# Tango Autumn 2020 Status webinar

Elettra Sincrotrone Trieste

Elettra
Sincrotrone
Trieste

# PUMA

# Reliable, secure, scalable and user-oriented design of a multi platform framework based on the most advanced stage of web technologies

Giacomo Strangolino
Lucio Zambon *

* inspired by an idea of Alessio I. Bogani

# HISTORY OF PUMA

- 2006 - Canone - Python server, drag and drop designer
- 2016 - ElettrApp - Responsive, Cordova, Jquery, Bootstrap
- 2017 - PWMA - C++ server + Websocket, React, React Native
- 2020 - PUMA - NChan server + SSE, designer tool for responsive pages

Elettra
Sincrotrone
Trieste

# DESIGN RATIONALE

## SECTION I

## THE SERVICE

# DESIGN RATIONALE (I)

## 1. RELIABLE


<u>The system must work</u>

- From any place, time, platform
- Always, regardless the number of clients
- Always, included when part of the system is unavailable
- Included when network performance is slow (or subject to charges!)

Performance independent of the number of clients

# DESIGN RATIONALE (I)

## 2. SECURE

**SSE! :-)**
(slide 11)

1) From external attacks (DoS, intrusion)
2) From ill designed clients flooding the service
3) Protect the downstream Control System Engine (Tango, EPICS)

- Network architecture
- Security-oriented **OS**es (BSD!?)
- Framework design, database and code

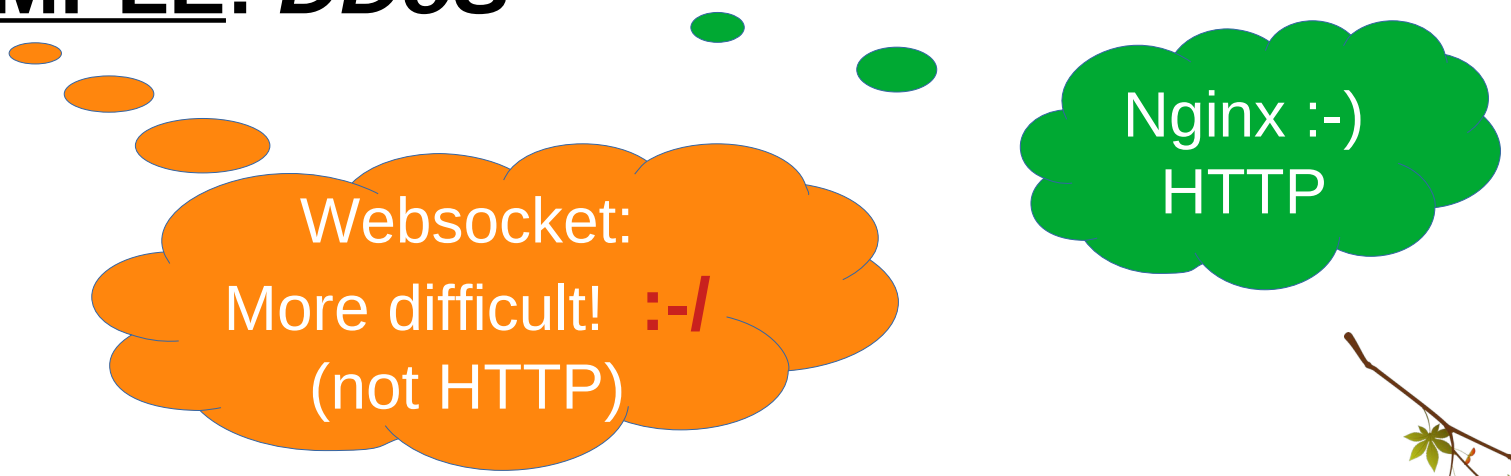
- No service performance decay
- Hamper unruly clients

# DESIGN RATIONALE (I)

## 2. SECURE (II)

**GOAL** secure enough as to avoid VPN or other additional client-side configuration hindering *usability*, especially on mobile devices

**EXAMPLE**: ***DDoS***

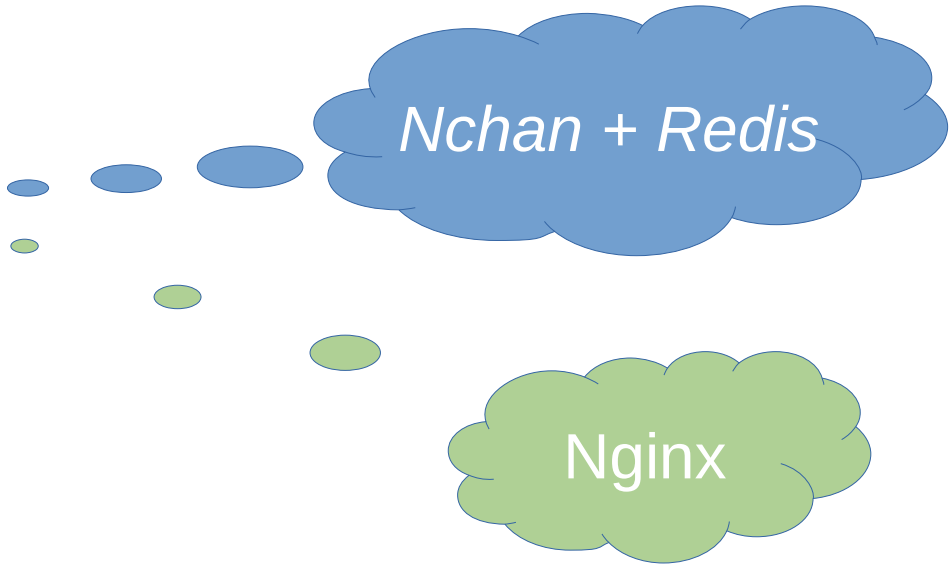


# DESIGN RATIONALE (I)

## 3. SCALABILITY

A good infrastructure must be designed with scalability in mind. It reinforces security and reliability.

