DBW – Databases and Web development

Josep Ll. Gelpí. gelpi@ub.edu

Aims

- Review technologies to handle bioinformatics data:
  - Computer communication, design of web applications, basic database design and optimization.

- This is NOT a programming course, it is about designing and building applications in an heterogenous scenario

- The final objective is to built a **fully operative application** using the appropriate combination of the techniques reviewed.

Bioinformatics & Internet

- Bioinformatics Tools and data should be available through web

- Ex. Nucleic Acid Research reviews:
  - Database Issue (January)
  - Web Server Issue (July)

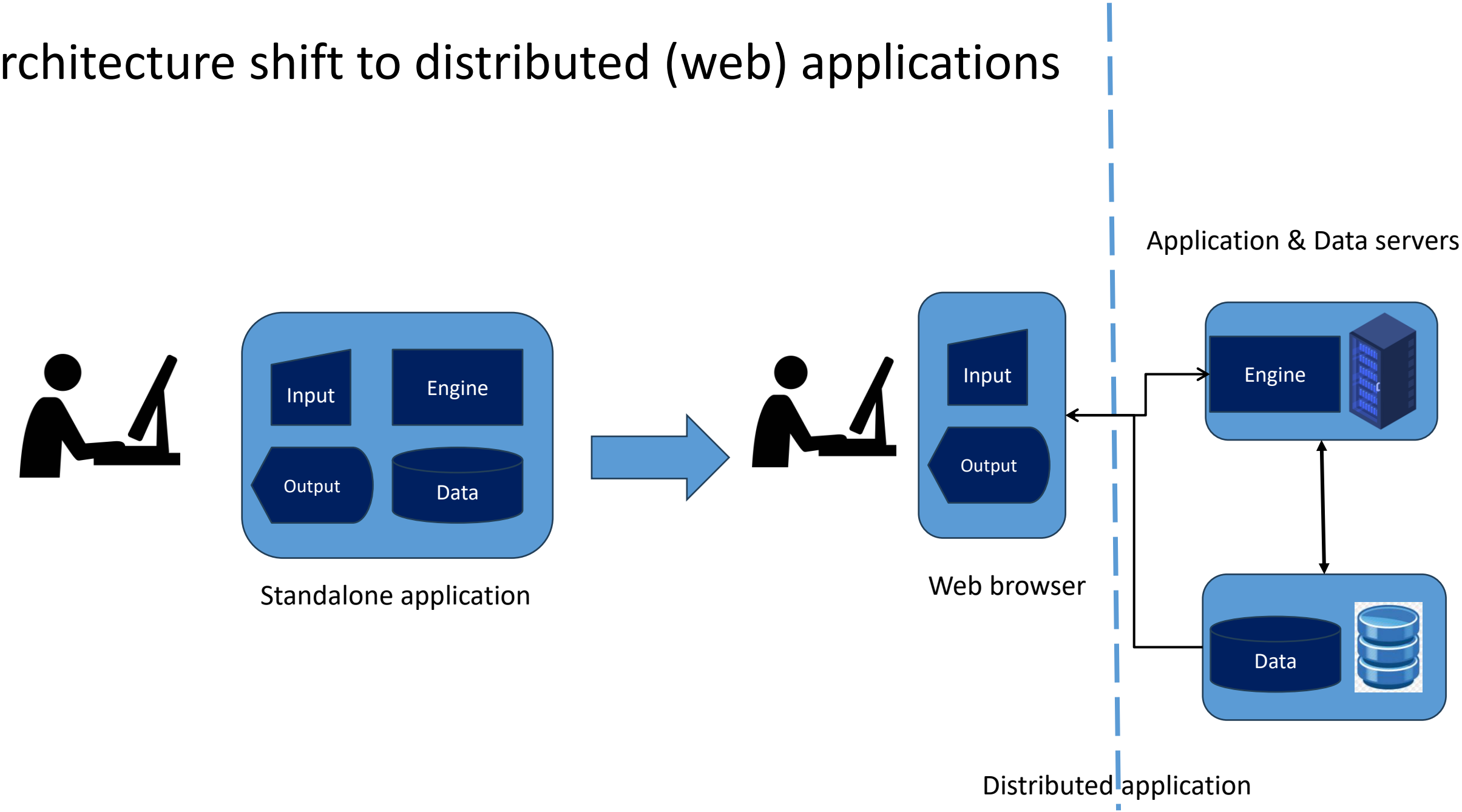
NAR Database issue recommendations for authors

