valid refresh token.

Parameters token

`string`

The expired JWT access token

refreshToken

`string`

The valid refresh token

Returns

`Promise` \< `LoginResult` >

Promise resolving to new authentication tokens

Throws

When refresh token is invalid, expired, or revoked

Example

```
try {
  const tokens = await refreshToken(oldToken, refreshToken);
  saveToken(tokens.token, tokens.refreshToken);
} catch (error) {
  // Refresh failed, redirect to login
  clearToken();
  window.location.href = '/login';
}
```

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/authApi / register

Function: register()

> **register**(`userName` , `password`): `Promise` \< `void` >

Defined in: utils/authApi.ts:92

Registers a new user in the system. Requires admin privileges.

Parameters userName

`string`

The desired username for the new user

password

`string`

The password for the new user

Returns

`Promise` \< `void` >

Promise that resolves on successful registration

Throws

When registration fails, username exists, or insufficient permissions

Example

```
try {
  await register('newuser@example.com', 'securePassword123');
  console.log('User registered successfully');
} catch (error) {
  console.error('Registration failed:', error);
}
```

September 2, 2025

TYPE-ALIASES

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/authApi / LoginResult

Type Alias: LoginResult

> **LoginResult** = `object`

Defined in: utils/authApi.ts:12

Represents the result of a successful login operation.

Properties refreshToken

> **refreshToken**: `string`

Defined in: utils/authApi.ts:16

Refresh token for obtaining new access tokens

---

token

> **token**: `string`

Defined in: utils/authApi.ts:14

JWT access token for authenticated requests

September 2, 2025

config

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/config

UTILS/CONFIG

Functions

- apiBase

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/config / apiBase

Function: apiBase()

> **apiBase**(): `string`

Defined in: utils/config.ts:22

Resolves the API base URL from runtime configuration or environment variables.

Priority order:
1. Runtime configuration from `window.__APP_CONFIG__.API_BASE_URL`
2. Build-time environment variable `VITE_API_BASE_URL`
3. Empty string as fallback

Returns

`string`

The API base URL with trailing slashes removed

Example

```
// Returns "https://api.example.com" (trailing slash removed)
const baseUrl = apiBase();
```

September 2, 2025

tokenHelpers

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/tokenHelpers

UTILS/TOKENHELPERS

Interfaces

- JwtPayload

Functions

- clearToken
- decodeToken
- getRefreshToken
- getToken
- getUserFromToken
- saveToken

September 2, 2025

FUNCTIONS

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/tokenHelpers / clearToken

Function: clearToken()

> **clearToken**(): `void`

Defined in: utils/tokenHelpers.ts:84

Removes both access and refresh tokens from localStorage.
Call this function when logging out or when tokens become invalid.

Returns

`void`

Example

```
// On logout
clearToken();
// Redirect to login page
```

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/tokenHelpers / decodeToken

Function: decodeToken()

> **decodeToken**(`token`): `null` | `JwtPayload`

Defined in: utils/tokenHelpers.ts:104

Decodes a JWT token and extracts the payload containing user information.
Does not verify the token signature - use only for reading claims.

Parameters token

`string`

The JWT token to decode

Returns

`null` | `JwtPayload`

The decoded payload object, or null if decoding fails

Example

```
const payload = decodeToken(accessToken);
if (payload) {
  console.log('Token expires at:', new Date(payload.exp * 1000));
}
```

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/tokenHelpers / getRefreshToken

Function: getRefreshToken()

> **getRefreshToken**(): `null` | `string`

Defined in: utils/tokenHelpers.ts:69

Retrieves the stored refresh token from localStorage.

Returns

`null` | `string`

The refresh token string, or null if not found

Example

```
const refreshToken = getRefreshToken();
if (refreshToken) {
  // Use to obtain new access token
}
```

September 2, 2025

isopruefi-frontend v1.0.0

---

isopruefi-frontend / utils/tokenHelpers / getToken

Function: getToken()

> **getToken**(): `null` | `string`

Defined in: utils/tokenHelpers.ts:52

Retrieves the stored JWT access token from localStorage.

Returns

`null` | `string`

The access token string, or null if not found

Example

```
const token = getToken();
if (token) {
  // Use token for authenticated requests
}
```

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/tokenHelpers / getUserFromToken

Function: getUserFromToken()

> **getUserFromToken**(`token`): `null` | `string`

Defined in: utils/tokenHelpers.ts:128

Extracts the user identifier from a JWT token.
The subject field typically contains the username or user ID.

Parameters token

`string`

The JWT token to extract user information from

Returns

`null` | `string`

The user identifier string, or null if extraction fails

Example

```
const currentUser = getUserFromToken(getToken());
if (currentUser) {
  console.log('Current user:', currentUser);
}
```

September 2, 2025

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/tokenHelpers / saveToken

Function: saveToken()

> **saveToken**(`token` , `refreshToken`): `void`

Defined in: utils/tokenHelpers.ts:34

Saves JWT tokens to browser localStorage for persistent authentication.

Parameters token

`string`

The JWT access token

refreshToken

`string`

The refresh token for obtaining new access tokens

Returns

`void`

Example

```
// After successful login
saveToken(response.accessToken, response.refreshToken);
```

September 2, 2025

INTERFACES

isopruefi-frontend v1.0.0

isopruefi-frontend / utils/tokenHelpers / JwtPayload

Interface: JwtPayload

Defined in: utils/tokenHelpers.ts:10

Standard JWT payload structure with common claims.
Extends to allow additional custom claims from the authentication system.

