SafeHelmet

by CleoTech

Mussini Simone

Siena Andrea

Stomeo Paride

Taccini Lorenzo

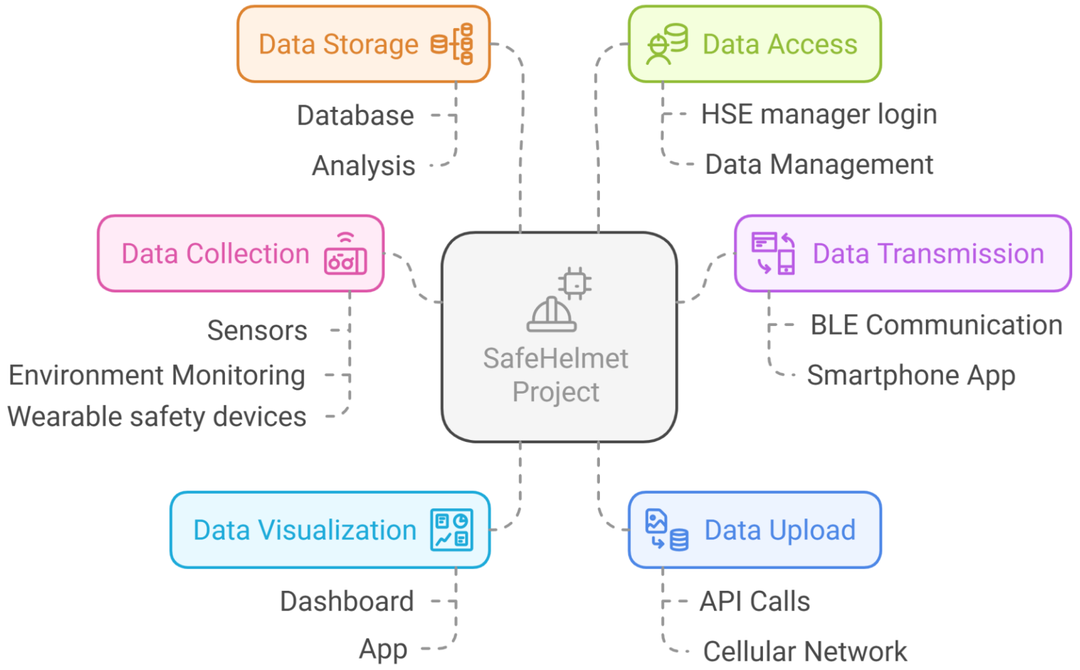


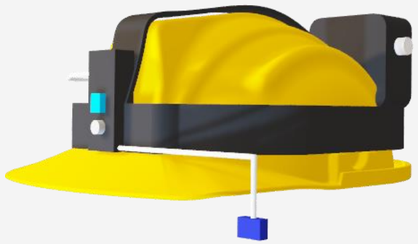
What is SafeHelmet?



Not only a physical product...

But an entire infrastructure.





SafeHelmet

Main goals



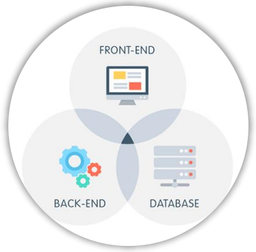
Enhance workers safety in critical environments



Worksite situation and logging history, at a glance



Modular approach, reducing waste



Full Stack development, infrastructure made from the ground up

Infrastructure - Overview



SafeHelmet



Smart Helmet

- Sensors and connectivity onboard
- Data retrieval and sending to **smartphone**



Smartphone

- Visualize data
- Send data to the **cloud**



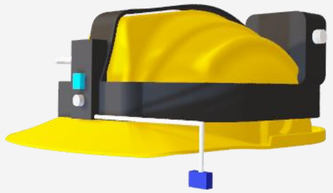
Cloud

- Database logging
- Data grouping from all sources
- Signaling anomalies



Web Dashboard

- Grouped data visualization
- Worksites monitoring



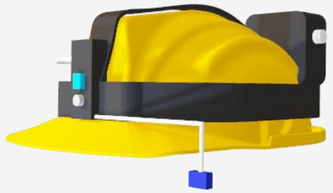
SafeHelmet

1. Smart Helmet



Smart Helmet periodically senses the surrounding environment:

- Temperature
- Humidity
- Brightness
- Smoke and nocive gases
- Accelerometer data
- Incorrect worker posture



SafeHelmet

1. Smart Helmet



- Temperature
- Humidity
- Brightness
- Smoke and nocive gases
- Accelerometer data
- Incorrect worker posture

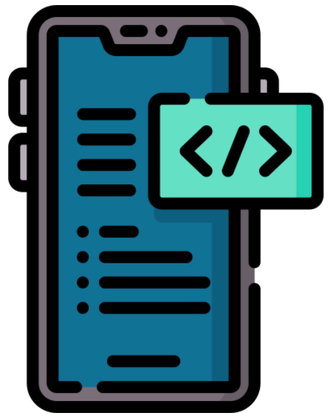
All retrieved data is sent to the smartphone via BLE



SafeHelmet

2. Mobile App

All retrieved data is sent to the smartphone via BLE

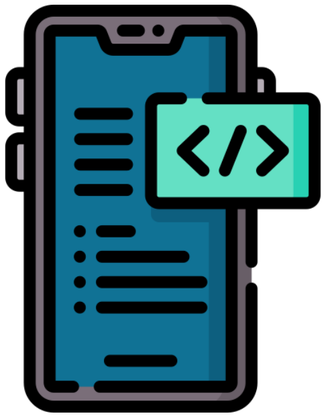


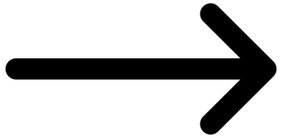
- Worker login
- Smart Helmet pairing
- Last reading visualization
- Anomalies and risks notification



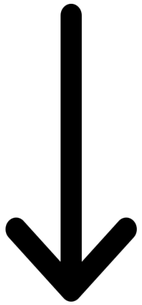
SafeHelmet

2. Mobile App



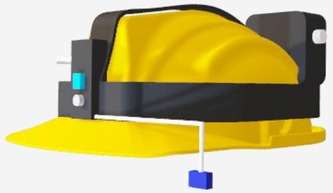


- Worker login
- Smart Helmet pairing
- Last reading visualization
- Anomalies and risks notification



Mobile application propagates data to the Cloud infrastructure via cellular network



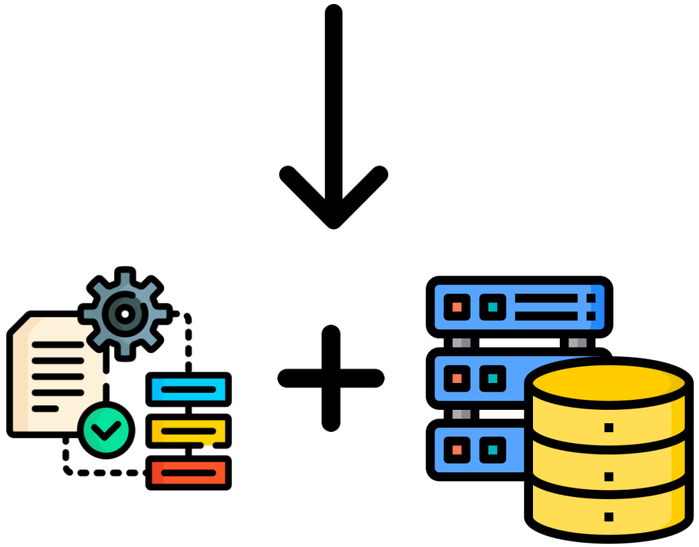


SafeHelmet

3. Cloud infrastructure (backend)

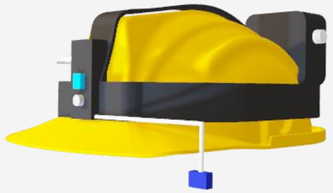
Mobile application propagates data to the Cloud infrastructure via cellular network





