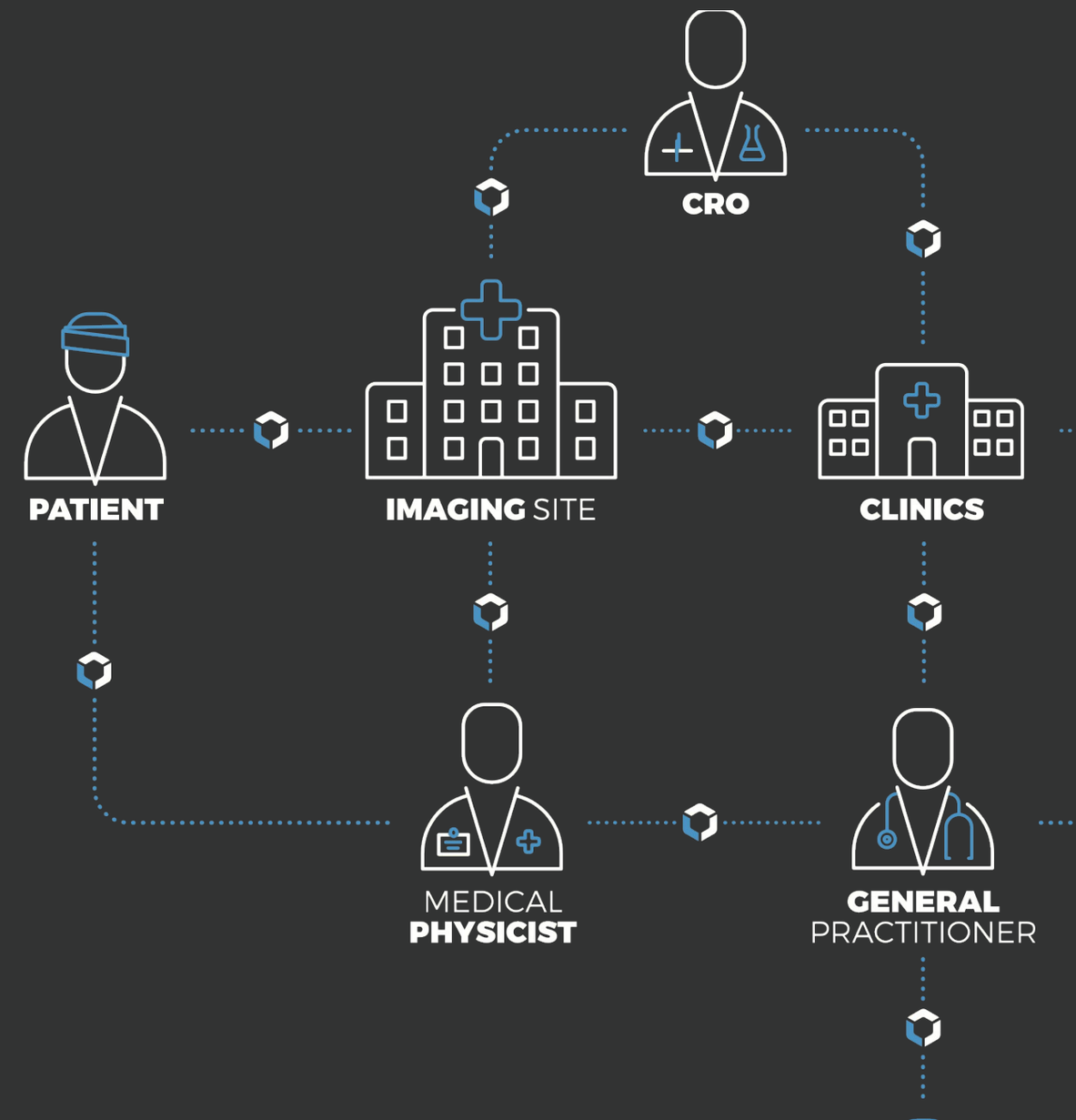
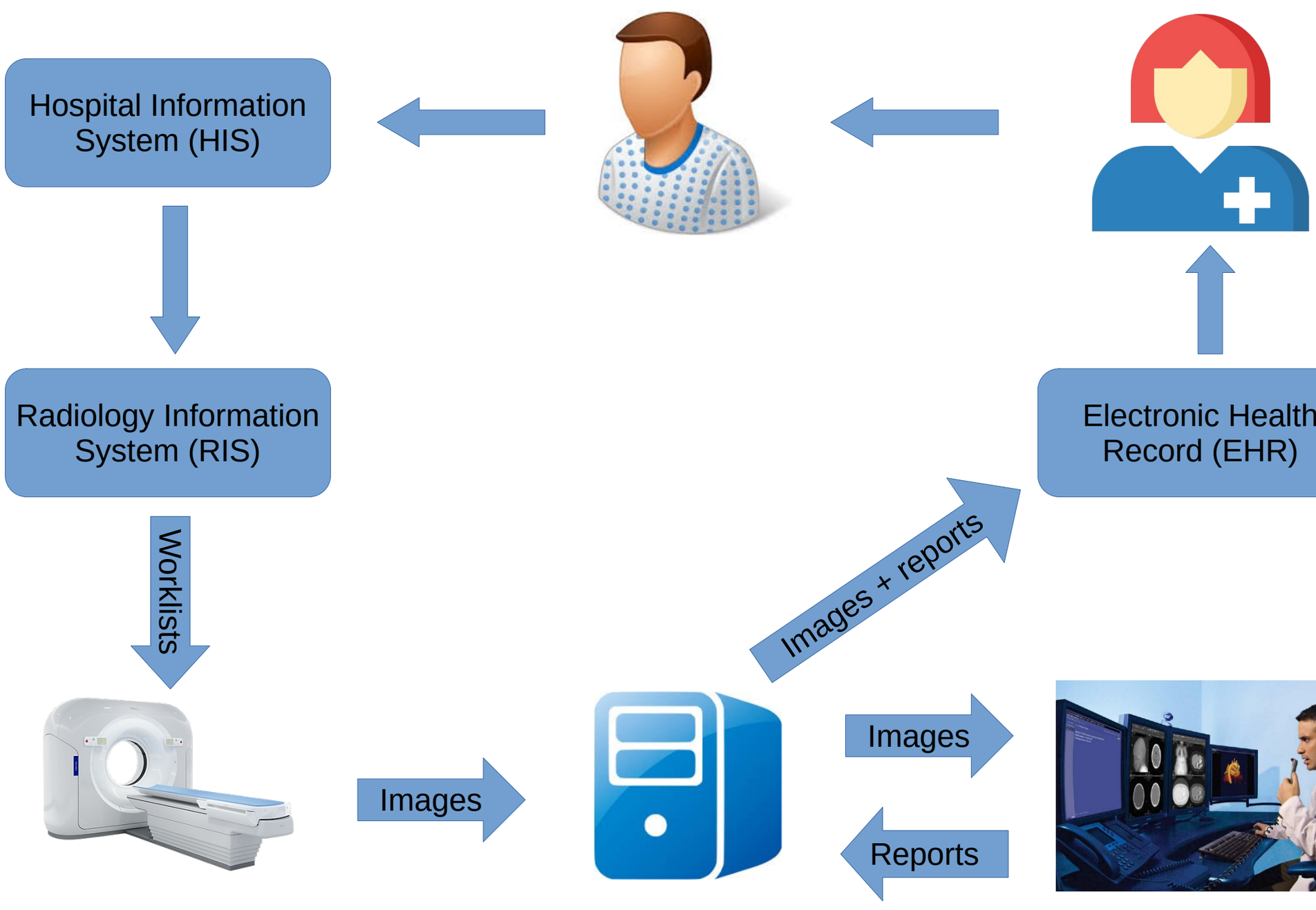
ORTHANC