Indexable

[`key` : `string`]: `unknown`

Allow additional custom claims

Properties exp?

> `optional` **exp**: `number`

Defined in: utils/tokenHelpers.ts:14

Expiration time (Unix timestamp)

iat?

> `optional` **iat**: `number`

Defined in: utils/tokenHelpers.ts:16

Issued at time (Unix timestamp)

sub?

> `optional` **sub**: `string`

Defined in: utils/tokenHelpers.ts:12

Subject (usually user ID or username)

September 2, 2025

5. Docker

5.1 Documentation of the Docker development environment

This documentation provides an overview of the Docker containers used, as well as their function and their addresses.

5.1.1 Overview of the Docker containers

Container	Image	Description	Adress
traefik	`traefik:3.4.4`	Reverse proxy and load balancer for external access to our containers (HTTPS certificates)	traefik.localhost, Ports: `80` , `443` , Dashboard-Port: `8432`
influxdb	`influxdb:3.2.1-core`	Time series database (InfluxDB 3.x) for data storage	Port: `8181`
influxdb-explorer	`influxdata/influxdb3-ui:1.0.3`	Web interface for managing and querying InfluxDB data	explorer.localhost, Port: `8888`
postgres	`postgres:alpine3.21`	PostgreSQL database for relational data storage	Port: `5432`
loki	`grafana/loki:3.5.2`	Log aggregation and management	Port: `3100`
prometheus	`prom/prometheus:v3.4.2`	Monitoring tool for collecting and evaluating metrics	Port: `9090`
alloy	`grafana/alloy:v1.9.2`	Observability platform for the integration of Loki and Prometheus	Ports: `12345` , `4317` , `4318`
grafana	`grafana/grafana:12.0.2`	Web-based visualization and dashboard for metrics and logs	grafana.localhost
isopruefi-frontend	own Build (React)	Frontend Application	frontend.localhost
isopruefi-backend-api	own Build (.NET REST-API)	Backend REST API for application logic	backend.localhost

5.1.2 Networks

The following Docker networks are used to logically separate the containers from each other:

- `isopruefi-network` : General network, used by Traefik
- `database-network` : Network for databases (InfluxDB, PostgreSQL)
- `isopruefi-custom` : Network for user-defined services (frontend, backend)
- `loki` : Network for observability tools (Loki, Grafana, Alloy)

5.1.3 Details of important containers

Traefik

Traefik serves as a reverse proxy that receives all HTTP(S) requests and forwards them to the appropriate Docker containers. It automatically manages the TLS certificates and provides a dashboard for administration.

Grafana

Grafana is used to visualize and analyze logs and metrics. It is connected to Loki (logs) and Prometheus (metrics).

InfluxDB und InfluxDB-Explorer

InfluxDB stores time series data, while InfluxDB Explorer provides a convenient web interface to access this data.

PostgreSQL

PostgreSQL stores relational data used by the backend API.

August 31, 2025

DianaTin23, deadmade, maratin23

6. IsoPrüfi Documentation

6.1 Introduction and Goals

6.1.1 Aim of our project IsoPrüfi:

Our project aims to test the effectiveness of building insulation based on outside temperature and present the data clearly using diagrams.

6.1.2 Features

Must-Have

- A website for a user-friendly presentation of temperature comparison diagrams
- Reliable sensors that measure interior temperature
- The ability to retrieve outside temperature data
- Clusterization of containers that we create ourselves

Should-Have

- Sensors should be capable of storing temperature data for a period of one day, even in the absence of an internet connection or synchronization with the server
- A website should be used to offer configuration options

Could-Have

- Database clustering

Won't Have

- The containers will only run on one server, however they are designed to function independently of each other
- Since this is a software project, we won't implement any resilience on the hardware side

6.1.3 Requirements Overview

Functional Requirements

- The system must provide three data sources: two for indoor measurements and one for outdoor measurements
- Data should be updated every 60 seconds
- Each data point must include both temperature and timestamp
- Users must be able to view diagrams and evaluations of the collected data
- Users should have access to historical data to observe long-term trends
- The system must use containers for deployment
- In case of no network and or MQTT broker connection, the temperature data will be saved on an SD card for up to 24 h

Non-Functional Requirements

- The system should achieve an availability of 99.5%
- The system must remain reliable even if one container fails
- Data must be persistently stored in the database
- Automated unit tests must cover core functionalities, including correct data transmission, successful data storage, and simulation of failure scenarios

6.1.4 Quality Goals

Quality Goal	Description
Persistence	Sensor readings must be logged centrally (database) and locally (SD card), if offline -> No data loss
Data Integrity	Data must include timestamps and sequance to prevent corruption or duplication
Availability	The system must remain partially operational during network outages and recover automatically

6.1.5 Stakeholders

Role/Name	Expectations	Influence
Developer	Solution that is easy to maintain and fulfills all requirements for the project	Quality of Code, Clean Architecture, Final product
Supervisor	Correct methodology, clear documentation and tracability of results	Sets expectations and reviews the final product
Coaches	Clear documentation, preparation of meetings and clear presentation of the results for each meeting	Review of the final product and support for the implementation
User/Owner	Want to reduce their heating costs through stable temperature measurements and correct assessment of the building's isolation	Requires easy usability and trustworthy temperature data
Systemadministrator	Stable infrastructure, easy deployments and clear logs for easy maintenance	Configuration of the system

August 31, 2025

DianaTin23, deadmade, maratin23

6.2 Quality Requirements

6.2.1 1. Persistence

Temperature data must be reliably and permanently stored in the database, even in the event of temporary system or connection failures.