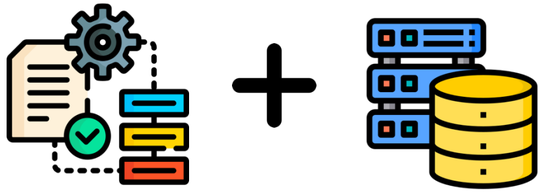
Backend system:

- **Stores all the received data into a Database**
- **Examines readings to detect risk-prone measures and marks them as anomalous**



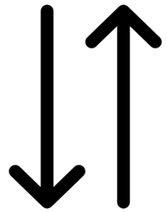
SafeHelmet

4. Web dashboard



Backend system:

- **stores all the received data into a Database**
- **Examines readings to detect risk-prone measures and marks them as anomalous**





Web dashboard:

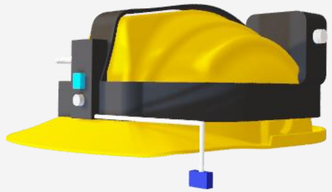
- **exchanges data with Backend system**
- **allows managers to interact with workers' and worksites' digital twins, hire and assign new employees, and much more.**

In depth



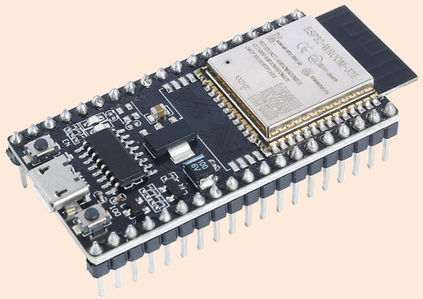
- Smart Helmet -

In depth - Hardware



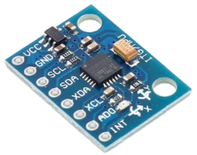
SafeHelmet

Smart Helmet module utilizes the following technologies onboard:



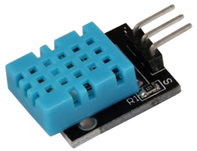
ESP32-WROOM *µ-controller*

- Dual-core 32-bit Xtensa® LX6 microprocessor, at 240 MHz
- 520 KB of on-chip SRAM
- External **4MB SPI flash**
- Wi-Fi 802.11 b/g/n
- Bluetooth v4.2 with **BLE**
- I^2C communication protocol



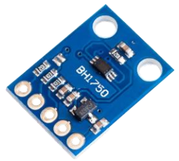
MPU6050

- Accelerometer and gyroscope, based on GY-521 sensor
- I^2C protocol



DHT11

- Temperature and humidity sensor
- One-wire communication protocol



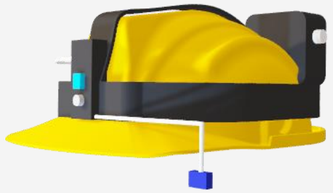
BH1750

- digital Ambient Light Sensor, based on GY-302
- I^2C protocol



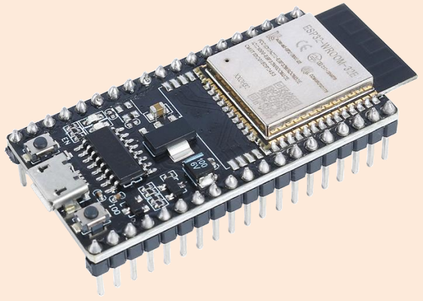
MQ2, MQ4, MQ7

- Set of analog-digital sensors detecting methane, LPG, smoke, CO (Carbon Monoxide), Propane, and more



SafeHelmet

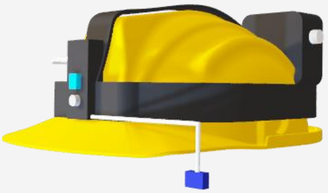
In depth – Helmet Software and features



ESP32-WROOM

- **Micropython** onboard – versatility and ease of use.
- Exploiting **BLE** communication with the smartphone using a **custom communication protocol**.
- Interrupt-driven execution, with **custom virtual timer implementation** exploiting high precision physical timers.
- **Safety devices detection**: Smart Helmet detects whether the worker is correctly wearing safety devices, notifying it to the backend.
- **Smart standby**: stop data collection and sending when not being worn by worker → **energy saving**.

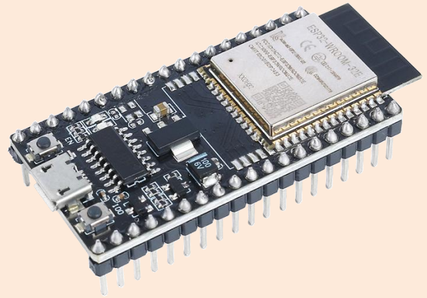
In depth – Data collection



- **Variable retrieval time** for each measurement:
  - Energy efficient → less critical data is collected fewer times than relevant measures.
  - Increased accuracy for data that really matters.

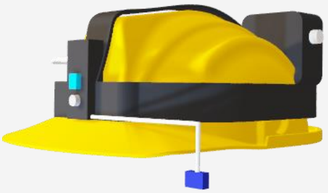
- Data collection methods run exploiting **virtual timers' *ad-hoc* infrastructure**, developed to overcome physical timers' limitations.

- Additional configurations allow ***custom behaviors***:
  - Smart standby mode
  - MQ sensors preheating phase at boot
  - data packing/formatting to be sent through BLE characteristics.

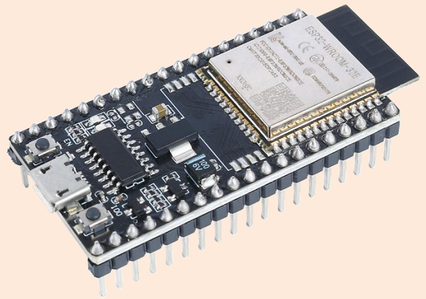


ESP32-WROOM

In depth – Data collection



SafeHelmet



ESP32-WROOM

Temperature Humidity Brightness

Mean avg

Gas detection

- Smoke presence
- CO presence
- Methane/LPG presence

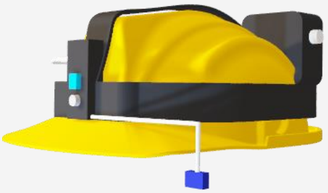
Accelerometer data

- Mean avg and std deviation for each axis
- Mean avg and std deviation for g acceleration
- Maximum acceleration measured in interval
- % of time spent keeping an incorrect posture

Safety devices detection

- Welding mask (worn/not worn)
- Gas mask (worn/not worn)

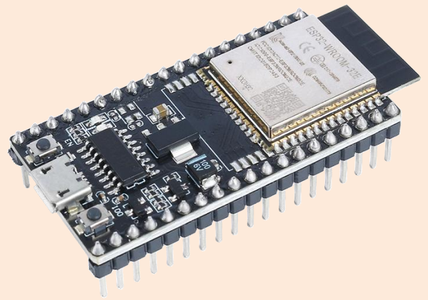
In depth – BLE communication



SafeHelmet

Given a **20 byte per-characteristic size constraint**, all retrieved data is distributed on 4 characteristics:

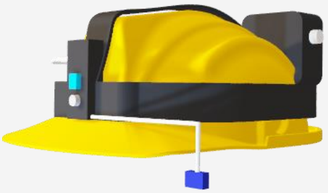
- Custom BLE descriptor is added to each characteristic to uniquely identify it.
- NOTIFY type → push notification events on the smartphone.
- WRITE type → smartphone can write data retrievable by *µ-controller.*