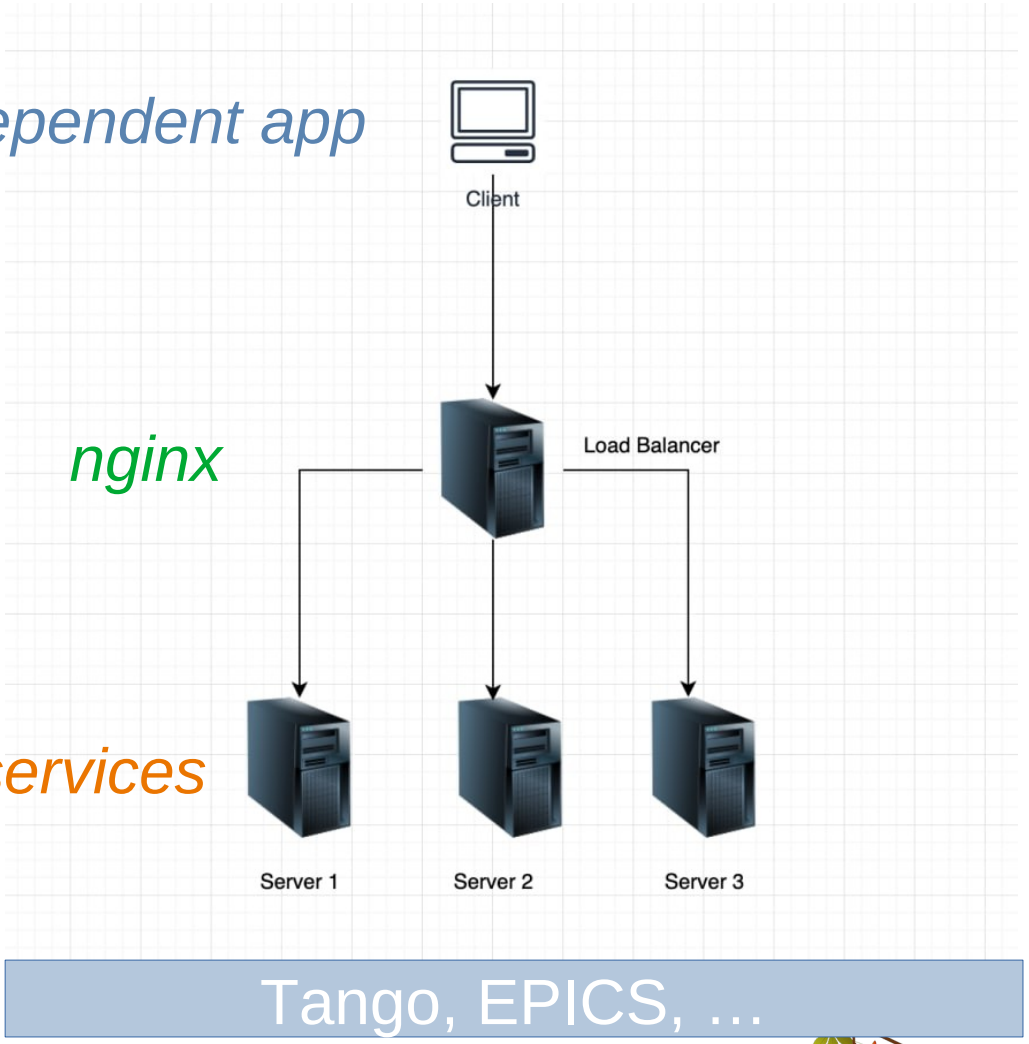
1) Horizontal
2) Vertical



# DESIGN RATIONALE (I)

## 3. SCALABILITY (II)

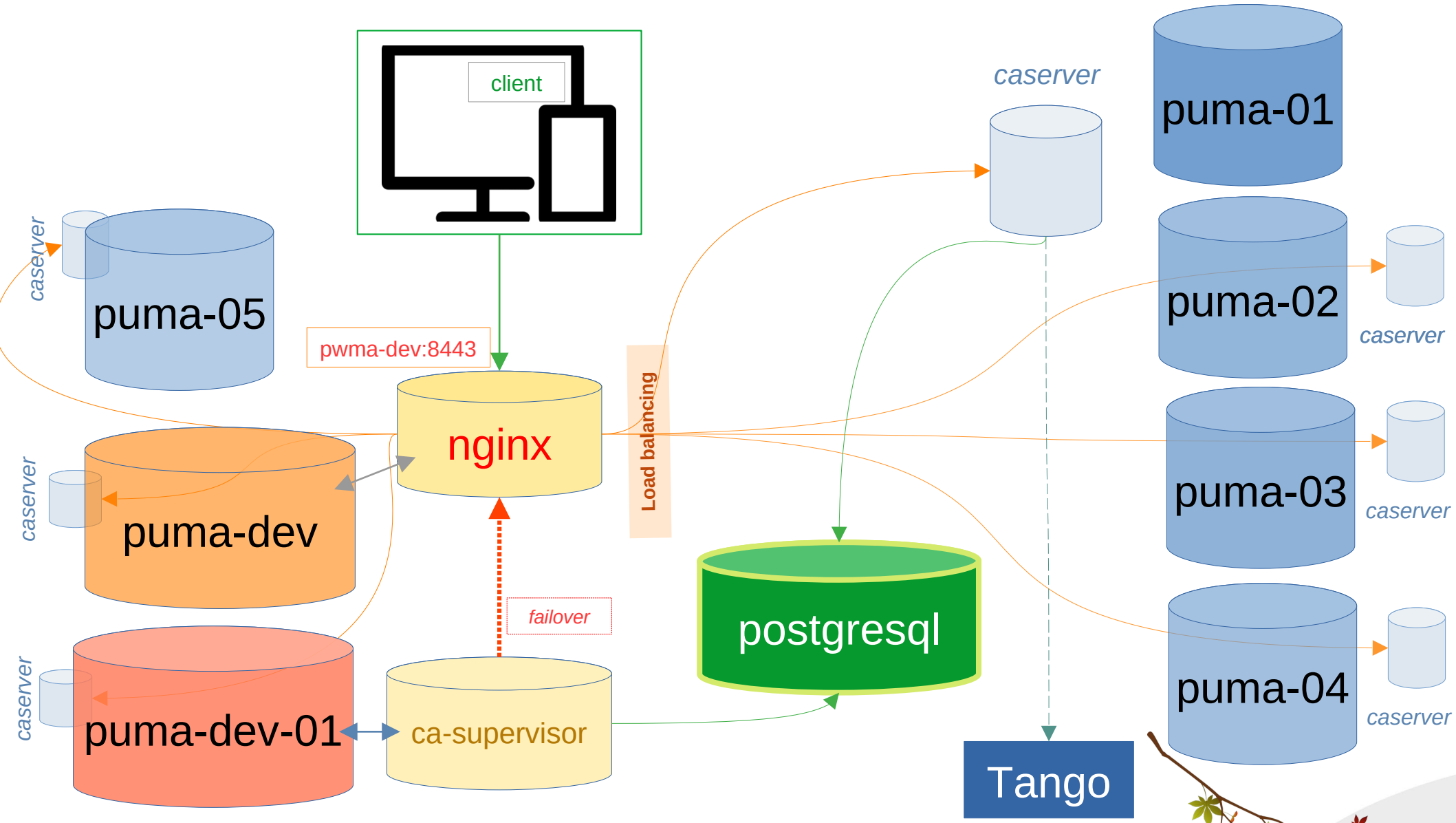
*Platform independent app*




*nginx*

*PUMA (canone3) services*

# DESIGN RATIONALE (I) - DEPLOYMENT



Elettra Sincrotrone Trieste

# DESIGN RATIONALE (I)

## 4. GENERIC API

Must serve

1) The web
2) Mobile applications
3) Desktop applications *