Measurable Criteria:

- **Data Loss Rate:** A maximum of **0.1%** of all recorded measurements may be lost.
- **Successful Write Operations:** At least **99.9%** of all database write operations must be completed without error.
- **Fallback Storage:** In case of missing connectivity, temperature data is written to the local SD card for up to 24h and synchronized once the connection is restored.
- **Retry and Confirmation:** Failed write operations to the central database are retried until confirmation is received.

Testability:

- Disconnect the system from the internet in a controlled way and verify that data is buffered on the SD card and later persisted in the database.
- Simulate database outages to check retry logic and final persistence.
- Run long-term operation tests with daily storage cycles (e.g., multiple days) to verify absence of data loss.
- Use SQL queries to compare the expected number of measurements with the actual count in the database, and to calculate the percentage of successful write operations, ensuring compliance with the defined thresholds.

6.2.2 2. Data Integrity

The recorded data must be correct, complete, and plausible to enable a reliable evaluation of the building's insulation.

Measurable Criteria:

- **Inconsistent Data Rate:** Less than **0.05%** of all records may be duplicates, incorrect, or implausible.
- **Validation Error Rate:** A maximum of **0.1%** of data may be rejected by validation mechanisms.
- **Automatic plausibility checks:**
    - **Range validation:** Outdoor readings must stay between -30 °C and 45 °C, indoor readings between -10 °C and 35 °C. Values outside this range are logged as warnings.
    - **Jump detection:** Sudden jumps >10 °C between consecutive readings are flagged.
    - **Difference and mean analysis:** Consecutive differences and moving averages are tracked to detect anomalies.
    - **Statistical window checks:** Mean and standard deviation over a defined time window are used to identify abnormal fluctuations.

Testability:

- Inject out-of-range or implausible test data and verify that the system logs warnings or rejects values.
- Simulate sudden temperature jumps to ensure they are flagged.
- Compare sensor readings against expected ranges (indoor vs. outdoor).

6.2.3 3. Availability

The system must remain functional even in the event of partial failures, so that users can always access the temperature data. Each critical service is deployed redundantly with at least two instances. If one instance fails, Traefik automatically routes traffic to the backup instance. All containers expose health checks, and stateless design ensures fast restart and recovery.

The system is resilient against the following single-instance failures:

- Website (frontend): one instance down → second instance continues serving requests
- REST API: one instance down → second instance handles API traffic
- Weather Data Worker: one instance down → second instance continues scheduled tasks
- MQTT Receiver: one instance down → second instance continues message processing

Measurable Criteria:

- **System Availability:** ≥ 99.5% overall operational time (software side)
- **Frontend Data Availability:** ≥ 99.5% of the time, current or last available data is accessible via the UI
- **Resilience Mechanisms:**
    - Redundant service instances per cluster (frontend, backend, workers)
    - Traefik load balancer distributes traffic and enables failover
    - Stateless service design for automatic restart or replacement
    - Health checks for all major containers
    - Local SD storage at Arduino nodes ensures sensor data buffering during backend or network outages

Testability:

- Controlled shutdown of one instance per cluster (frontend, REST API, Weather Data Worker, MQTT Receiver) to verify automatic failover via Traefik
- Disable one database or monitoring component to confirm health checks and recovery strategies
- Simulate network outage between Arduino and backend to verify SD-card buffering and later synchronization
- Long-term monitoring of uptime metrics to confirm compliance with ≥ 99.5% availability