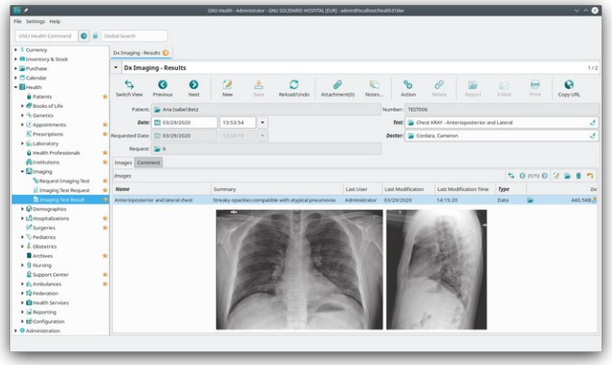
Using WebAssembly to render medical images

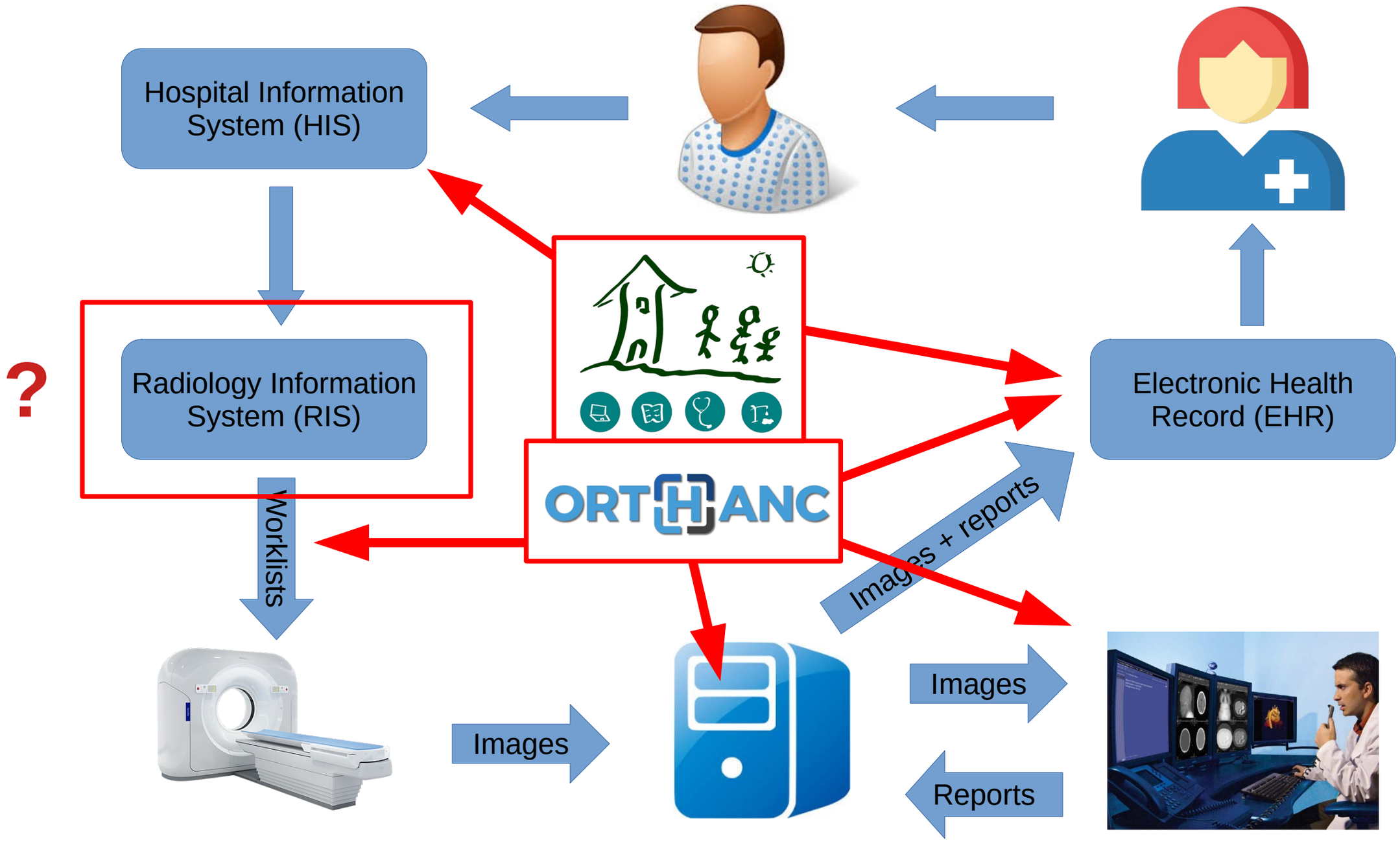


The radiology workflow in hospitals

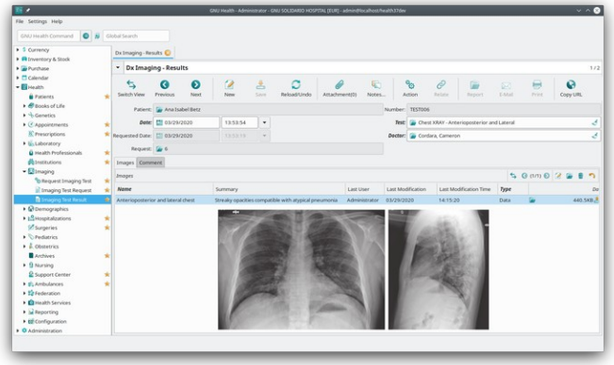


Libre Software?