* Cumbia libs already support the API so that any Qt application can be instructed to rely on either the native control system engine or the HTTP service at runtime

Elettra
Sincrotrone
Trieste

# DESIGN RATIONALE
# (I – the service)

Nginx + nchan + http/SSE * = ?

* inspired by an intuition of Alessio I. Bogani

Elettra Sincrotrone Trieste

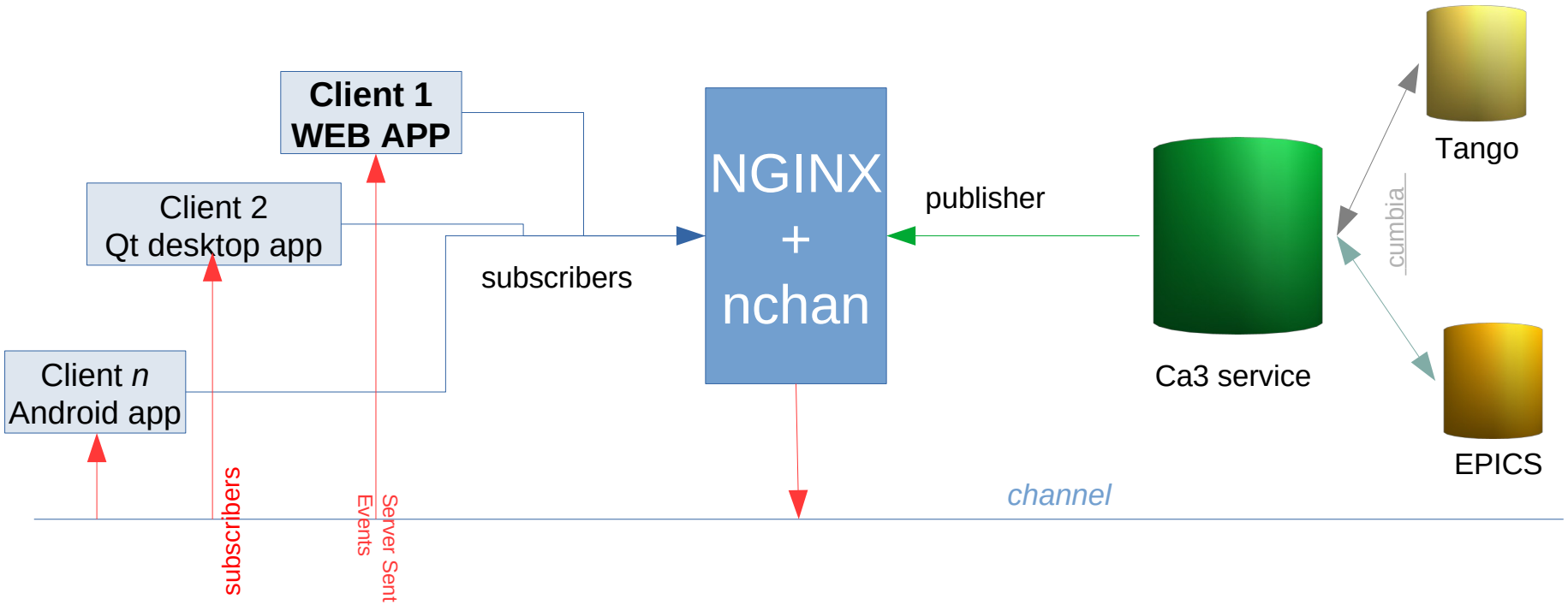
# Nginx + nchan + http/SSE = ?