ESP32-WROOM



In depth – Anomaly Notification



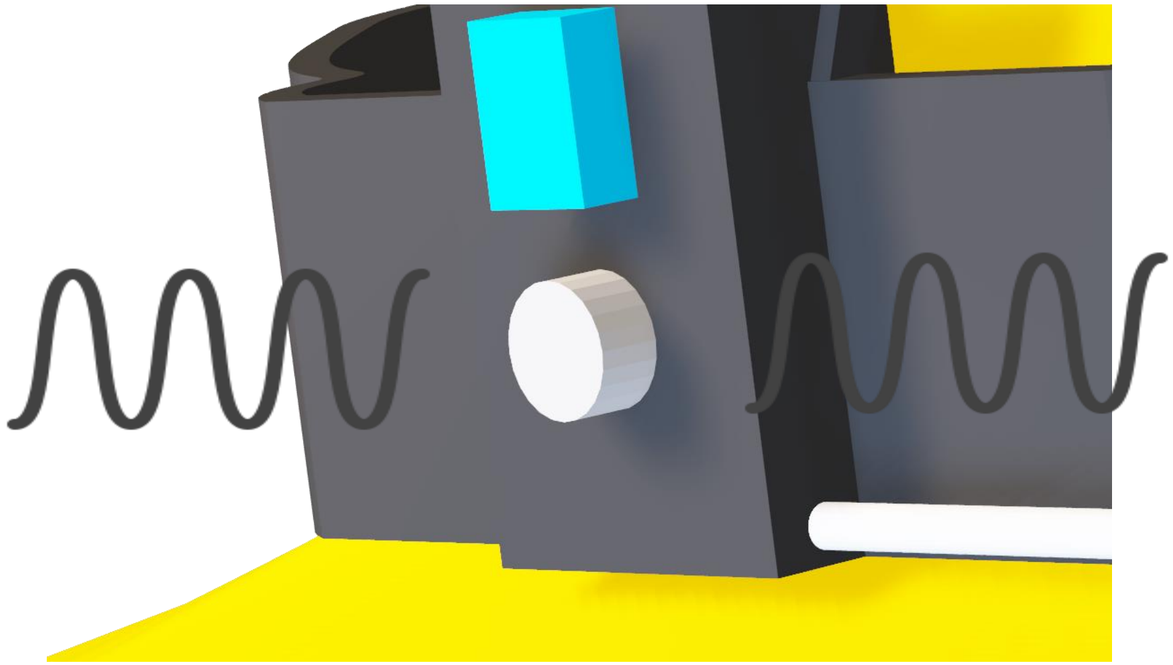
SafeHelmet

Smart Helmet notifies the worker about recent anomalies through the help of an **integrated vibration motor**:

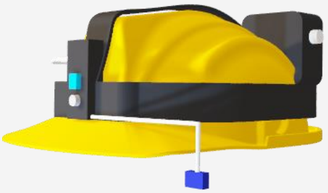
- **Smoke and gas personal anomalies** are notified **as soon as possible** to the worker, without waiting for backend evaluation → high risk situations.
- **Other anomalies**, regarding the worker himself and/or his colleagues, are retrieved once every few minutes **after backend logging and validation.**



ESP32-WROOM



Smart Helmet – Pros and cons



SafeHelmet

- **MicroPython**:
  - ✅ Versatility and ease of use
  - ✅ Lots of libraries for sensors and components
  - ❌ Less performant than other firmwares (FreeRTOS, Arduino Framework) → exploits only 1-core

- **Hardware (sensors and actuators)**:
  - ✅ Cheap and readily availabile
  - ✅ Lots of documentation and libraries online
  - ❌ Precision could be better (MQ sensors only detect presence and not level, etc....)
  - ❌ Miniaturization isn’t really a thing

In depth



- Mobile Application -

In depth – Mobile app



SafeHelmet Mobile was developed in **Kotlin for Android native** using the following technologies:

SAFEHELMET MOBILE

Kotlin

- Programming language for source code

Jetpack Compose

- Android’s modern native UI toolkit with declarative syntax.

OkHttp3

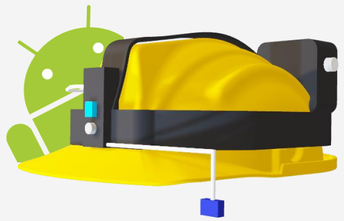
- Powerful HTTP client for network API responses handling

Bluetooth Low Energy (BLE)

- Enables communication with BLE devices.

JSON

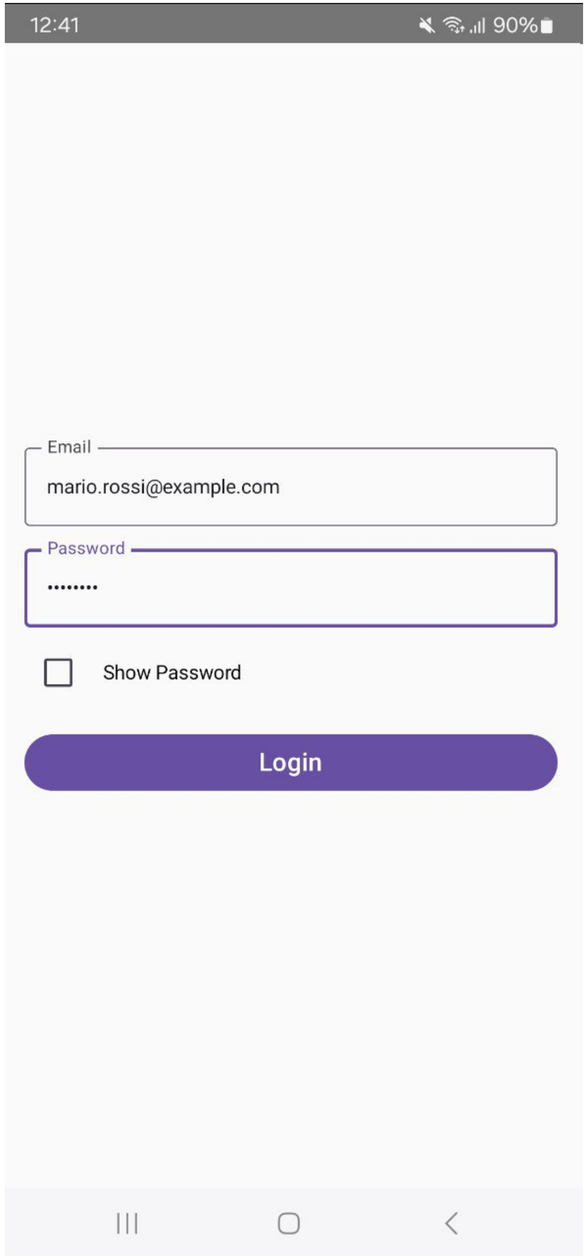
- Used for app-backend data exchange through RESTful APIs.

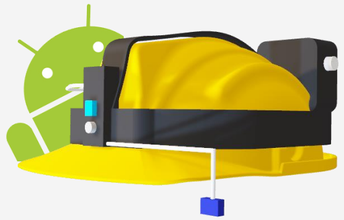


SafeHelmet

In depth – Login and connection

At the start of the shift, the worker **logs in to the app using his credentials**.