September 1, 2025

DianaTin23

6.3 Architecture Constraints and Solution Strategy

6.3.1 Architecture Contraints

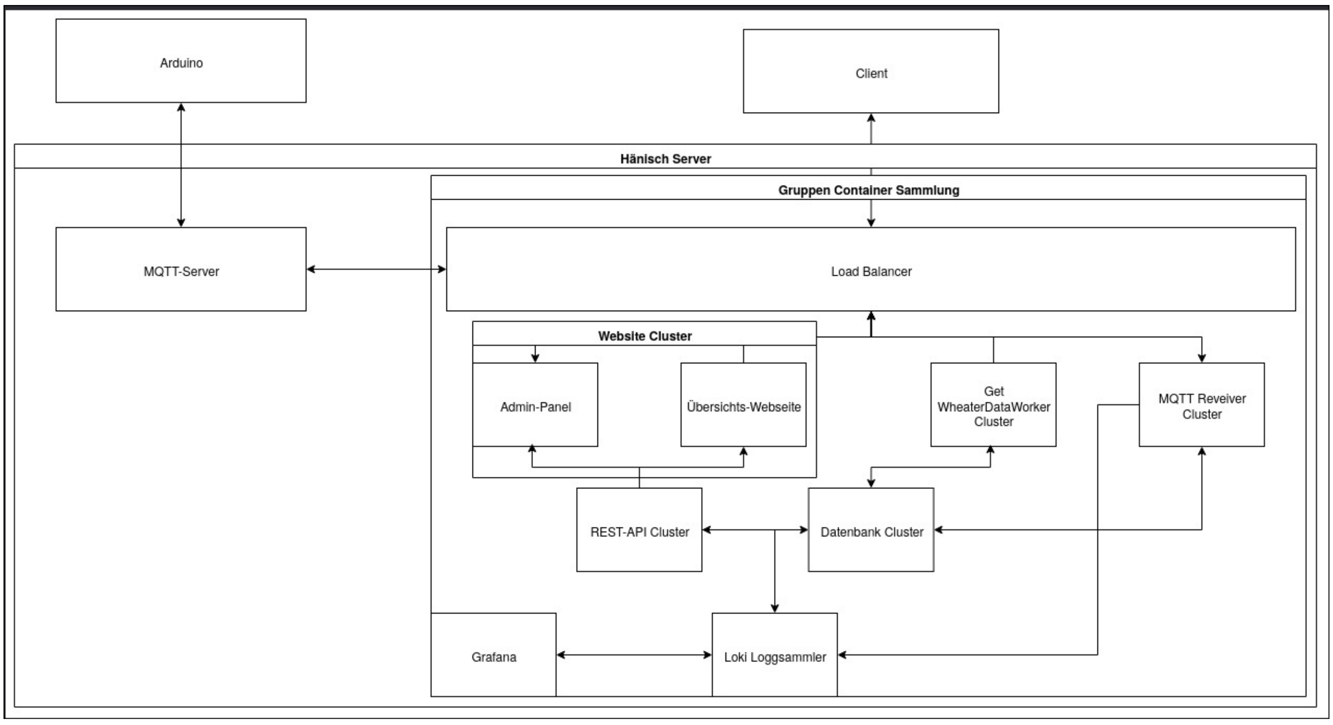
Category	Constraint
Technical	The project will be hosted on a single server provided by Prof. Hänisch
Technical	Indoor temperature measurement hardware is supplied by the university
Technical	At least two data sources are required, with at least one being an Arduino device
Technical	The hardware and database are not specifically designed for high reliability
Technical	The final system must run in a clean environment with no prior setup required
Organizational	Weekly meetings with a coach are scheduled for project discussions
Political	The submission deadline is the 04.09.2025