- ✓ **NGINX**: high performance load balancer, web server and reverse proxy *https://www.nginx.com/*
- ✓ **NCHAN**: flexible *pubsub* for the modern web
- ✓ **SSE**: a server *push* technology: a client receives automatic updates from a server via HTTP connection

**=**

A scalable, secure, efficient service with multiplexing for web, mobile and desktop applications

# Nginx + nchan + http/SSE = ?

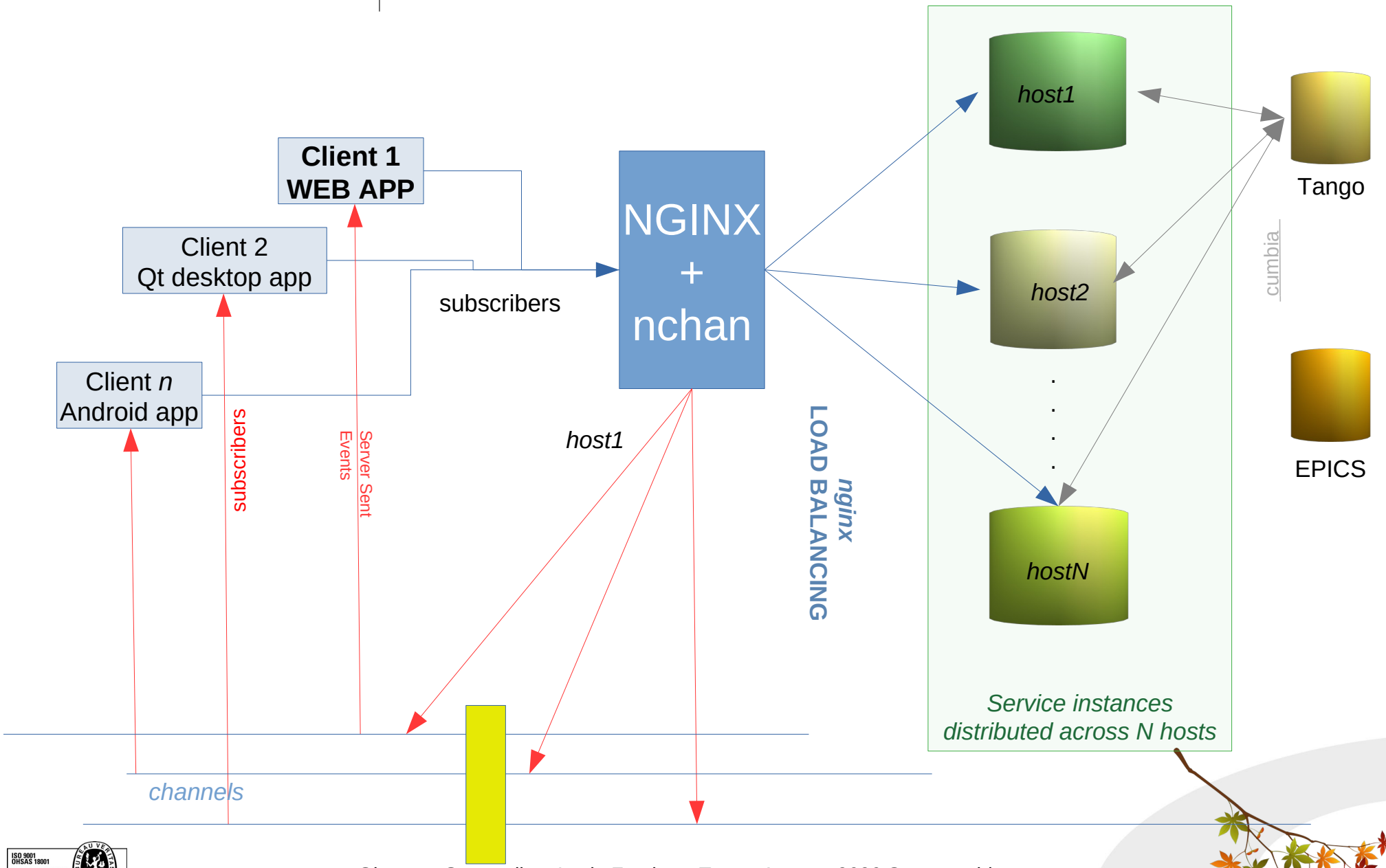



**Additionally**

- Synchronous readings
- Synch database property fetch
- Authenticated synchronous writings

**Multiplexing**: *n* clients reading *x* ⇒ *1* reader to the native engine

# Nginx + nchan – scalability and load balancing






**Nchan + Redis = additional scalability**

Elettra Sincrotrone Trieste

# Nginx + Nchan + Redis

Redis (Remote Dictionary Server) is an in-memory data structure project implementing a distributed, in-memory key–value database with optional durability.

✔ Redis can be used to add data persistence and horizontal scalability, failover and high availability to a Nchan setup.

✔ *Redis Cluster* provides a way to run a Redis installation where data is automatically sharded across multiple Redis nodes.