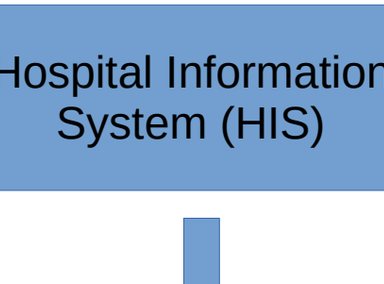
This talk: Viewers



Hospital Information System (HIS)

Radiology Information System (RIS)

Worklists



Images

ORTHANC

Images

Picture Archiving and Communication System (PACS)

Images + reports

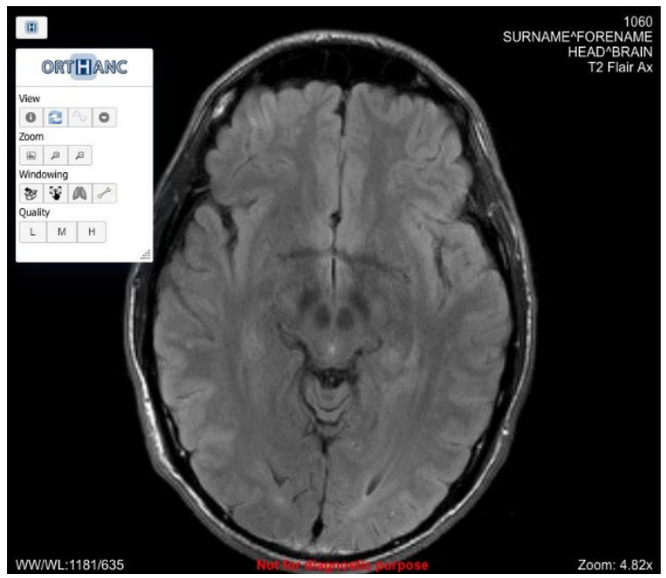
Images

Reports

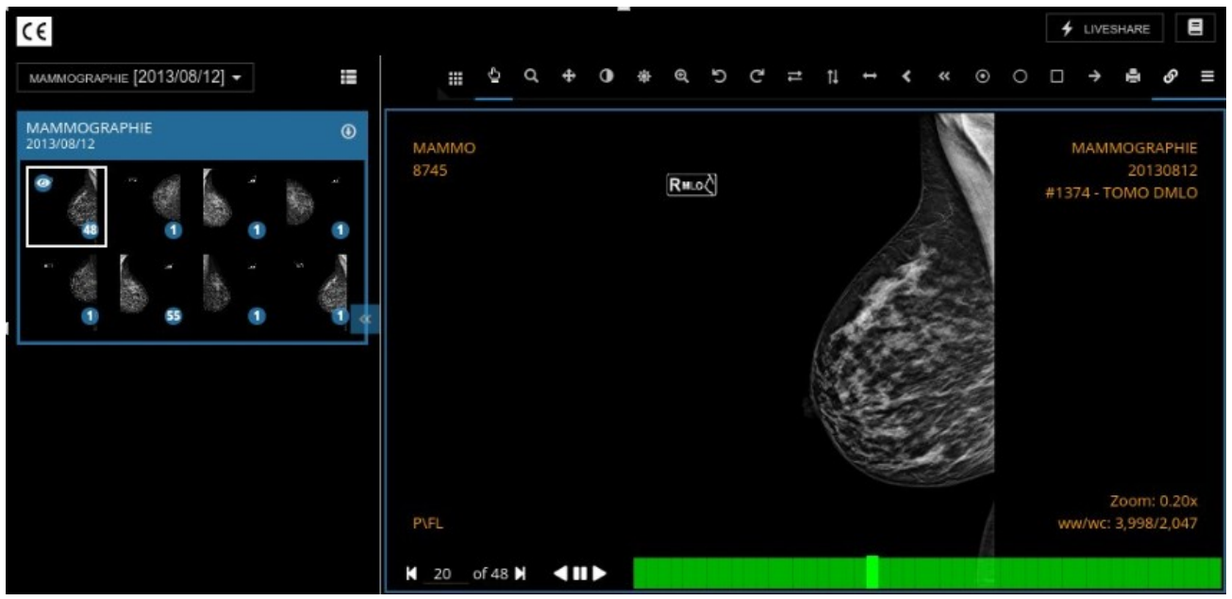
Electronic Health Record (EHR)

Libre viewers for Orthanc

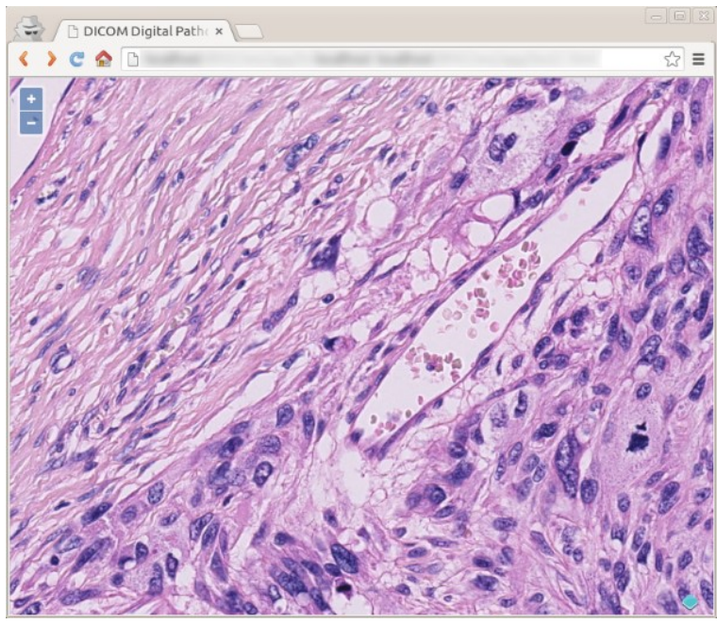
Basic reviewing



Advanced teleradiology (Osimis Web viewer)



Whole-slide imaging



External, Web:

Open Health Imaging Foundation

DWV, ...