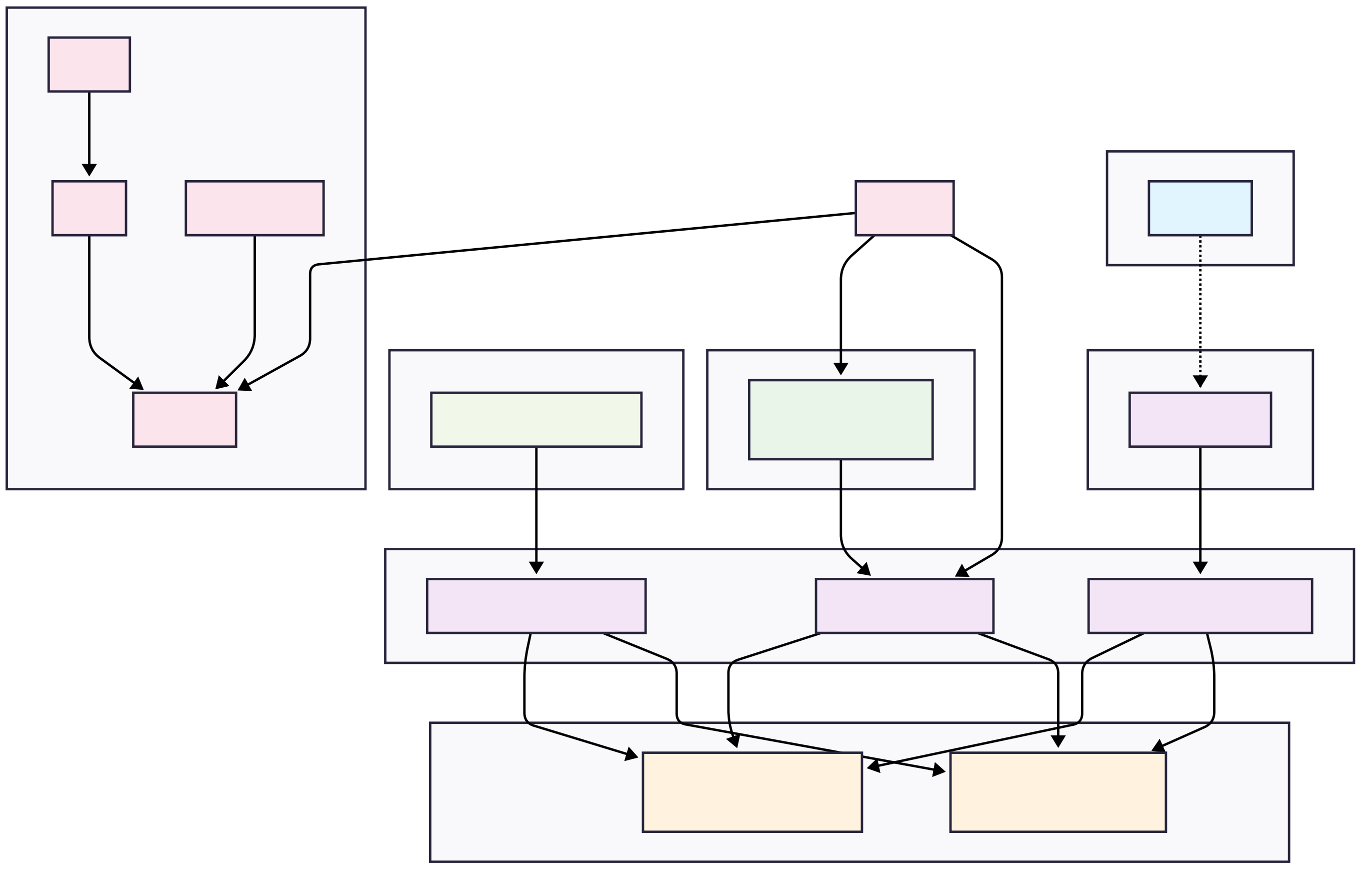
6.3.2 System Decomposition Strategy

All services are containerized and grouped under a shared container orchestration layer. The system is fronted by a load balancer, ensuring scalability and high availability. MQTT acts as the bridge between the hardware (Arduino sensors) and the backend. Logging and observability are handled via the Loki stack, and external visibility is offered via a user-facing website and Grafana deshboards.

Old Architecture

All services are containerized and grouped under a shared container orchestration layer (labelled "Gruppen Container Sammlung"). The system is fronted by a load balancer, ensuring scalability and high availability. MQTT acts as the bridge between the hardware (Arduino sensors) and the backend. Logging and observability are handled via the Loki stack, and external visibility is offered via a user-facing website and Grafana deshboards.





6.3.3 Organizational / Development Process Decisions

- **Source Control**: GitHub with structured branches and CI pipelines
- **Documentation**: Based on **arc42 template**, managed in MkDocs
- **Infrastructure as Code:** All services defined in `docker-compose.yml` and version-controlled
- **Architecture Decisions**: Documented using ADRs (Architecture Decision Records). The detailed technology choices are documented in 04 Architecture Decisions

6.3.4 Cross-cutting Concepts

The following concepts ensure consistency and support the quality goals:

Domain Concepts

- **ΔT metric:** Temperature difference between indoor and outdoor sensors as key indicator.
- **Sensor units:** Each Arduino node has a unique ID and physical placement (e.g., north vs. south façade).
- **Enriched records:** Each reading contains value, timestamp, and source ID.

Fault Tolerance

- **Redundant logging:** Default to central database, with SD card fallback on network/MQTT outages (buffering up to 24h).
- **MQTT QoS (1):** Guarantees at-least-once delivery.
- **Stateless services:** Enable fast restart and failover without data inconsistencies.

Architecture and Design Patterns

- **Message-driven architecture:** Asynchronous data flow via MQTT.
- **Microservices:** Independent services for ingestion, enrichment, API.
- **API gateway pattern:** REST API shields internal complexity and exposes a single entry point.

Development Concepts

- **Containerization:** Consistent deployment via Docker.
- **Continuous Integration:** Automated checks and tests before merges.
- **Infrastructure as Code:** Networks and dependencies tracked in source control.

Operational Concepts

- **Observability:** Logs via Loki, metrics via Prometheus, dashboards via Grafana.
- **Health monitoring:** `/health` endpoints with automated restart on failure.
- **Scalability:** Additional instances (frontend, MQTT receivers, workers) can be added behind Traefik.

August 31, 2025

DianaTin23, deadmade

6.4 Architecture Decisions

6.4.1 ADR 1: Backend Technology Stack

Status:

Accepted (July 2025)

Context:

System needs robust backend technology for REST API and worker services. Team has existing familiarity with C# development.

Decision:

Use .NET 9 with C# for all backend services (REST API, MQTT Receiver, Weather Worker).

Alternatives Considered:

Option	Pros	Cons
Node.js	Rapid iteration, huge ecosystem	Different stack; weaker static typing by default; less team experience
Go	High perf/concurrency; small binaries	Less team experience; different tooling
Python	Rich libs; fast prototyping	Lower throughput; weaker typing by default

Consequences:

Positive:

- Team familiarity reduces development time
- Strong typing prevents runtime errors
- Excellent tooling and debugging support
- Modern async/await support for I/O operations

Negative:

- Platform dependency (though mitigated by containers)
- Larger memory footprint than some alternatives

Neutral:

- Containerization standardizes runtime

6.4.2 ADR 2: Microservices Architecture

Status:

Accepted (July 2025)

Context:

System has distinct responsibilities: API serving, MQTT message processing, and weather data fetching. Need modularity and independent scaling.

Decision:

Split backend into separate services: REST API, MQTT Receiver Worker, Weather Data Worker.

Alternatives Considered:

Option	Pros	Cons
Monolith	Simple deploy; easy local dev	No independent scaling; fault blast radius
Modular monolith	Clear boundaries in one process	Still one deploy unit; limited isolation
Serverless	No servers to manage; auto-scale	Cold starts; platform coupling; ops visibility variance

Consequences:

Positive:

- Clear separation of concerns
- Independent scaling and deployment
- Fault isolation between services

Negative:

- Increased deployment complexity
- Network communication overhead between services

Neutral:

- Requires basic observability to manage complexity

6.4.3 ADR 3: Dual Database Strategy

Status:

Accepted (July 2025)

Context:

System needs both structured application data (users, authentication) and time-series sensor data with different access patterns.

Decision:

Use PostgreSQL for application data and InfluxDB for time-series sensor data.