**Nchan + Redis**

✔ add scalability via sharding channels among cluster nodes.
✔ *Redis cluster* provides automatic failover, high availability,
✔ *Redis cluster* eliminates the single point of failure of one shared Redis server

# DESIGN RATIONALE

## SECTION II

## THE CLIENTS

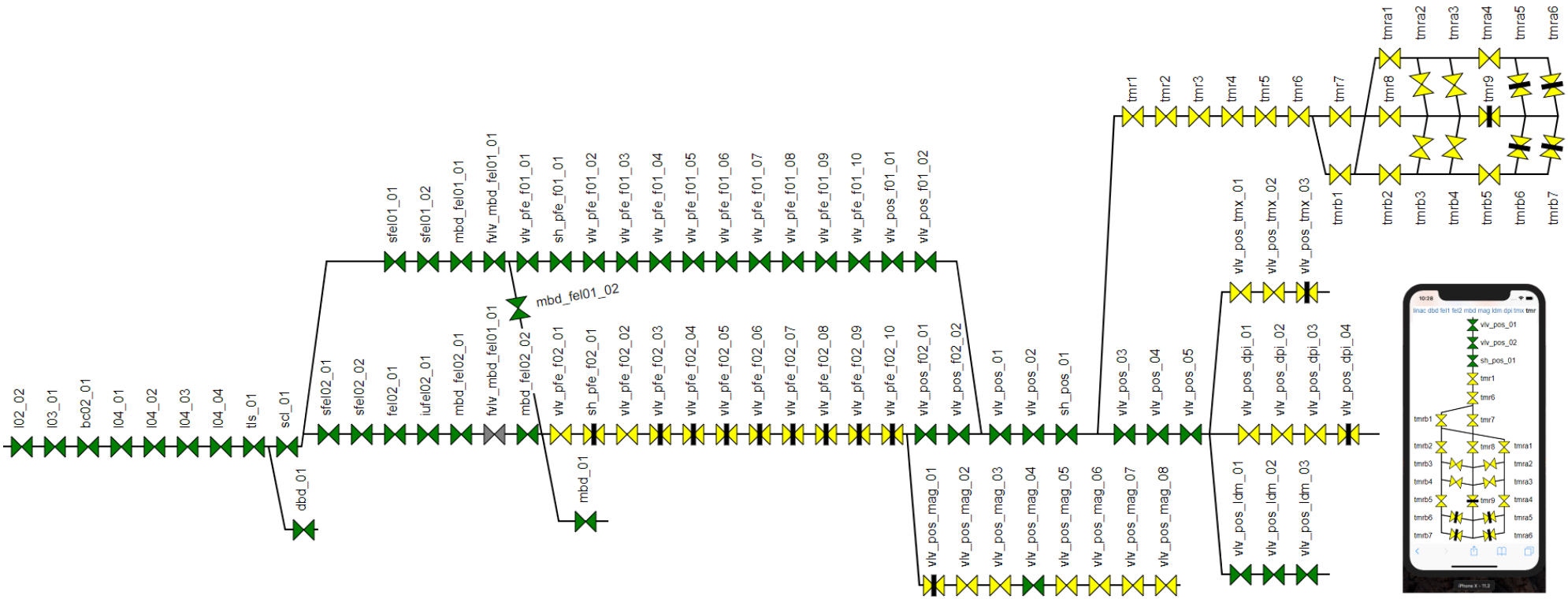
# DESIGN RATIONALE (I)

## 1. RESPONSIVE

The system must be adaptive

- The application adapts to every device automatically
- On portable devices, screen rotation must not frustrate but rather exploited to increase usability
- Certain actions (especially commands and write operations) are logged