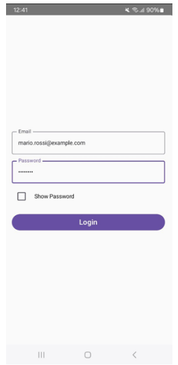


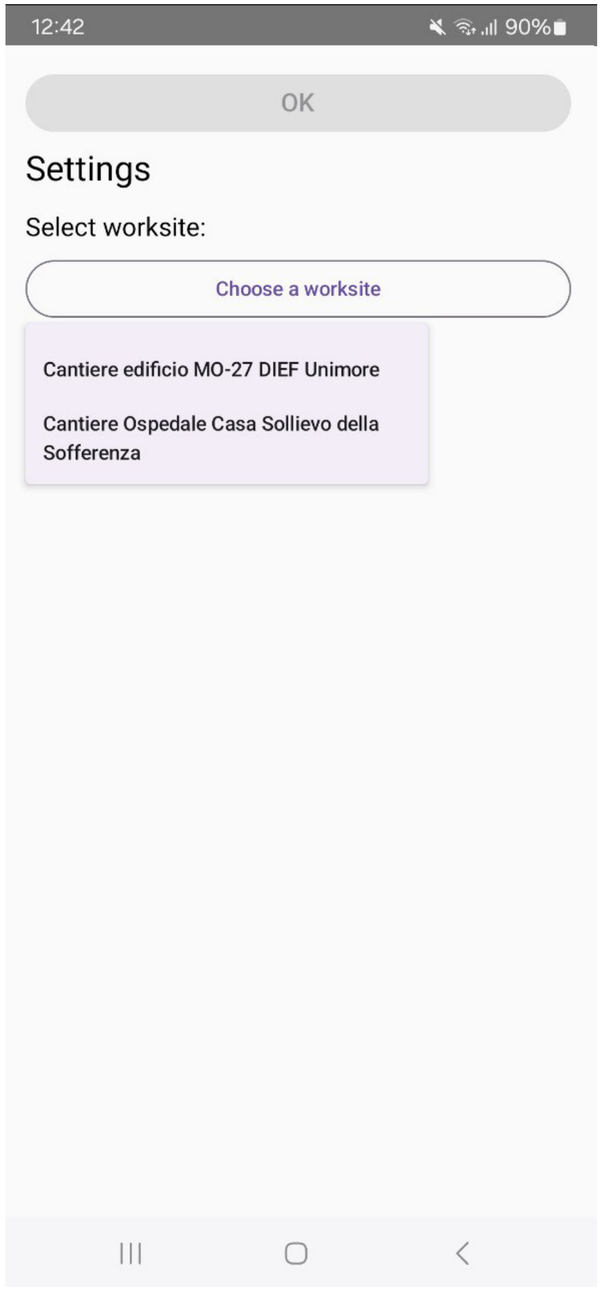


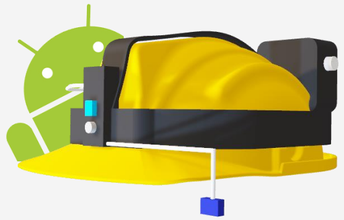
SafeHelmet

In depth – Login and connection

He **chooses the worksite in which he’s checking in** among all the ones he’s assigned to.



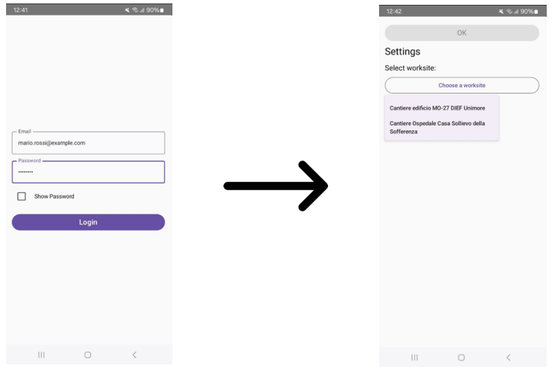


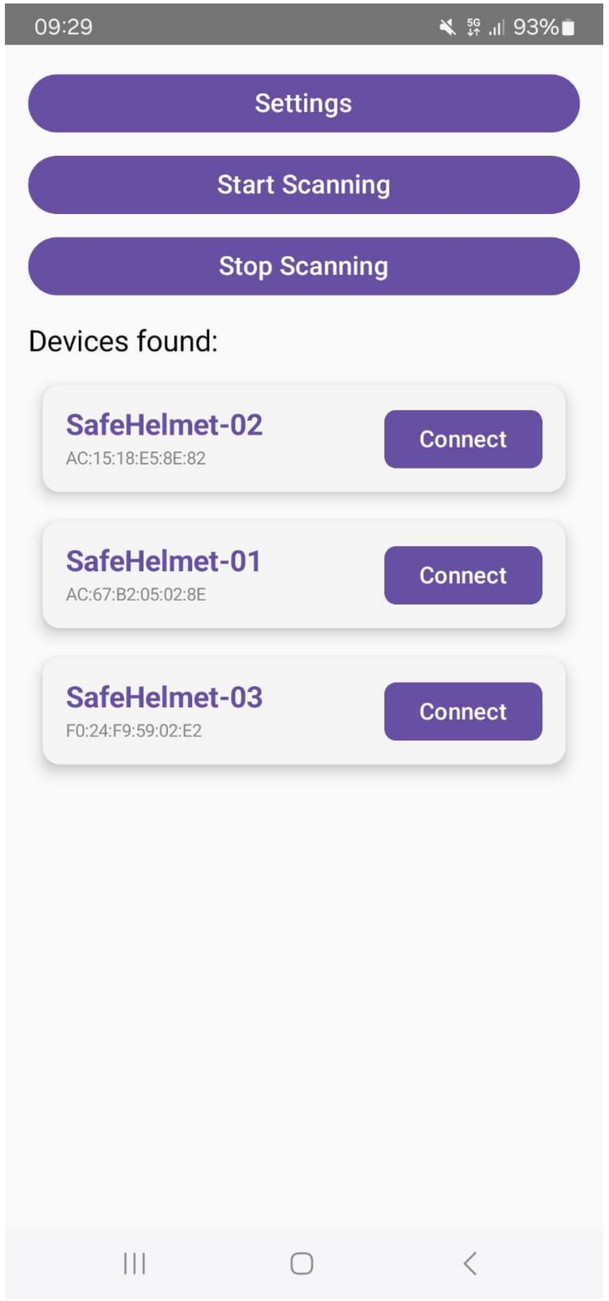


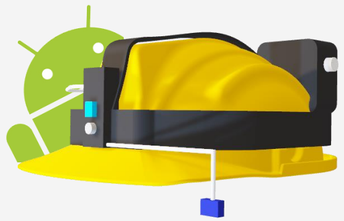
SafeHelmet

In depth – Login and connection

He can now pick up a Smart Helmet, power it on and **connect it to the app**, that shows a list of all the available devices in range.



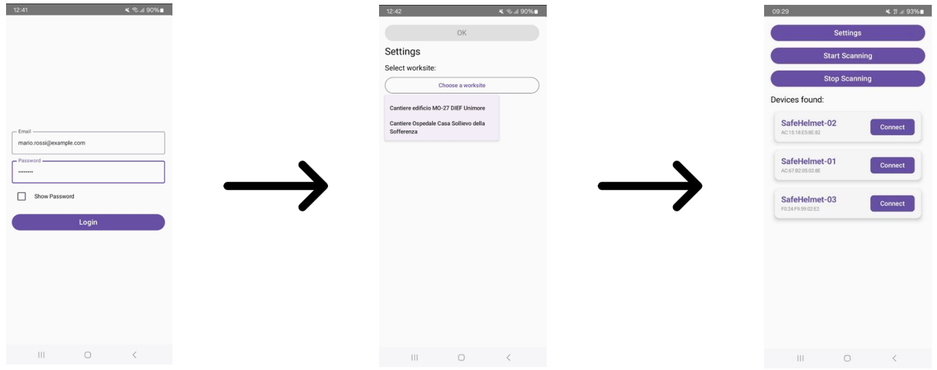




SafeHelmet

In depth – Readings' visualization

The smartphone is now paired with the chosen Smart Helmet and begins data exchange, **showing values of the last received reading** on screen.





In depth – Data collection



SafeHelmet

Data retrieval from BLE characteristics is managed by 3 Kotlin instances:

- **BLEManager class**: manages subscription and data retrieval to and from ESP32 BLE characteristics.
- **BaseParse class**: extracts and parses sensor data from the characteristic byte array.
- **ParseCollector object**: formats parsed data into JSON, then sends it to the backend via API POST request.