External, desktop:

Horos (Apple only)

3DSlicer

Ginkgo CADx
Open Core DICOM Viewer + Dicomizer + CADx

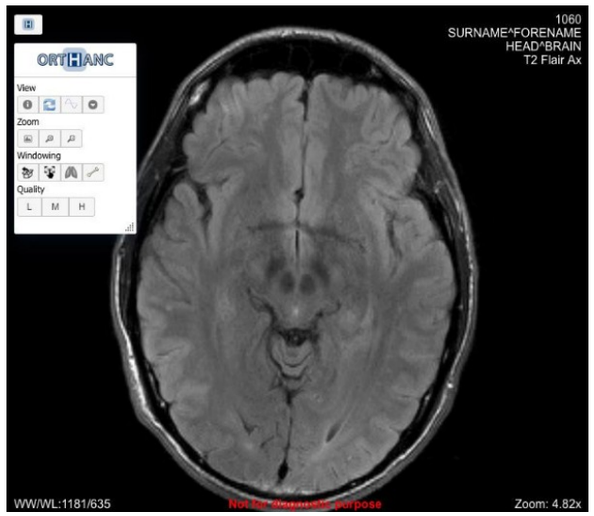
medInria

Aeksulap, ...

Two fully separate worlds

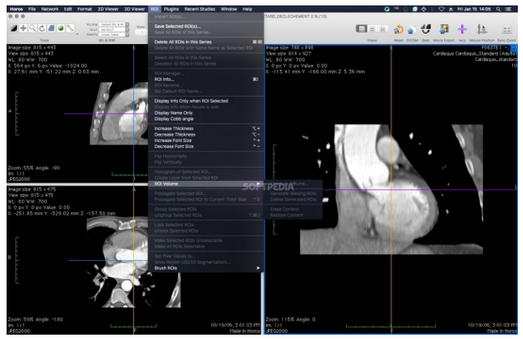
Web applications (teleradiology)

JavaScript + Cornerstone library



Desktop software (clinical radiology)

C++ +



The problems

No code reuse between Web and Desktop!
=> Fully redundant developments, separate teams

Desktop teams: How can I relocate some part of my software as a Web application for easy, fast delivery and to avoid the high cost of maintaining different ports and installers?

Web teams: How to use existing libraries for DICOM?

Side note: The AGPL license is essential in such a context!

Question

Is it possible to run C++ client-side in Web browsers?

Yes!

mozilla

Microsoft
Google
intel

redhat

What is WebAssembly?

- **Bytecode for the Web**
- Open standard maintained by the W3C
- Official recommendation since 2019-12-05
- Precursors: Java applets, PNaCl from Google, asm.js from Mozilla…
- Supported by all the major Web browsers (including proprietary ones)

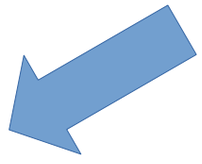
Official “C++ to WebAssembly” compiler

Hello, world! (1/2)

Bullseye or Sid/Unstable

```
$ sudo apt install emscripten
```

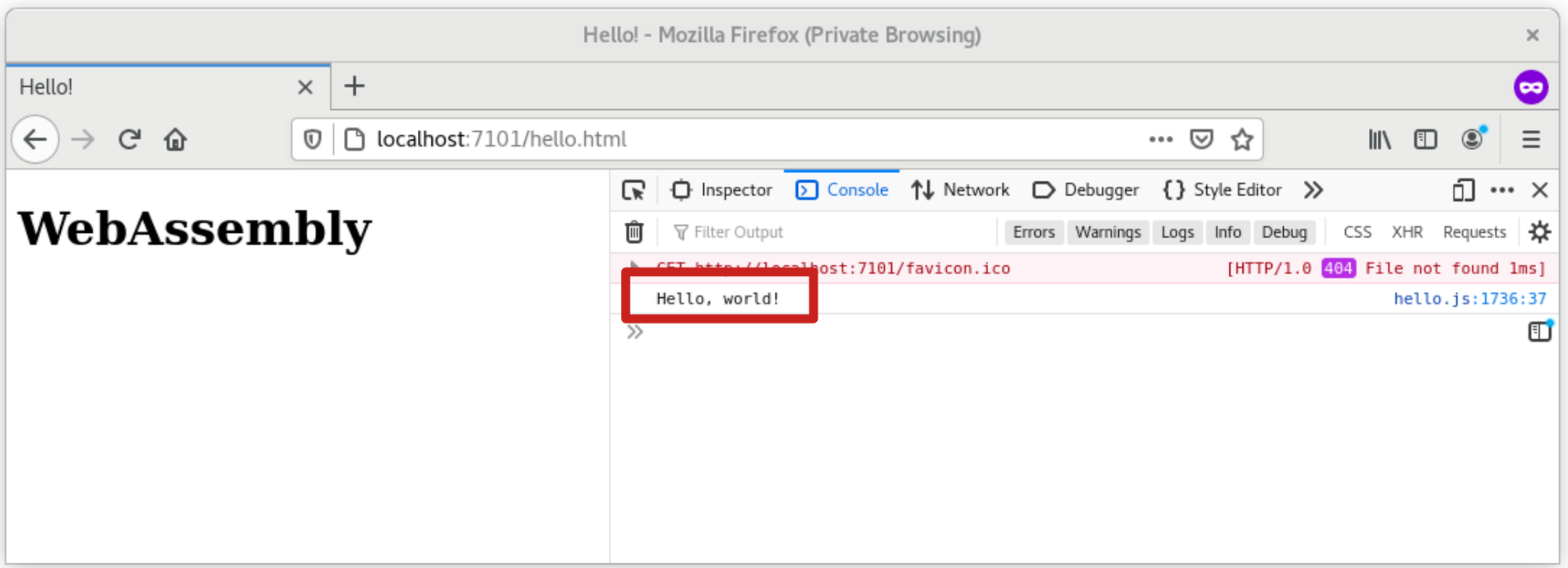
```c
#include <stdio.h>

int main()
{
  printf("Hello, world!\n");
  return 0;
}
```



```
$ em++ ./hello.cpp -o hello.js
$ ls
hello.cpp    => C++ source code
hello.js     => JavaScript wrapper
hello.wasm   => WebAssembly bytecode
```