# DESIGN RATIONALE (II)



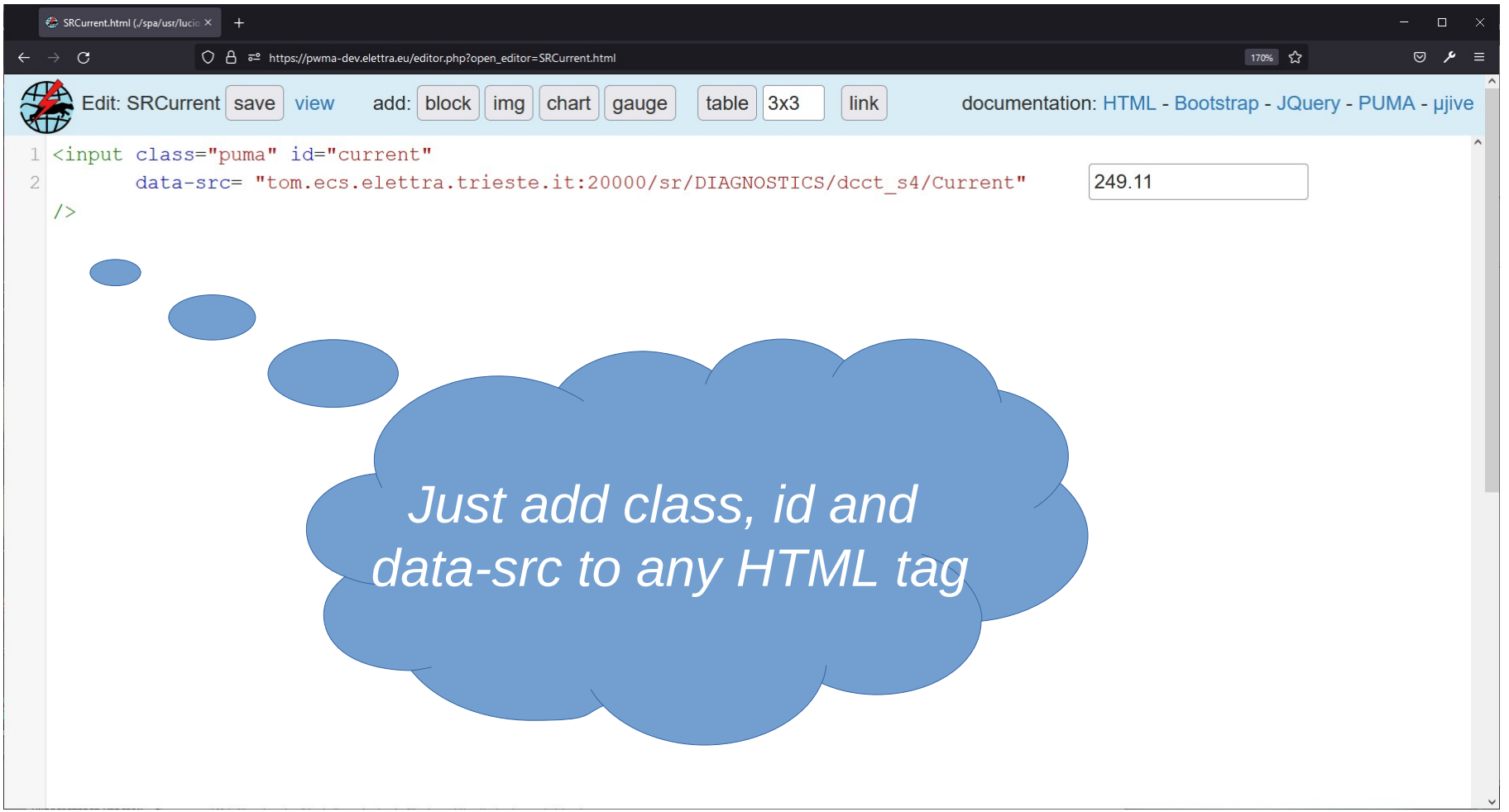
Elettra Sincrotrone Trieste

# DESIGN RATIONALE (III)

## 3. DESIGNER

- The designer produces *responsive* pages using flexbox which is CSS3 standard
- pages are saved in JSON format in a DB
- JSON files are interpreted by a web page and by an app
- saved pages can be used as modules embedded in new pages
- Advanced users can use a web HTML text editor integrated with an instant preview (triggered by keyup event)

# DESIGN RATIONALE (IV)



# DESIGN RATIONALE (V)

## 4. MULTI PLATFORM

We deem *Telegram* a perfect example of multi platform application:

1. A *native* app on all mobile devices
2. A *desktop* applications for all platforms (even *FreeBSD*)
3. A *web* interface
4. Has a simple and open API to create clients, bots, …
5. Efficiency, security and privacy centered

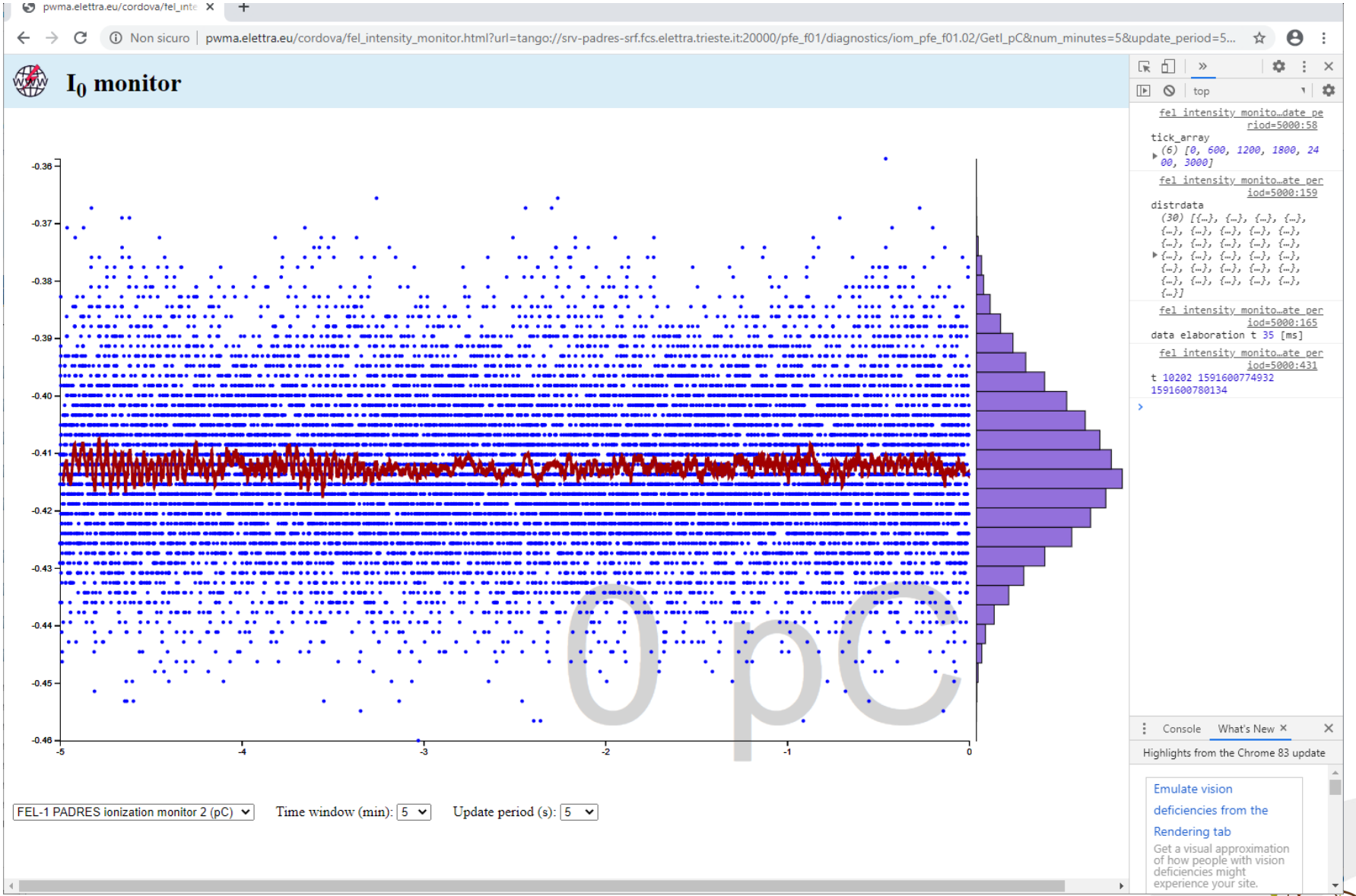
These traits have continuously inspired the development of *canone3*