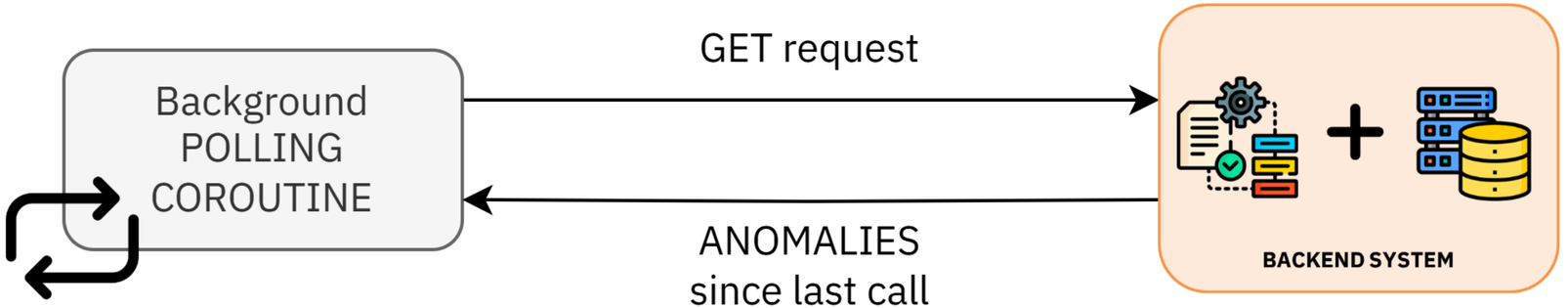


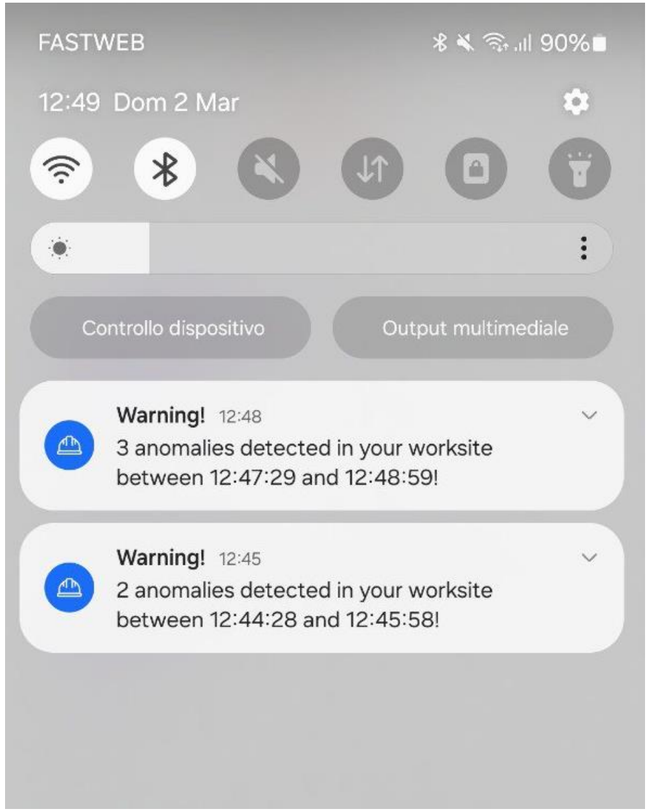
In depth – Polling and Anomalies



SafeHelmet

SafeHelmet mobile app keeps working in background until disconnection from the Smart Helmet.





This information is displayed through **smartphone notifications** to the user.

Mobile App – Pros and cons

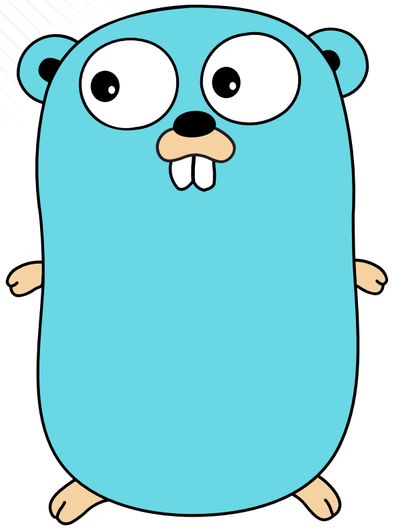


SafeHelmet

- **Kotlin programming language**:
  - ✅ Performance similar to Java
  - ✅ Versatile and complete, possibilities are endless
  - ❌ Steep learning curve → overwhelming at first glance
  - ❌ Kotlin Multiplatform is early stage, incomplete and lacking in support and documentation

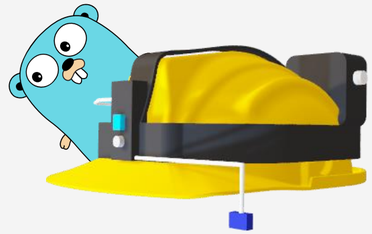
- **SafeHelmet Mobile App**:
  - ✅ Easy to use
  - ✅ Works in background
  - ✅ Sleek and modern interface
  - ❌ Android only
  - ❌ Only compatible with cutting-edge Android versions (13+, SDK 32+)

In depth

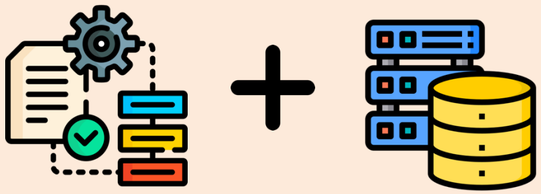


- Backend system and Database -

In depth – Backend



SafeHelmet Backend system has been developed using these technologies:



BACKEND SYSTEM

GO

- Programming language for source code

GORM

- ORM for managing DB creation and migrations

OpenWeather APIs

- Retrieve **dynamic thresholds** about weather-related anomalies

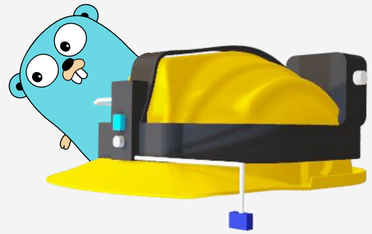
PostgreSQL 16

- Database manager

Render

- Backend and database Cloud hosting service

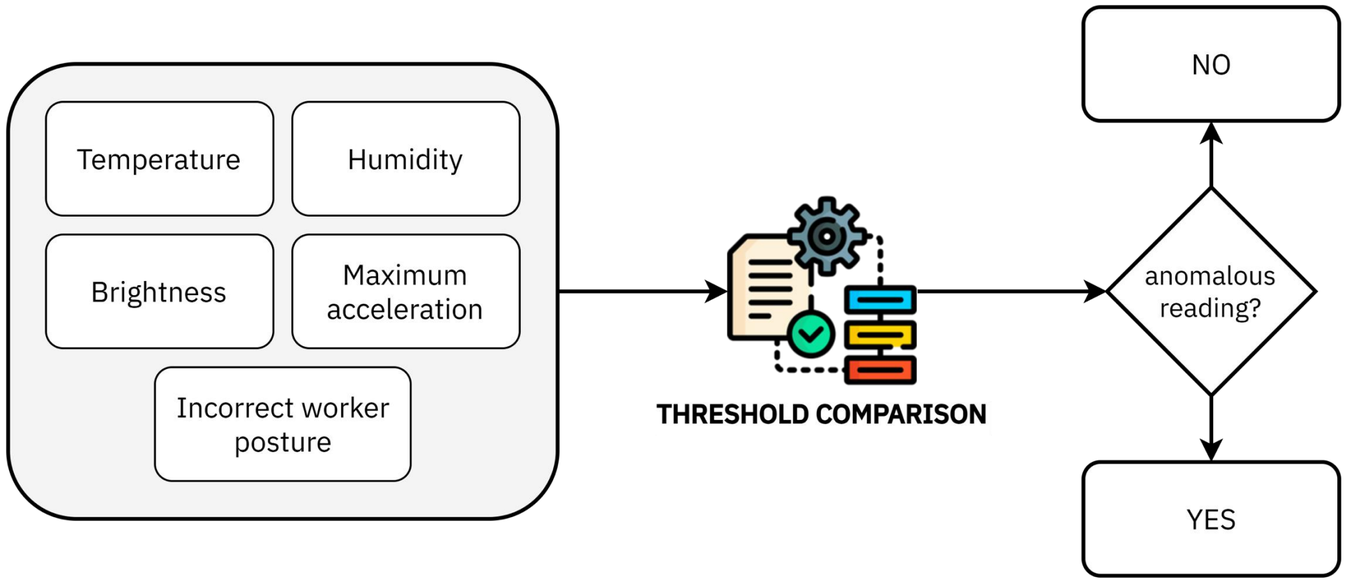
In depth – Reading Representation



SafeHelmet

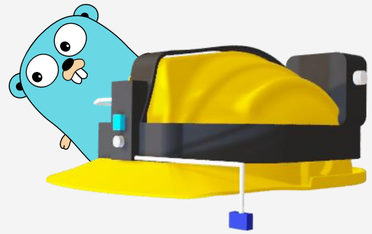
- Every Smart Helmet reading is stored in “**readings**” DB table.