Hello, world! (2/2)

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello!</title>
  </head>
  <body>
    <h1>WebAssembly</h1>
    <script src="hello.js" async></script>
  </body>
</html>
```



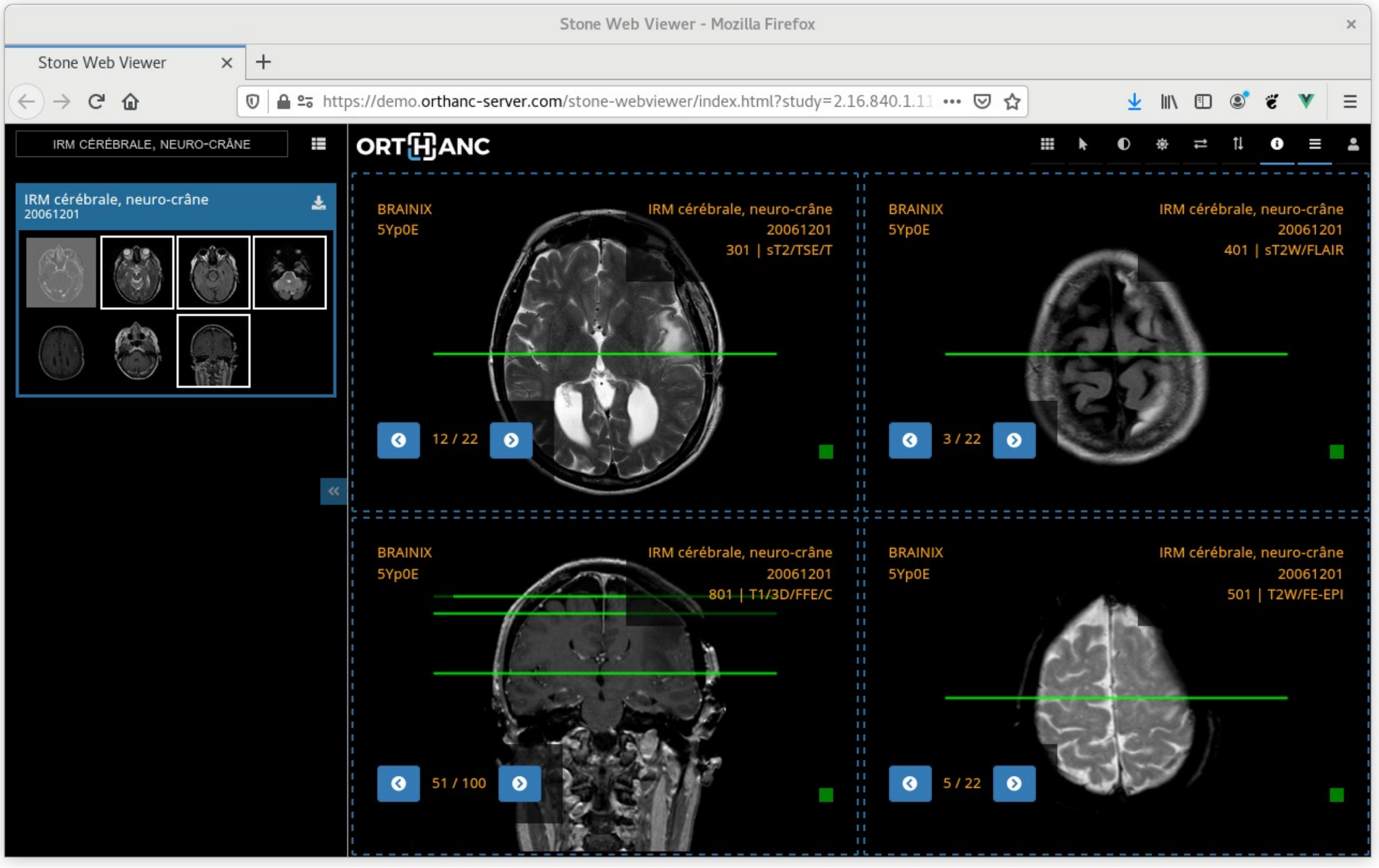
Stone of Orthanc

- **Lightweight, cross-platform C++ library to render medical images** (cf. VTK)
- Part of the Orthanc ecosystem
- Compatible with **WebAssembly**
- Compatible with **GUI libraries** (SDL, Qt…)
- Building block for the **Stone Web viewer**
- Obviously, libre software!

More features:

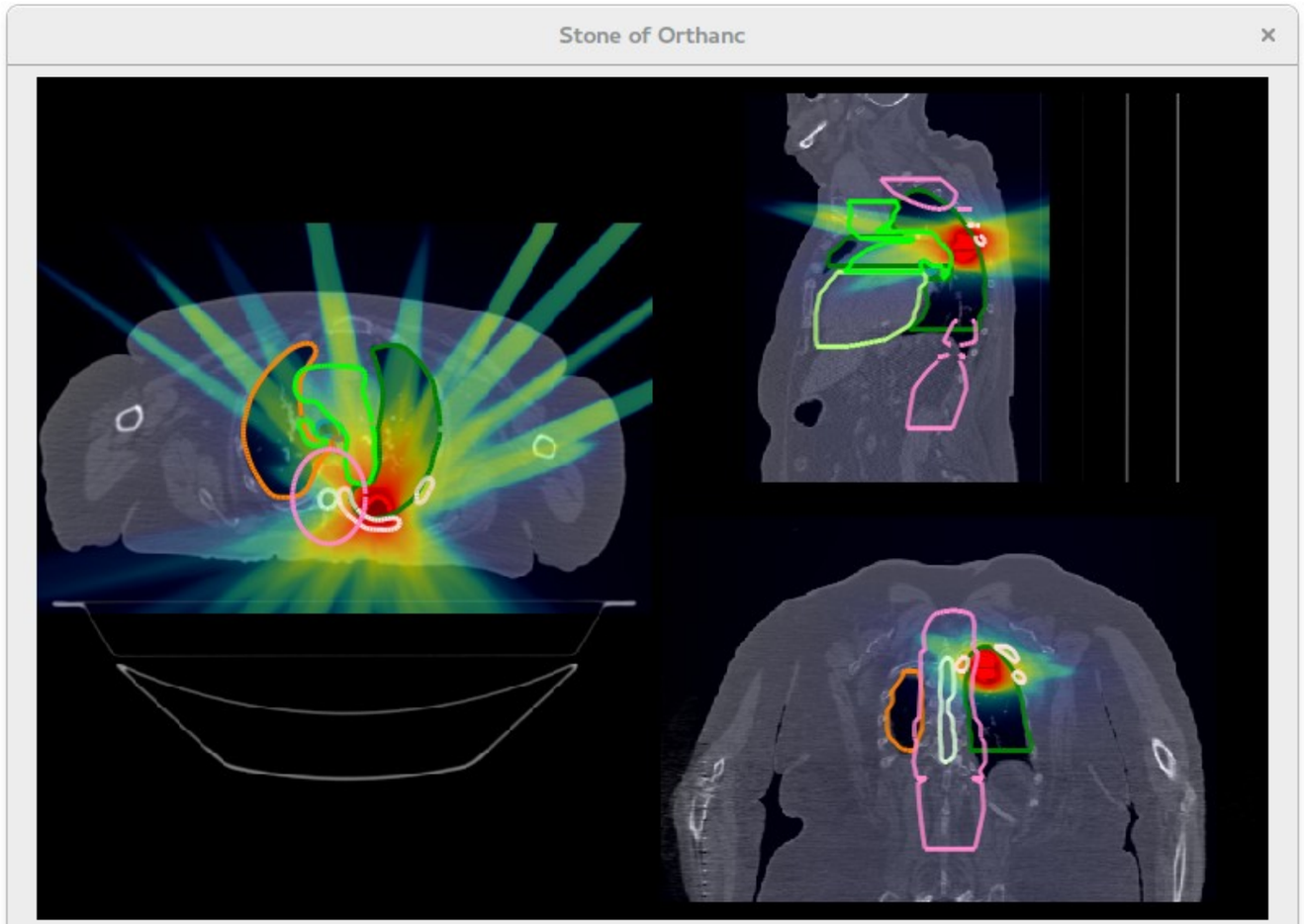
- *2D hardware acceleration (WebGL/OpenGL)*
- *Primitives for DICOM (parsing and DICOMweb)*
- *Built-in support of 3D volumes (MPR, volume reslicing)*
- *Support of oncology: PET-CT fusion, doses, contours…*