# DESIGN RATIONALE

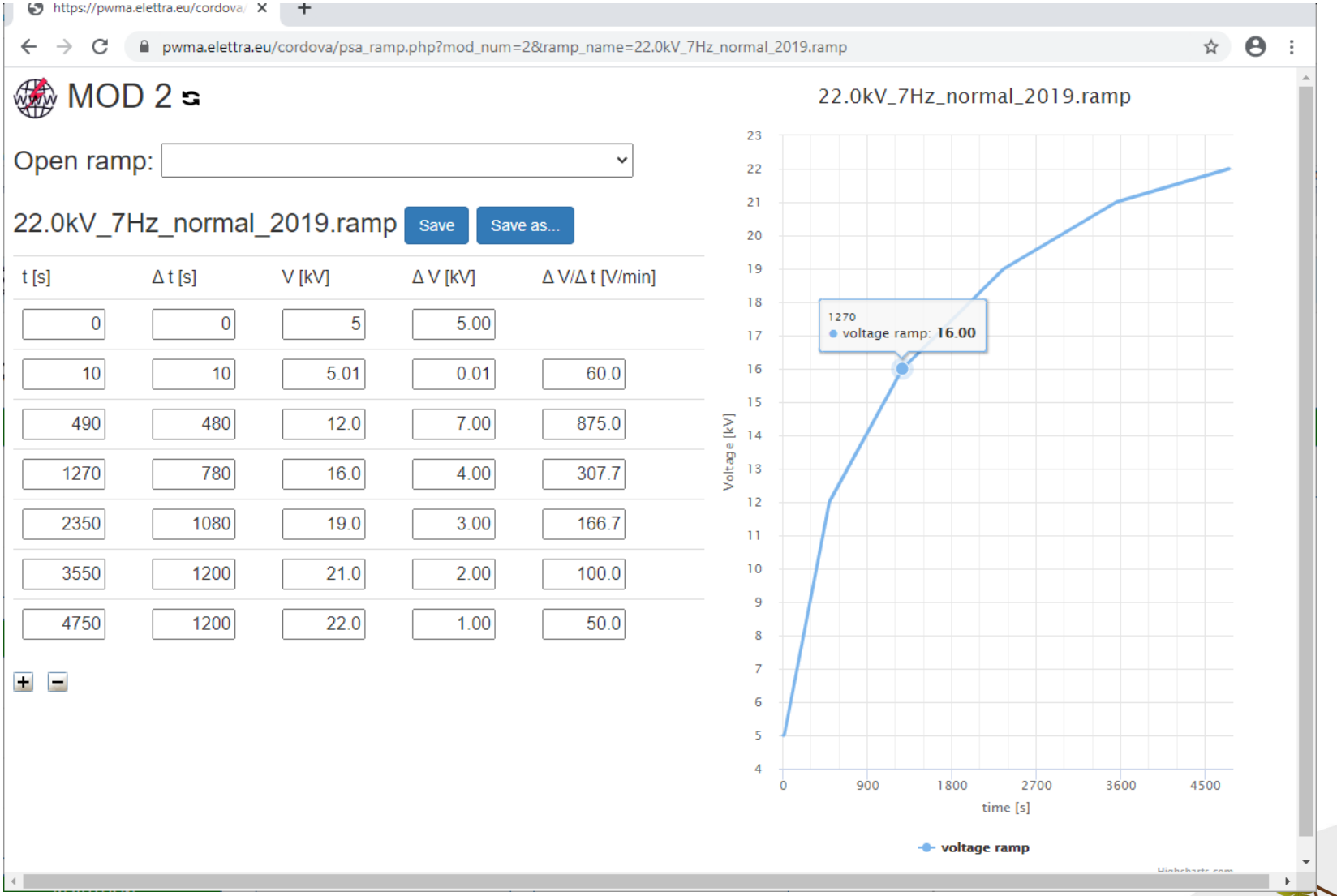
## SECTION III

## SOME WORKING EXAMPLES

# Clients – Web Apps (I)

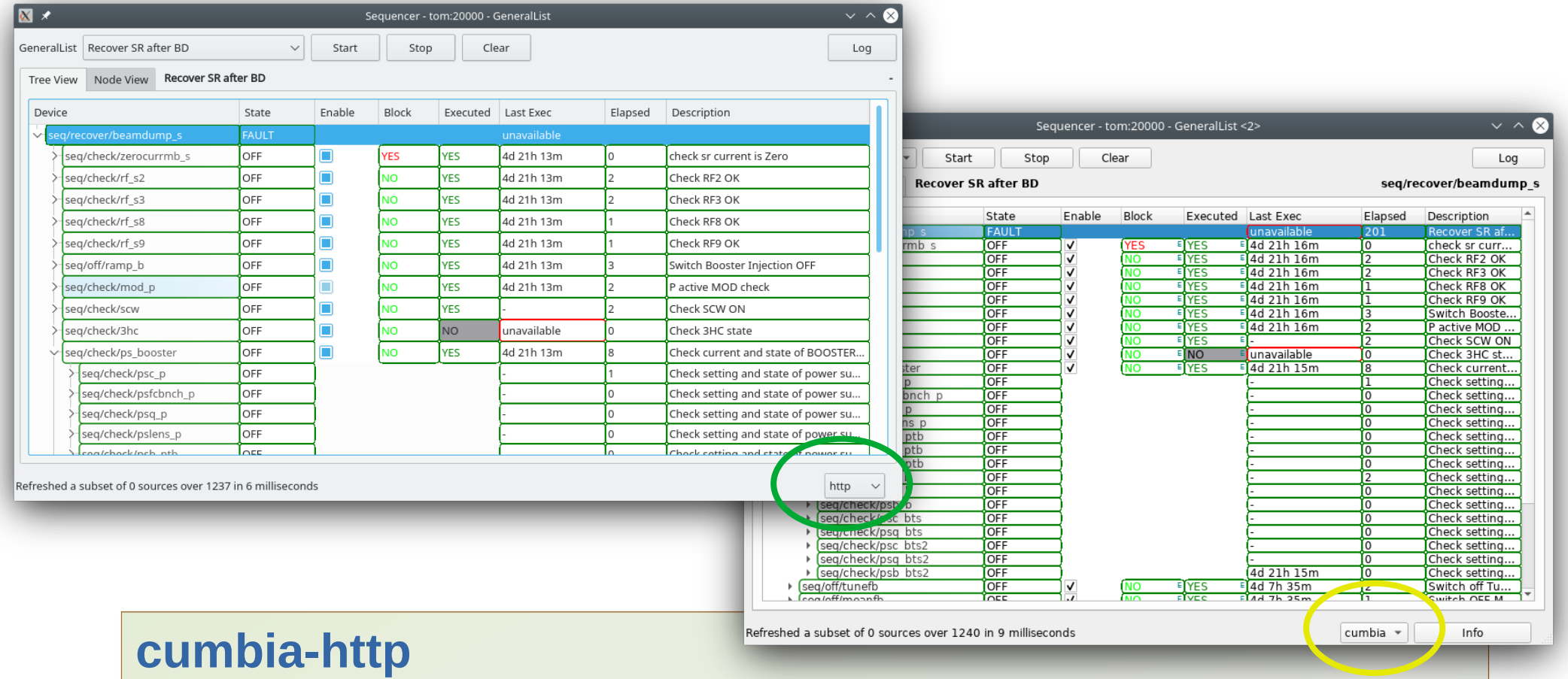


# Clients – Web Apps (II)



# Clients – Web Apps (III)

# Clients – Qt desktop apps (I)




## cumbia-http

- ✗ the same app is *designed* and *run* transparently regardless the engine in use (*native Tango* or *http/SSE*)
- ✗ No *engine-specific* coding
- ✗ Run with the same command line!
- ✗ A virtual machine run at home can reproduce exactly the control room desktop

# Clients – Control Room apps

Approaches to running control room applications remotely

## Now

✗ Control room virtual machines run on the server side

✗ Although optimized, a full graphical session is streamed, with a strong impact on *bandwidth* and *battery life* of portable devices. If the former today may not be considered a critical limit (however, estimate the load of a server streaming to hundreds of clients), the latter is indeed a precious resource for portable devices and laptops. *The only relevant piece of information is* ***data****.*