Alternatives Considered:

Option	Pros	Cons
PostgreSQL + TimescaleDB	One stack; SQL everywhere	Ops complexity; perf tuning for time series needed
InfluxDB only	Optimized for time series	Awkward relational modeling; joins missing
SQLite + InfluxDB Lite	Simple, lightweight	Limited concurrency; feature gaps

Consequences:

Positive:

- PostgreSQL optimized for relational data and transactions
- InfluxDB optimized for time-series queries and compression
- Each database serves its specific use case efficiently

Negative:

- Two databases to maintain and backup

Neutral:

- Extract, Transform, Load (ETL) between stores is minimal

6.4.4 ADR 4: Observability Stack

Status:

Accepted (July 2025)

Context:

Distributed microservices architecture requires comprehensive monitoring, logging, and alerting capabilities.

Decision:

Loki for logs, Prometheus for metrics, Grafana for dashboards, Alloy as agent.

Alternatives Considered:

Option	Pros	Cons
ELK (Elasticsearch, Kibana)	Powerful search/analytics	Heavier footprint; more ops effort
OTel collector + vendor	Standards-based; flexible pipelines	Vendor lock-in and/or cost
Managed cloud observability	Minimal ops	Ongoing costs; data residency limits

Consequences:

Positive:

- Complete observability into system health and performance
- Industry-standard tools with good integration
- Unified dashboard for all monitoring data

Negative:

- Additional infrastructure to maintain

Neutral:

- Can swap components later

6.4.5 ADR 5: Traefik as Reverse Proxy

Status:

Accepted (July 2025)

Context:

Multiple services need unified entry point, SSL termination, and service discovery in containerized environment.

Decision:

Use Traefik as reverse proxy with automatic service discovery and HTTPS termination.

Alternatives Considered:

Option	Pros	Cons
Nginx	Mature; high performance	Manual routing/config; no auto-discovery
Caddy	Simple TLS; easy config	Fewer discovery features
HAProxy	Very fast; robust LB features	More manual config; fewer HTTP niceties

Consequences:

Positive:

- Automatic service discovery via Docker labels
- Built-in SSL certificate management
- Load balancing capabilities

Negative:

- Single point of failure if not properly configured
- Additional configuration complexity

Neutral:

- Replaceable by Nginx if needed

6.4.6 ADR 6: JWT Authentication Strategy

Status:

Accepted (July 2025)

Context:

REST API requires secure authentication mechanism. Need stateless authentication for microservices architecture.

Decision:

Implement JWT token-based authentication with Entity Framework for user management.

Alternatives Considered:

Option	Pros	Cons
Server-side sessions	Simple; revocation is trivial	Stateful; sticky sessions; scale limits
OAuth2/OIDC proxy	Standards-based; SSO ready	More moving parts; infra complexity
API keys	Simple; easy for machines	Poor granularity; rotation burdens

Consequences:

Positive:

- Stateless authentication scales well
- Standard approach with good library support
- Tokens can carry user claims

Negative:

- Token revocation complexity
- Requires secure token storage on client side

Neutral:

- Token TTL balances risk and UX

6.4.7 ADR 7: Docker Compose for Development Environment

Status:

Accepted (July 2025)

Context:

Complex multi-service architecture needs consistent development environment setup across team members.

Decision:

Use Docker Compose to orchestrate all services for local development.

Alternatives Considered:

Option	Pros	Cons
Dev Containers	Great DX; reproducible	Editor-coupled; learning curve
Kind/Minikube	Closer to k8s	Heavier locally; slower feedback
Scripts/Makefiles	Minimal tooling	Fragile; drift across machines

Consequences:

Positive:

- Consistent development environment
- Easy service dependency management
- Simplified onboarding for new developers

Negative:

- Requires Docker knowledge from all developers
- Resource intensive on development machines

Neutral:

- Can migrate to Kubernetes later

6.4.8 ADR 8: Frontend

Status:

Accepted (July 2025)

Context:

System needs a frontend to display charts from measured/collected temperature data and to generate API docs with TypeDoc.

Decision:

v0

JavaScript React app via Docker. Reason: quick start.
Issue: Schema changes not caught at build time caused runtime UI errors (no static typing).

v1

TypeScript React with Create React App (CRA). Reason: typing and better tooling.
Issue: TypeDoc generation failed due to CRA/tooling version conflicts.

React + TypeScript built with Vite for the frontend.

Alternatives Considered:

Option	Pros	Cons
CRA (TS)	Familiar, out-of-the-box setup	Tooling conflicts with TypeDoc
Next.js	SSR/ISR, ecosystem	Unneeded complexity for our use
Custom Webpack	Full control	More maintenance

Consequences:

Positive:

- TypeDoc works
- faster startup
- lean tooling

Negative:

- Some devs must learn Vite

Neutral:

- No server-side rendering (SSR) required

6.4.9 ADR 9: Hardware Platform Decision (board, sensors)

Status:

Accepted (July 2025)

Context:

MKR1010 and ADT7410 were provided. Requirements: offline buffering, precise time, dual sites.

Decision:

Use Arduino MKR1010 with RTC DS3231 and SD card; deploy two identical units.

Alternatives Considered:

Option	Pros	Cons
ESP32 boards	Wi-Fi integrated; strong community	Different toolchain; requalification
Different sensors	Potential accuracy/cost benefits	Revalidation effort; integration risk
Single hardware unit	Simpler setup	No north/south comparison; less robust

Consequences:

Positive:

- Local data persistence via SD card enables offline data storage for ≤24h
- Timestamp reliability through RTC with battery
- Compact hardware, low power, WiFi-ready (MKR1010)

Negative:

- RTC and SD modules require additional wiring and SPI/I2C handling
- Time must be synchronized manually once (e.g., via compile-time setting or initial sync)

Neutral:

- The Arduino MKR1010 was predefined, not evaluated
- Final visualization and backend will depend on further platform choices (e.g., MQTT, REST, database)

6.4.10 ADR 10: Arduino Development Environment Decision

Status:

Accepted (July 2025)

Context:

Arduino firmware needs modular builds and host-side unit tests.