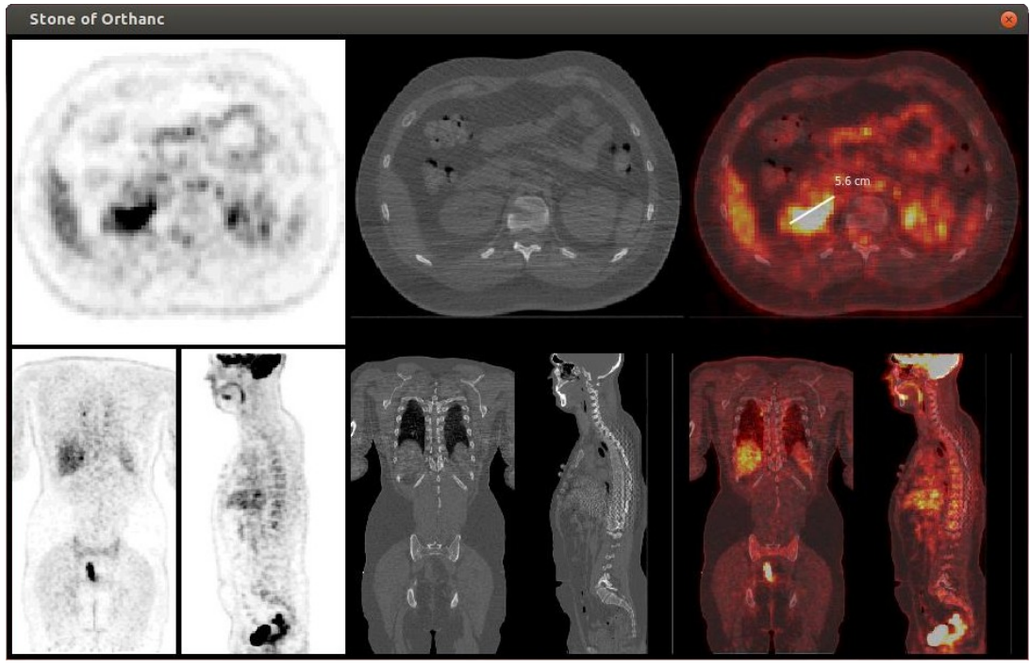
Stone Web viewer

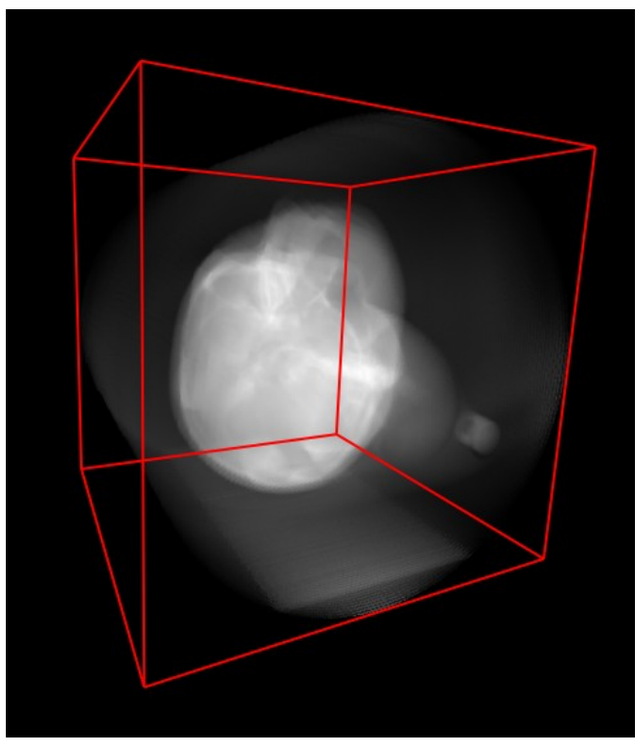


- Reuse the layout of the Osimis Web viewer (now unmaintained)
- Online demo: `https://demo.orthanc-server.com/`
- Nightly build in Docker image: `jodogne/orthanc-plugins:1.8.0`

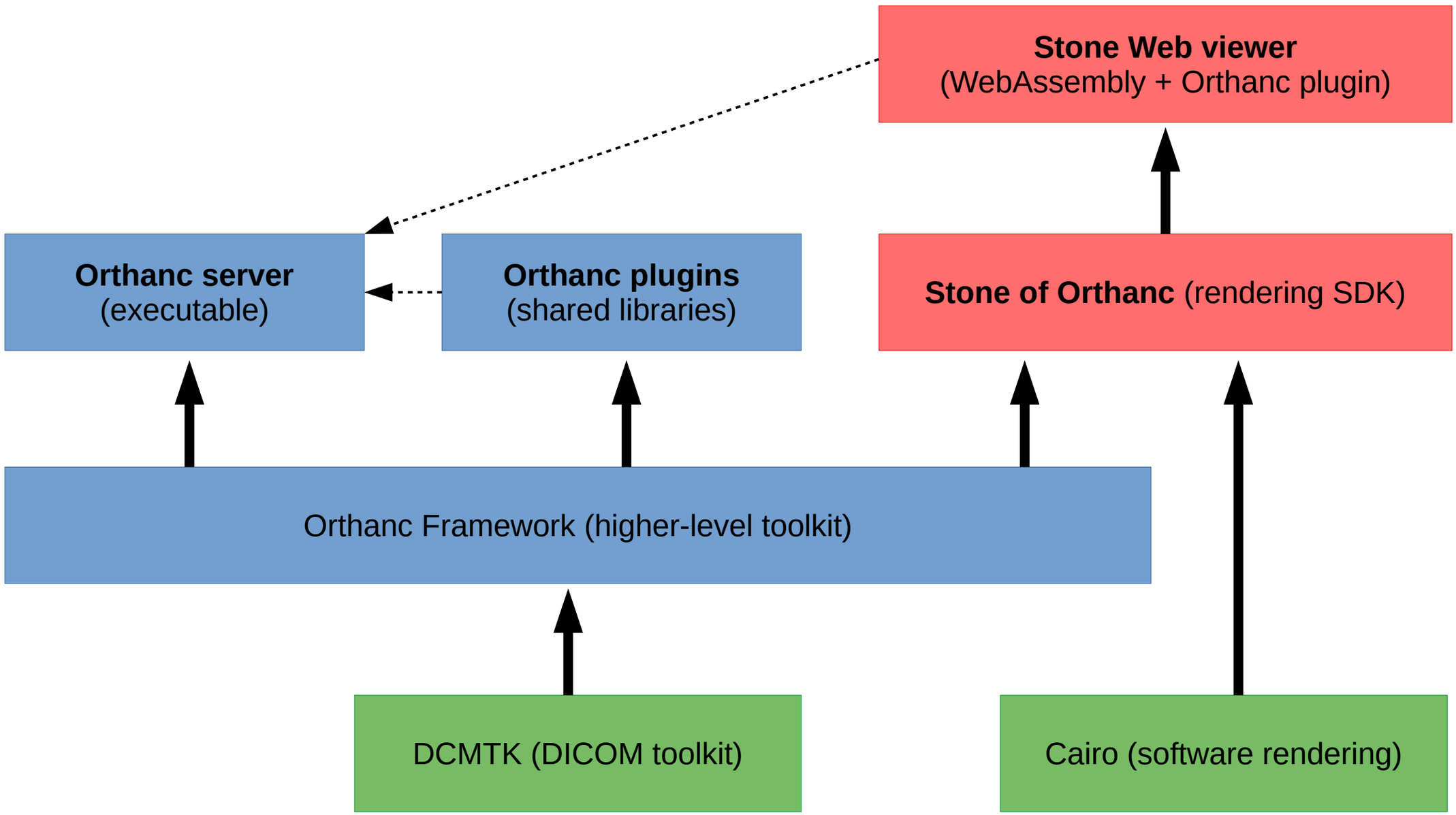
More advanced applications: 3D/MPR rendering







Place in the Orthanc ecosystem



Overview of the Stone architecture

Controller

Desktop: Qt, SDL Simple Directmedia Layer, ...

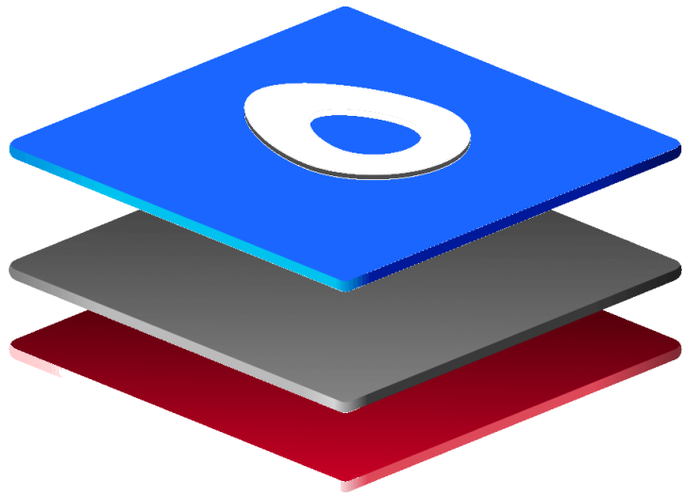
WebAssembly: HTML5, JavaScript

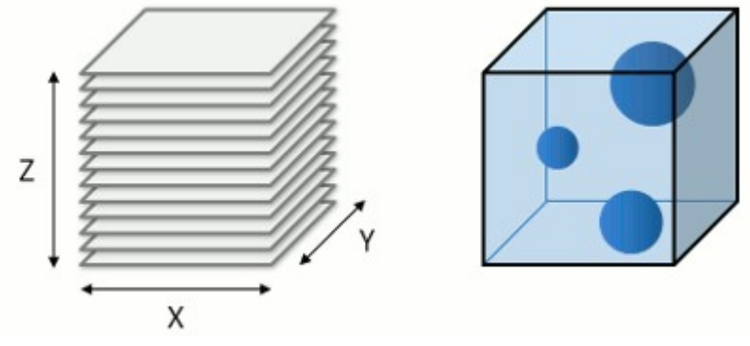
DICOM Digital Imaging and Communications in Medicine - DICOMweb

ORTHANC

Oracle and loaders

Layered 2D scene rendering

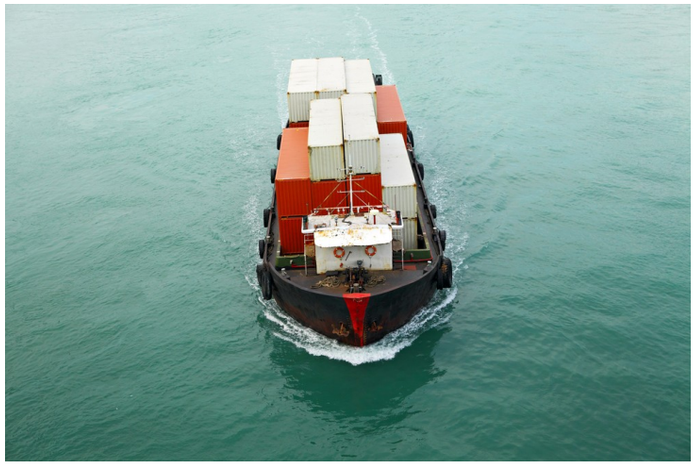




Model (slices and volumes)

Challenges of Stone