## Then

✗ Control room virtual machines run on the client

✗ The user runs only the apps he needs

✗ The apps run natively (i.e. the fastest possible on the client device), look exactly as in the control room (they are the same) without lags

✗ *Only data through the network*, for native and web apps alike

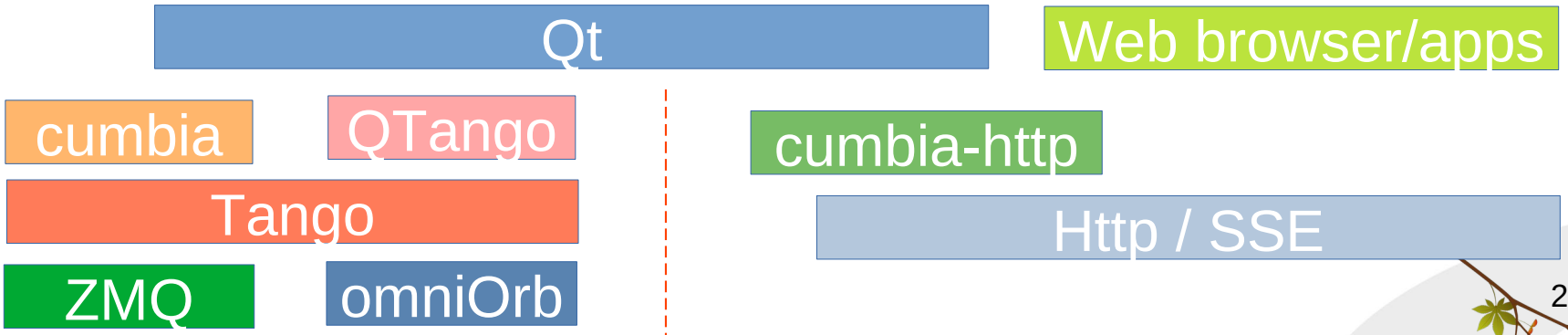
# Clients – Control Room apps

## Approaches to running control room applications remotely (II)

**Additionally**

- ✗ Either *virtual* or *physical* machines do not need Tango + dependencies
- ✗ They need only a web browser and Qt for native apps
- ✗ Events only
- ✗ Tango control system potentially *highly relieved* (remember: *N* clients reading *x*, 1 read to the control system)

**+** all benefits discussed in the *design rationale* section



Elettra
Sincrotrone
Trieste

# Thank you!

Elettra
Sincrotrone
Trieste

www.elettra.eu