Decision:

PlatformIO for builds; Unity with native target for tests.

Alternatives Considered:

Option	Pros	Cons
Arduino IDE	Easy; official	No native tests; inflexible structure
CMake toolchain	Flexible; IDE-agnostic	More setup; custom plumbing
Ceedling	Solid C test framework	Extra integration effort

Consequences:

Positive:

- Reproducible builds and consistent project structure
- PC-native unit tests for business logic (Unity, native target)
- Seamless VS Code integration
- Use of modern C++ structure and dependency management

Negative:

- Additional setup effort for non-Arduino users (e.g., Unity, test runners)
- Developers must learn PlatformIO's structure (src/lib/test)

Neutral:

- The PlatformIO toolchain abstracts away the underlying GCC setup
- Unit tests cannot cover board-specific behavior (e.g., Wire, SD, RTC) directly without mocks

6.4.11 Sources

Documenting Architecture Decisions by Michael Nygard

August 31, 2025

DianaTin23, deadmade

6.5 Context and Scope

6.5.1 Technical Context

The IsoPrüfi system operates in a distributed container-based architecture hosted on the DHBW Server infrastructure. It integrates multiple services for data ingestion, processing, storage, and visualization.

Components and Channels

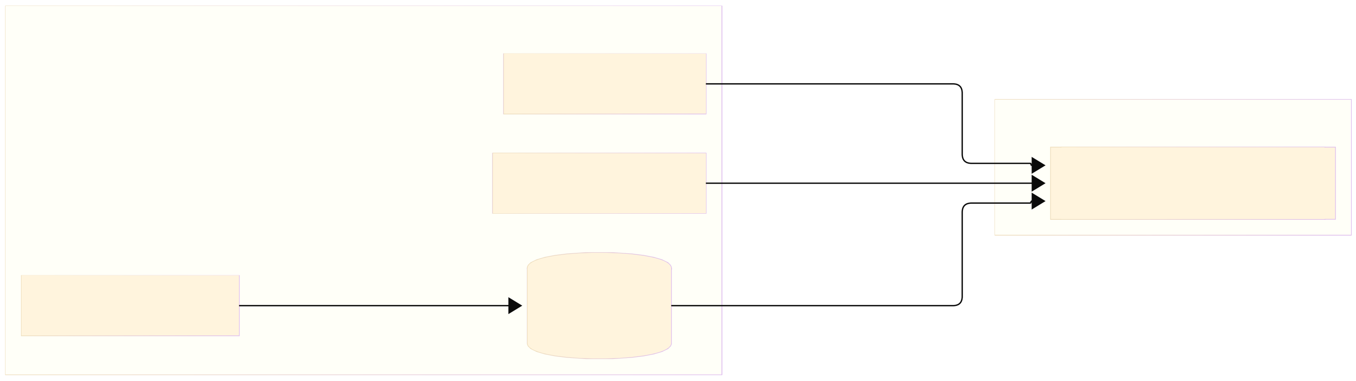
Component	Communication Channel	Description
Arduino	Wi-Fi / MQTT	Publishes temperature readings to the MQTT broker
MQTT Broker	MQTT	External broker for sensor communication
traefik	HTTP / HTTPS	Reverse proxy and load balancer, entry point to the system
isopruefi-backend	HTTP (REST)	Provides unified access to system data
isopruefi-frontend	HTTPS	User interface for visualizing measurements
influxdb	TCP / SQL	Time-series database for sensor data
postgres	TCP / SQL	Relational database for application data
loki	gRPC / HTTP	Collects and stores logs from all services
prometheus	HTTP	Collects metrics from services
grafana	HTTP	Dashboard for metrics and logs
alloy	HTTP / gRPC	Integrates Loki and Prometheus for observability
Weather API	HTTPS	Provides external weather data
Client (Browser)	HTTPS	Interacts with the frontend

For a detailed list of all Docker containers, their images, addresses and networks, see the separate documentation page: Docker Development Environment

Mapping I/O to Channels

I/O Type	Channel	Source	Destination
Temperature Reading	WiFi / MQTT	Arduino	MQTT Broker → MQTT Receiver Worker
Weather Data Pull	HTTPS (API)	Weather API Service	Weather Data Worker
Web Page Access	HTTPS	Client Browser	Traefik → React Frontend
API Request	HTTP / REST	Frontend (via Traefik)	REST API Service
Data Storage	SQL / TCP	Backend Services	PostgreSQL (app data), InfluxDB (time series)
System Logs	gRPC / HTTP	All Services	Alloy → Loki/Prometheus (visualized in Grafana)

Technical Context Diagram



August 31, 2025

DianaTin23, deadmade

6.6 Risks and Technical Debts

6.6.1 1. Description of the Process/System

Overview of the Entire Product:

- **Temperature Measurement and Transmission:**
    - Involves temperature sensors, RTC modules, Arduino, SD module/card, and access to online weather data
- **Data Storage:**
    - Utilizes a database for storing temperature data
- **Analysis/Evaluation:**
    - Data is analyzed and evaluated, with results visualized via website or analytics tools