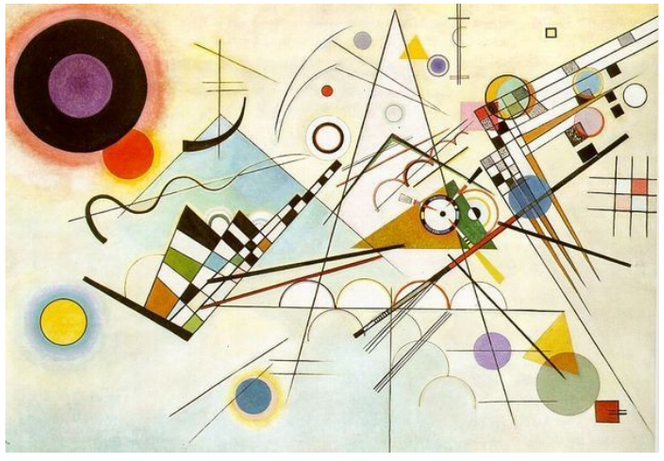
Deployments



- Need a Web server to serve WebAssembly
- Complexity of interaction with many technologies (C++, HTML, JavaScript, DICOMweb…)

- Partial solution: **Orthanc plugins** can add routes in the HTTP server embedded into Orthanc

Different models



- JavaScript is single-threaded and promise-driven
- C++ is multi-threaded and sequential
- Management of windows differ strongly

- Solution: **Oracle that abstracts** the system and network primitives, plus **platform-specific 2D viewports**

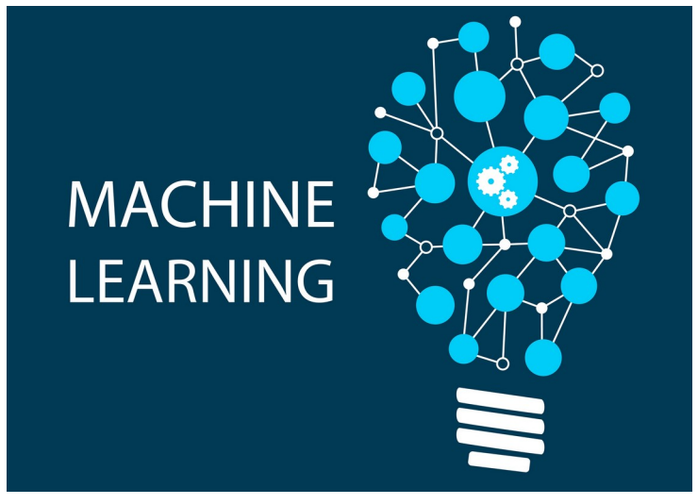
Software libraries

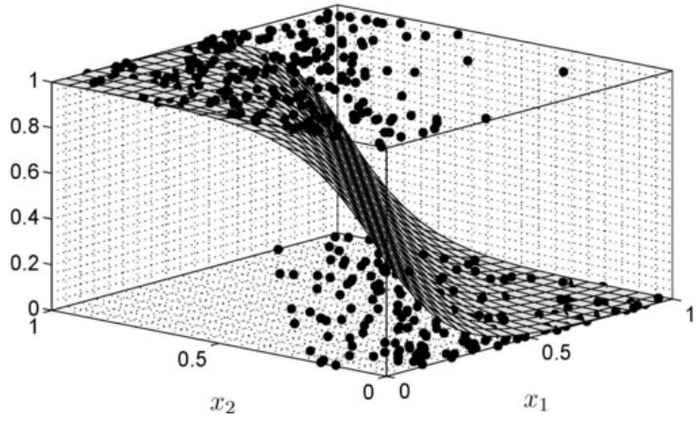
- Need to compile each third-party library for WebAssembly (no repository of “side modules” so far)
- Few thought about packaging WebAssembly in GNU/Linux distros so far