- Safety-critical values of the reading are compared by the backend with **static or dynamic thresholds** to determine whether to notify an anomaly or not.



readings	
id	int
read_at	timestamp
attendance_id	int
temperature	float
weather_max_temperature	float
weather_min_temperature	float
anomalous_temperature	bool
humidity	float
weather_humidity	float
anomalous_humidity	bool
brightness	float
weather_brightness	float
anomalous_brightness	bool
methane	bool
carbon_monoxide	bool
smoke	bool
using_welding_protection	bool
using_gas_protection	bool
max_g	float
crash_detected	bool
incorrect_posture	float
anomalous_posture	bool
anomaly	bool

In depth – Attendances



SafeHelmet

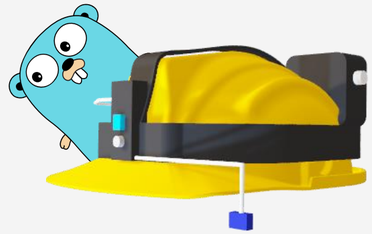
- **Each reading is assigned to an attendance**, that represents a work shift:
  - ✓ Of a certain **worker**
  - ✓ With a certain **helmet**
  - ✓ In a certain **worksite**

attendance	
id	int
worker_id	int
worksite_id	int
helmet_id	int
start_at	timestamp
end_at	timestamp

- This table also keeps track of **workers' start/end shift times** → Once a day, when a Smart Helmet module initially pairs to the mobile app, a new entry is created.

- When a new reading is received and stored, it's assigned to the current shift → **If any accident happens you can determine who was on the jobsite and may have caused it**.

In depth – Weather APIs



- **Exploiting OpenWeather APIs to get:**
  - Minimum predicted temperature
  - Maximum predicted temperature
  - Predicted humidity

- The values are retrieved for each worksite **every hour from 8:00am to 20:00pm**.

- These predicted values are then confronted with the ones read from the helmets to check for discrepancies → dynamic thresholds

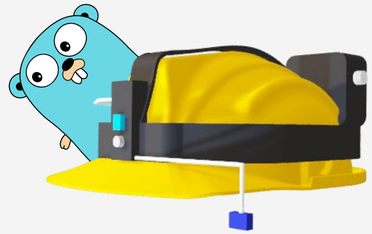
- Each worksite can have **custom thresholds** based on these dynamic values.

$$MaxPredictedTemp - ActualTemp > TempThreshold$$

$$ActualTemp - MinPredictedTemp > TempThreshold$$

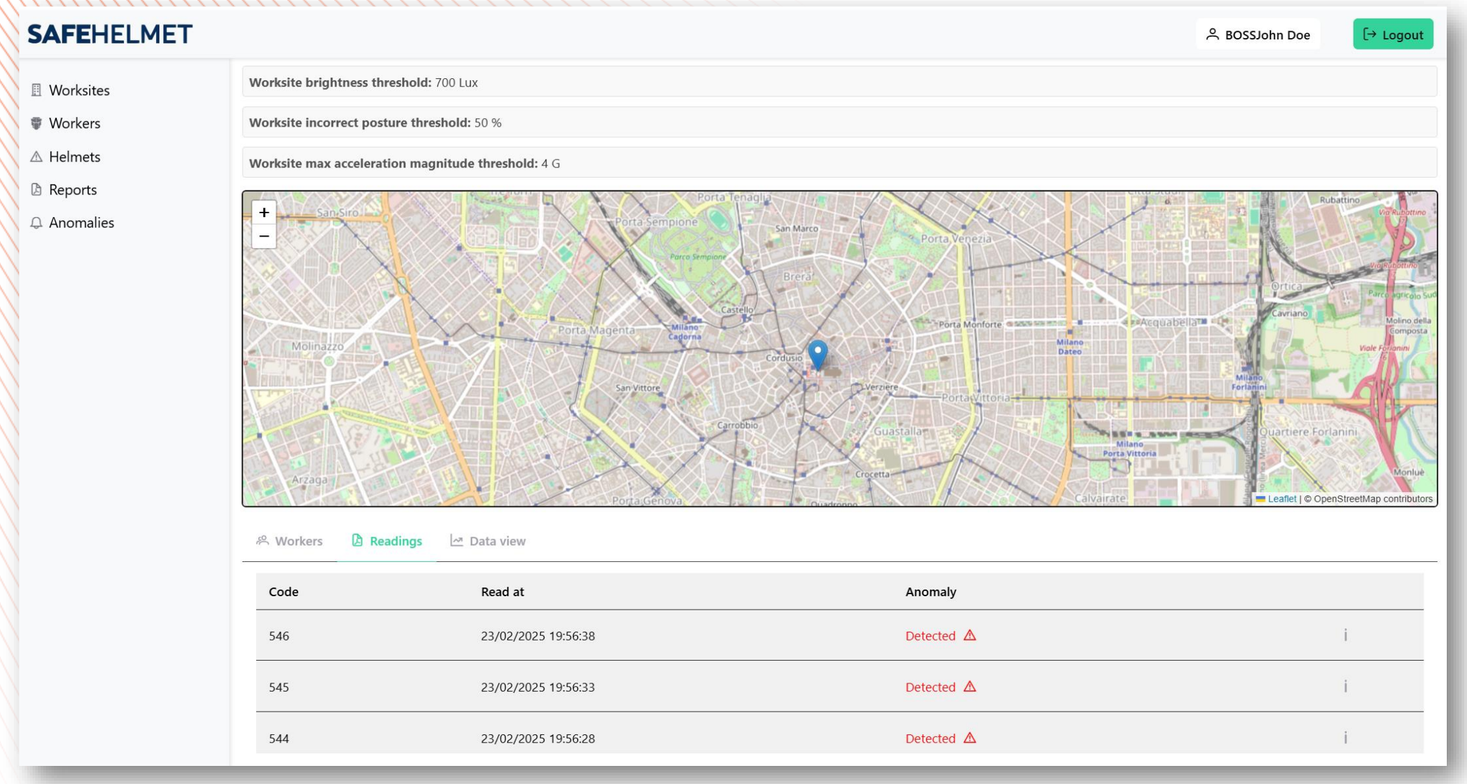
$$|PredictedHum - ActualHum| > HumThreshold$$

Backend - Pros and Cons



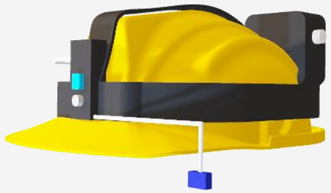
- **OpenWeather** APIs:
  - ✅ Free plan allows a lot of requests per-day
  - ✅ Fast responses
  - ❌ Free plan APIs are limited (no UV, Brightness nor air pollution data)
- **Render** for backend and DB hosting:
  - ✅ Fast service
  - ✅ Easy to setup and nice auto-deployment integration with Github
  - ❌ Free plan has an inactivity timer (impossible to run a coroutine as we did for the weather APIs)
- **GO** language and **GORM**:
  - ✅ Complete and easy to learn language
  - ✅ Better, performance-wise, than most alternatives (Node.js, Python, ...)
  - ❌ Deleting a table row in GORM requires lots of hooks to remove depending entries

In depth



- Web Dashboard -

In depth – Web dashboard



SafeHelmet Web Dashboard has been developed using these technologies:



Angular

- Framework for single page applications

PrimeNG

- UI Suite for Angular

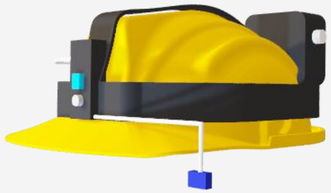
Github Pages

- Hosting service

Chart.JS

- Dynamic charts generation library

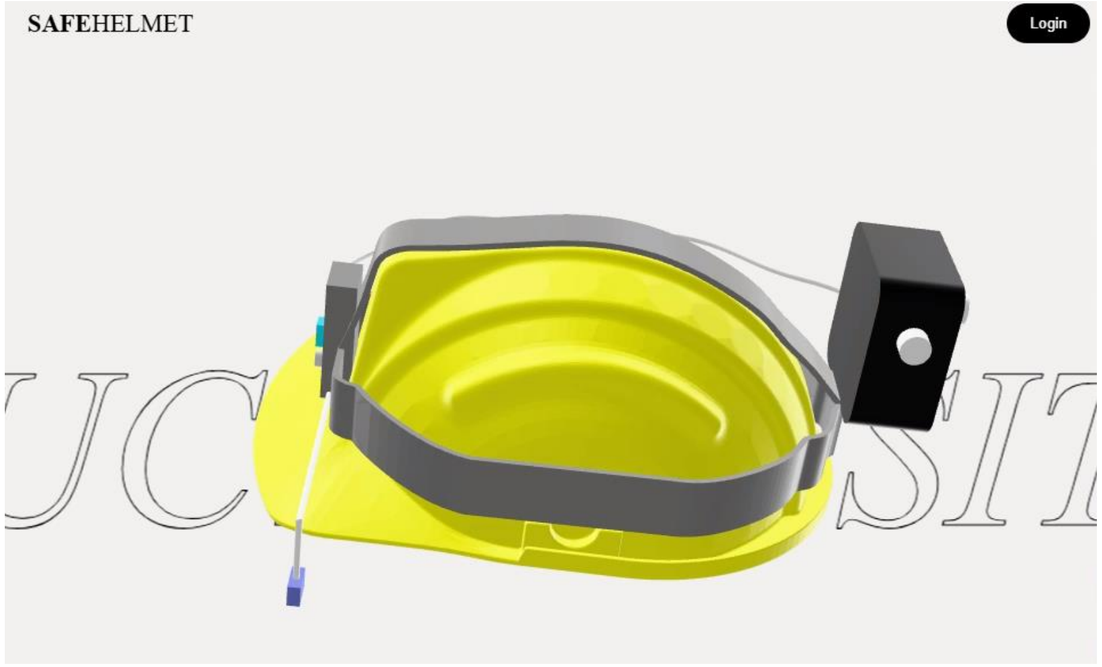
In depth – Main page aim

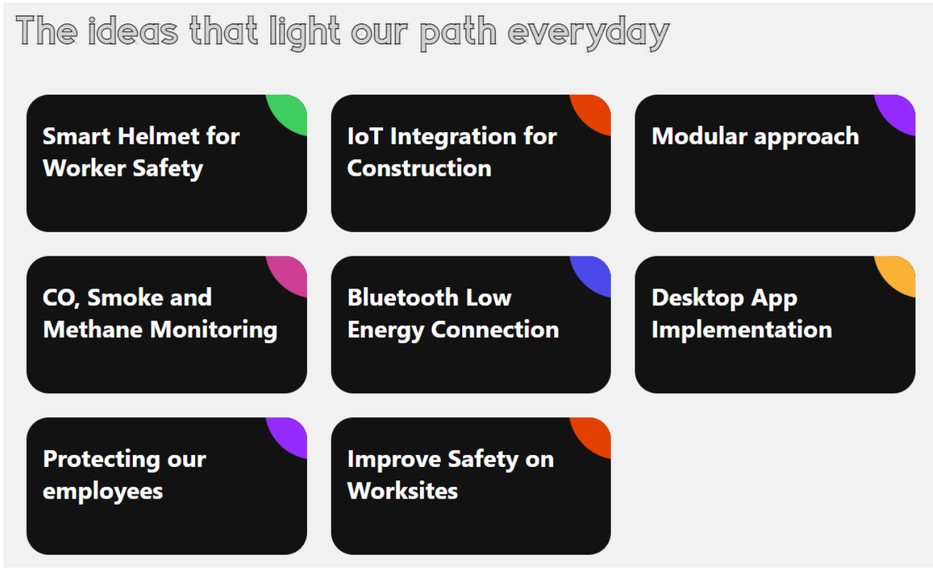


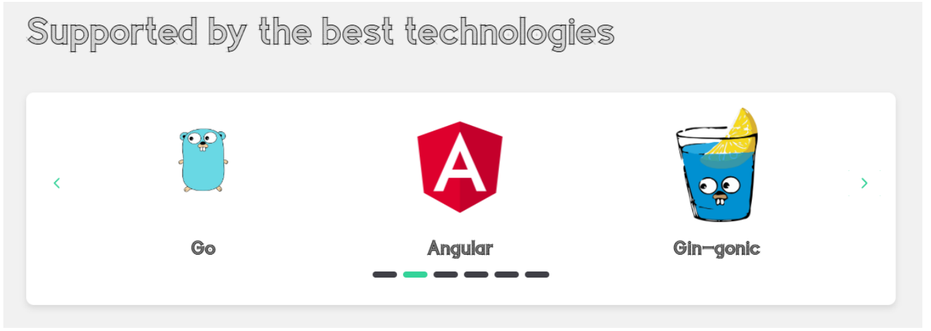
SafeHelmet

Also, the Dashboard **main page** and **sleek UI** aims to impress potential stakeholders and investors:

- ESP32 and Helmet **3D model showdown**
- “Technologies involved” **carousel**
- Main features **overview**









SafeHelmet

In depth – Web Dashboard features

SafeHelmet **Web Dashboard represents the frontend** for our entire system.

On this platform, a registered Boss can:



- Manage his worksites, opening new ones or closing them



- Hire workers and assign them to worksites



- View attendances' details containing information on workers' shifts



- View every reading's details and possible anomalies



- Export readings in PDF format



- **Monitor the weather** on the worksite in real time



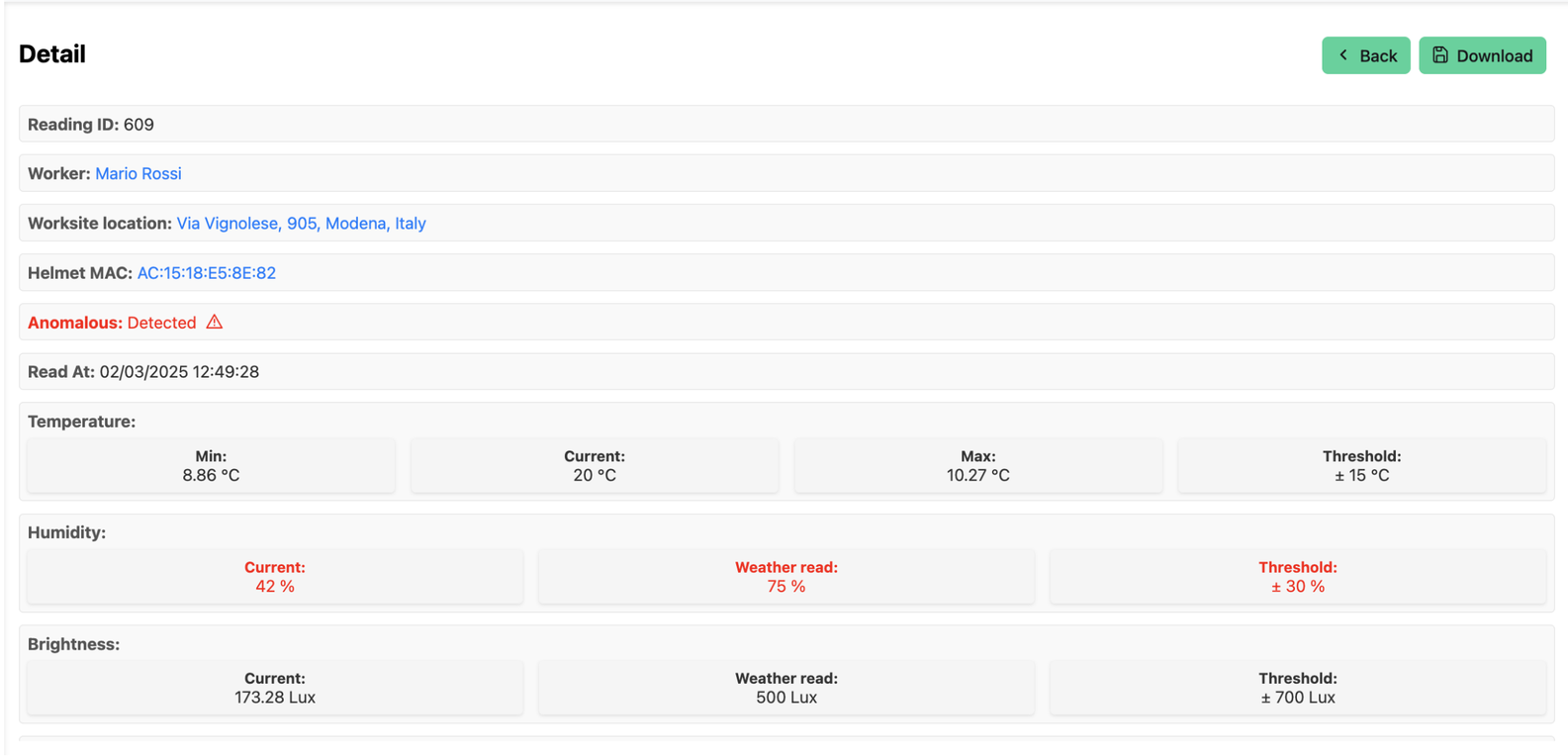
- **Register** new Smart Helmets in the system