Components Involved:

- **Temperature Measurement:**
    - Temperature sensors, RTC modules, Arduino, SD module/card, online weather data availability
- **Temperature Transmission:**
    - Network availability, server infrastructure
- **Data Storage:**
    - Database systems
- **Visualization/Analysis:**
    - Data availability, website, analytics platforms

Process Aspects:

- Data flow throughout the system
- Handling of failure and recovery scenarios

6.6.2 2. Error Analysis

Possible Errors

- Incorrect or missing data
- Unavailability of services or functions (e.g., website, Grafana)

Causes

- Compatibility issues due to software or hardware updates
- Security vulnerabilities
- **Temperature Measurement:**
    - Sensor errors (e.g., incorrect calibration, hardware malfunction, sensor failure, power supply issues, incorrect interval configuration)
    - Misassignment of data (e.g., north/south confusion)
    - Weather service outages
- **Data Transmission:**
    - Network outages or connectivity issues
    - Duplicate data transmission

- **Data Storage:**
    - Incorrect or duplicate entries
    - Database corruption or failure
- **Visualization/Analysis:**
    - Website or Grafana unavailability
    - Incorrect data presented for visualization

Impacts

- Gaps in data analysis
- Misinterpretation or incorrect assessment of results
- Lack of long-term evaluation or comparison basis
- No or limited access to collected data

6.6.3 3. Evaluation of Errors and Consequences

Error	Probability of Occurrence	Severity	Probability of Detection	Risk Priority Number
Sensor error	2-3 (unlikely)	8-9 (severe)	2-3 (inevitable detection)	32-81
Misassignment of data	3 (low)	6-7 (disturbance)	5-6 (only detected during targeted checks)	90-126
Weather service outage	1 (almost impossible)	8-9 (severe)	2-3 (inevitable detection)	16-27
Network outage	2 (unlikely)	8-9 (severe)	2-3 (inevitable detection)	31-45
Duplicate transmission	2 (unlikely)	2 (irrelevant)	5-6 (only detected during targeted checks)	20-24
Incorrect/missing entries	2-3 (unlikely)	8-9 (severe)	3-4 (high probability of detection)	48-108
Database corruption	2-3 (unlikely)	8-9 (severe)	3-4 (high probability of detection)	48-108
Website/Grafana malfunction	1 (almost impossible)	8-9 (severe)	2-3 (inevitable detection)	16-27
Power outage	3 (low)	8-9 (severe)	2-3 (inevitable detection)	48-81

6.6.4 4. Corrective actions

Error	Risk Priority Number	Mitigation Measure
Sensor error	32-81	-
Misassignment of data	90-126	Implement data validation and labeling checks
Weather service outage	16-27	Use fallback data sources
Network outage	31-45	Local storage of data on the Arduino
Duplicate transmission	20-24	-
Incorrect/missing entries	48-108	Input validation
Database corruption	48-108	-
Website/Grafana malfunction	16-27	Monitor uptime

6.6.5 5. Technical Debts

Debt	Impact	Mitigation	Priority
Single-server deployment (no HA for DB/Traefik)	Outage stops whole system; RTO/RPO undefined	Define RTO/RPO; periodic restore drills; consider DB replication later	High
External single MQTT broker	Ingestion is SPOF; no controlled failover	Document broker SLA; add reconnect/ backoff; plan broker redundancy/bridge later	High
SD-card buffering deduplication	Risk of duplicate inserts on reconnect	Idempotent writes (sensorId + timestamp + seq unique); DB upsert/unique index	High
Time synchronisation of sensors	Clock drift → wrong ΔT and ordering	Regular NTP sync or backend time anchor; RTC drift check procedure	High
Missing/uneven health/ readiness endpoints	Load balancer may route to bad pods	Standardize `/health` and `/ready`; Traefik forward-auth or ping checks	Medium
No alerting rules/SLOs	Failures unnoticed; 99.5% not enforced	Prometheus alert rules + Grafana alerts; SLO dashboards for availability	Medium
Secrets in env files	Leakage risk; no rotation	Use Docker secrets; rotate regularly; restrict file perms; avoid committing	High
TLS/auth on MQTT not specified	Data spoofing/sniffing possible	Enable TLS; client auth (user/pass or certs); topic ACLs	High
Schema/migration strategy	Breaking changes risk data loss	Versioned EF migrations; InfluxDB bucket retention + downsampling plan	Medium
Config scattering (topics, URLs)	Drift and hidden coupling	Central config per env; validated at startup; document defaults	Medium
Limited automated fault tests	Availability regressions unnoticed	CI: chaos/failure tests (DB down, broker down, network flap)	Medium
Weather API limits/caching	Rate-limit failures; latency	Add caching, retries with jitter, circuit breaker, fallback to last-known	Low
Backup without periodic restore test	False sense of safety	Quarterly restore test; document runbook; verify integrity checks	High
Logging/PII retention not defined	Storage bloat; compliance risk	Retention policy in Loki; scrub PII; log level guidelines	Medium
Rate limiting/DoS on API	Resource exhaustion	Traefik rate limits; API quotas; request size limits	Medium
Ownership/runbooks	Slow incident response	Define service owners; on-call matrix; SOPs for common incidents	Low

6.6.6 Sources

FMEA from the Orgahandbuch (Bundesministerium des Inneren)

August 31, 2025

DianaTin23, deadmade, maratin23

7. License

July 14, 2025

DianaTin23