- Solution: CMake scripts of Orthanc already knows how to **statically build** many libraries

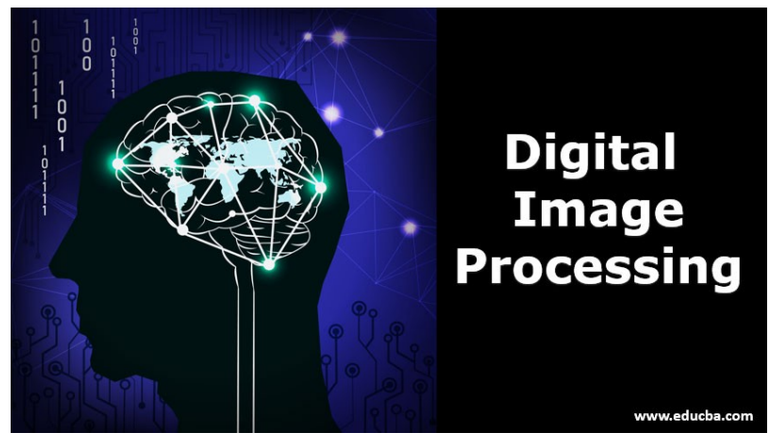
WebAssembly besides medical imaging

Ease the creation of Web applications
in any scientific field that deals with C++ code!





Bioclinical models



Conclusions

Our mission statement:

“Freely share knowledge about medical imaging”

- The Orthanc ecosystem is also about displaying medical images!
- **Stone of Orthanc** is a lightweight, cross-platform C++ library
- **Stone Web viewer** combines Stone of Orthanc with WebAssembly
- The viewer can be used with other PACS servers than Orthanc (DICOMweb)
- First official release: December 2020!
- Easy integration with GNU Health: Simply open the URL of the study :-)

ORTHANC

Thanks for your attention!