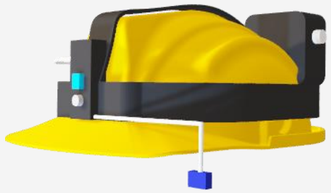
SafeHelmet

In depth – Readings and export

- Each reading detail is a snapshot of worker’s condition in a determinate instant of his shift.
- Each measure is visually compared to the relative static/dynamic threshold.
- Red color indicates an anomalous value.

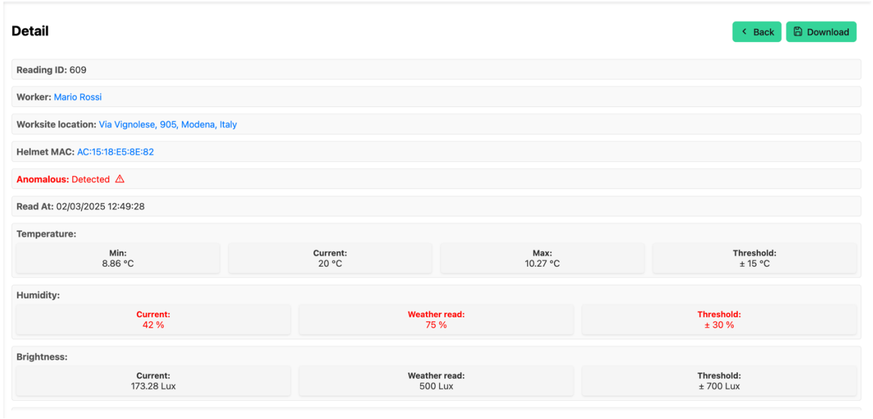


In depth – Readings and export



SafeHelmet

A reading can easily be exported in PDF to be later shared or saved locally.



SAFEHELMET Report Details

Reading ID: 609

Reading Date: 02/03/2025 12:49:28

Worker name: Mario Rossi

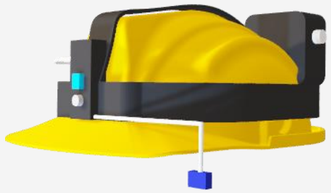
Worksite location: Via Vignolese, 905 Modena, Italy

Anomaly: Detected

Helmet Readings

Property	Value / Threshold
Temperature	-6.14 °C | 20 °C | 25.27 °C
Humidity	42 % | 100 %
Brightness	173.28 Lux | 1200 Lux
Methane Detected	No
Carbon Monoxide Detected	No
Smoke Detected	No
Welding Protection	Not plugged in
Gas Protection	Not plugged in
Incorrect Posture	0 %
Anomaly Detected	Yes
Mean acceleration value X-axis	0.32 m/s^2
Mean acceleration value Y-axis	0.11 m/s^2
Mean acceleration valuen Z-axis	10.43 m/s^2
Mean acceleration value magnitude	1.06 G
Value of acceleration standard deviation X-axis	0.11 m/s^2
Value of acceleration standard deviation Y-axis	0.08 m/s^2
Value of acceleration standard deviation Z-axis	0.08 m/s^2
Value of acceleration standard deviation magnitude	0.01 G
Max acceleration magnitude value	1.15 G

In depth – Difficulties and caveats



SafeHelmet

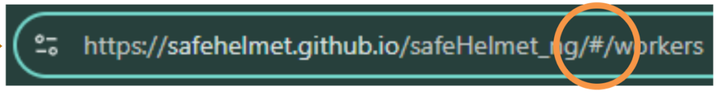
ISSUES FACED DURING DEVELOPMENT

PROPOSED SOLUTIONS

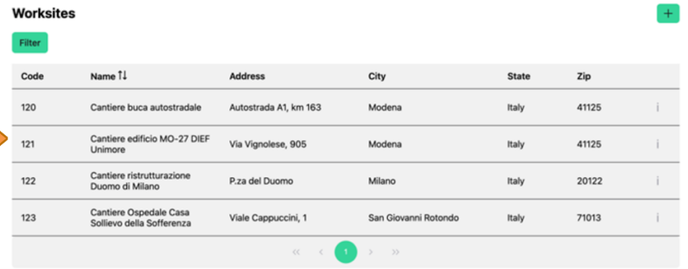
Single Page routing on deploy mode → Hashing

Angular secrets and environment files → GitHub Pages

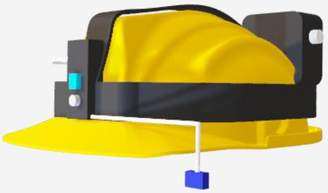
Develop a polished and intuitive UX → Minimalistic Design







Web dashboard – Pros and Cons



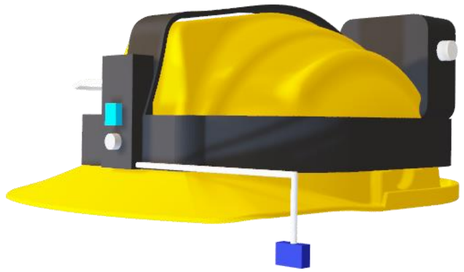
SafeHelmet

- **Angular**:
  - ✅ Single page → Fast rendering of the site
  - ✅ Enterprise oriented, with high scalability and built-in routing tools
  - ❌ Steep learning curve and quite hard to mantain for newcomers
- **Github pages** hosting:
  - ✅ CI/CD Integration with GitHub Actions, that automates compiling and deploying phase for every new build
  - ✅ Free hosting
  - ❌ Server configuration is not editable, forcing to adopt client-side solutions
- **PrimeNg**:
  - ✅ Rich and easy to use collection of UI components
  - ✅ Seamless integration with Angular
  - ❌ Limited to their own layouts and objects

Future works and improvements

Now what?

With a project so articulated, improvement possibilities are almost endless!



Smart Helmet:

- Prototype has no **battery** nor on/off switch
- **Miniaturization** of the entire module
- Improvement in **sensors' quality**
- **ML** model training and embedding for crash detection and more
- Change ESP32 firmware to a more performant one

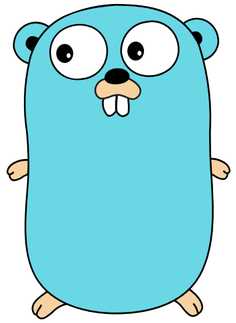


SafeHelmet Mobile:

- Improve **UI and features**
- **Multiplatform** support (Android and iOS)
- Data and login persistency in-app
- Completely bypass the need of a mobile application

Now what?

With a project so articulated, improvement possibilities are almost endless!





Backend system and Database:

- Upgrade OpenWeather API plan
- Improve db structure
- Weather coordinates grouping

Web Dashboard:

- **Have an enterprise safe SSO**
- Expand the showroom
- Add more features in the Admin area

Thanks for your attention

For inquiries:
paridestomeo@gmail.com,
mussini.simone01@outlook.com

3D model: Chiara Valli