- “The pre-submission enquiry must present a working web accessible database “
- “The quality, quantity and originality of data as well as the quality of the web interface are the most important. Good data with a poor interface or vice versa are never sufficient for consideration. “
- “Do get a domain name for your website. URLs to specific IP addresses/ports are unlikely to stand the test of time.”
- (…)

Nucleic Acids Research, Volume 35, Issue suppl_1, 1 January 2007, Pages D1–D2

https://academic.oup.com/nar/article/35/suppl_1/D1/1088333

Architecture shift to distributed (web) applications

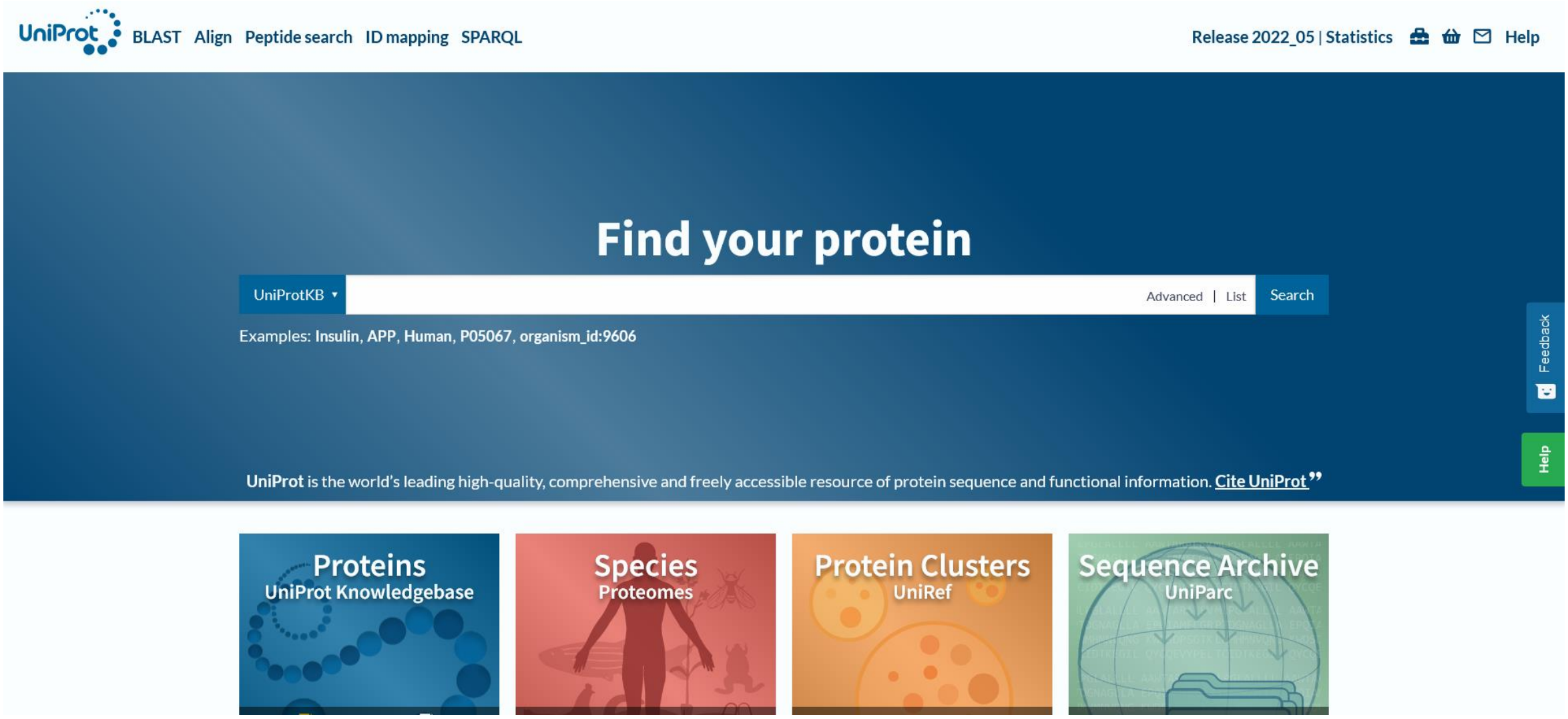


Web applications by access type

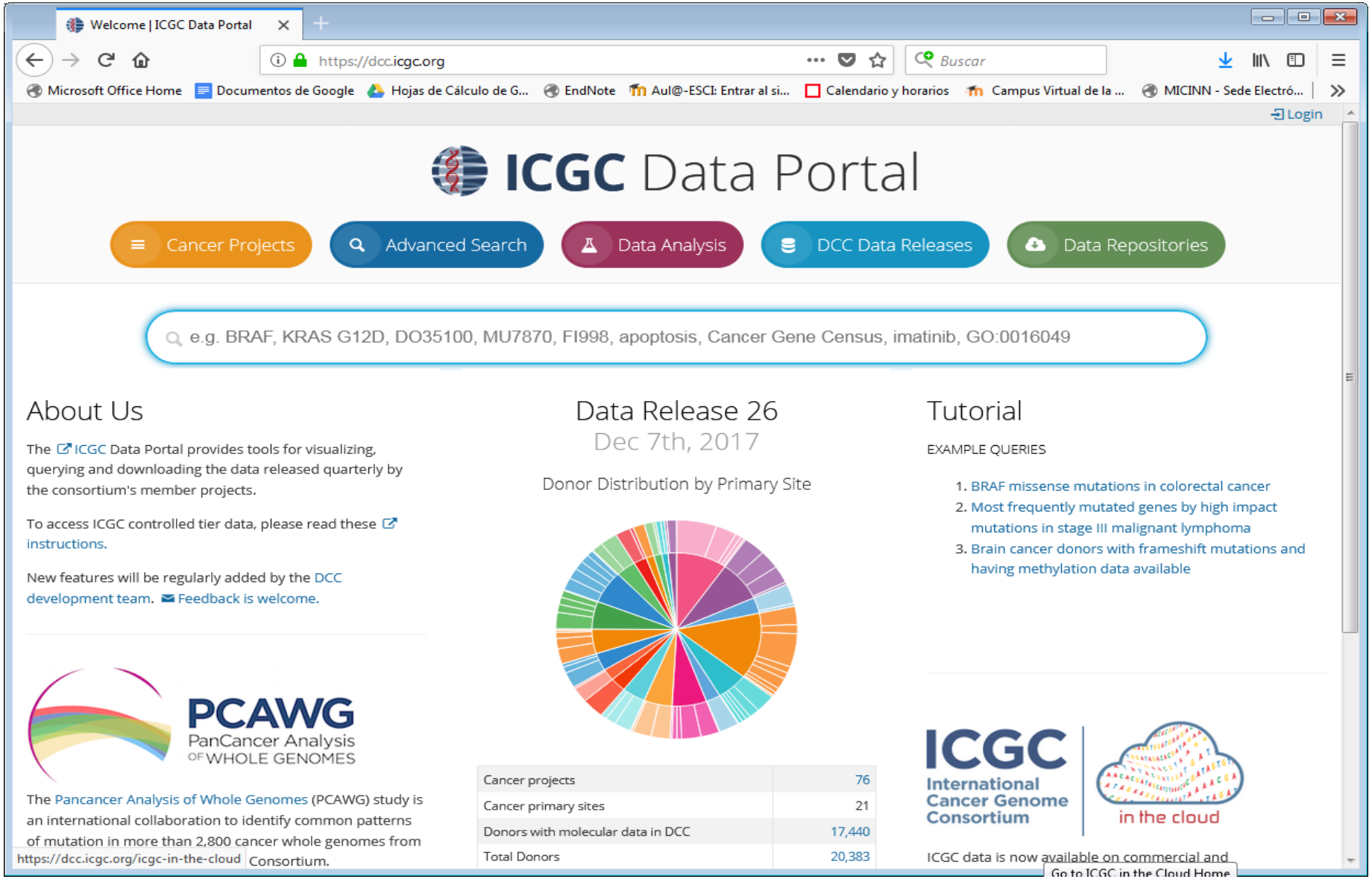
- **Web interfaces**
  - Provide a user friendly interface (web based) to "human" users
    - Users known how to use the interface
    - There is no need to install software
    - Single operations (no large scale)
- **Web services & APIs**
  - Provide a programmatic interface (using Web protocols)
  - Intented to interact with software, not with humans
    - Well-defined data formats required.
    - Adequate for large scale operations
- Modern applications will normally offer both
  - Web frontend is normally just another client of the API's

- **Application styles**
  - Access to data
    - Friendly interface to data repositories (aka Data Portals)
  - Web Interfaces to stand-alone software
    - Collect input parameters, run, and redirect output
  - Workbenches (e.g. Galaxy)
  - On-purpose applications & DBs

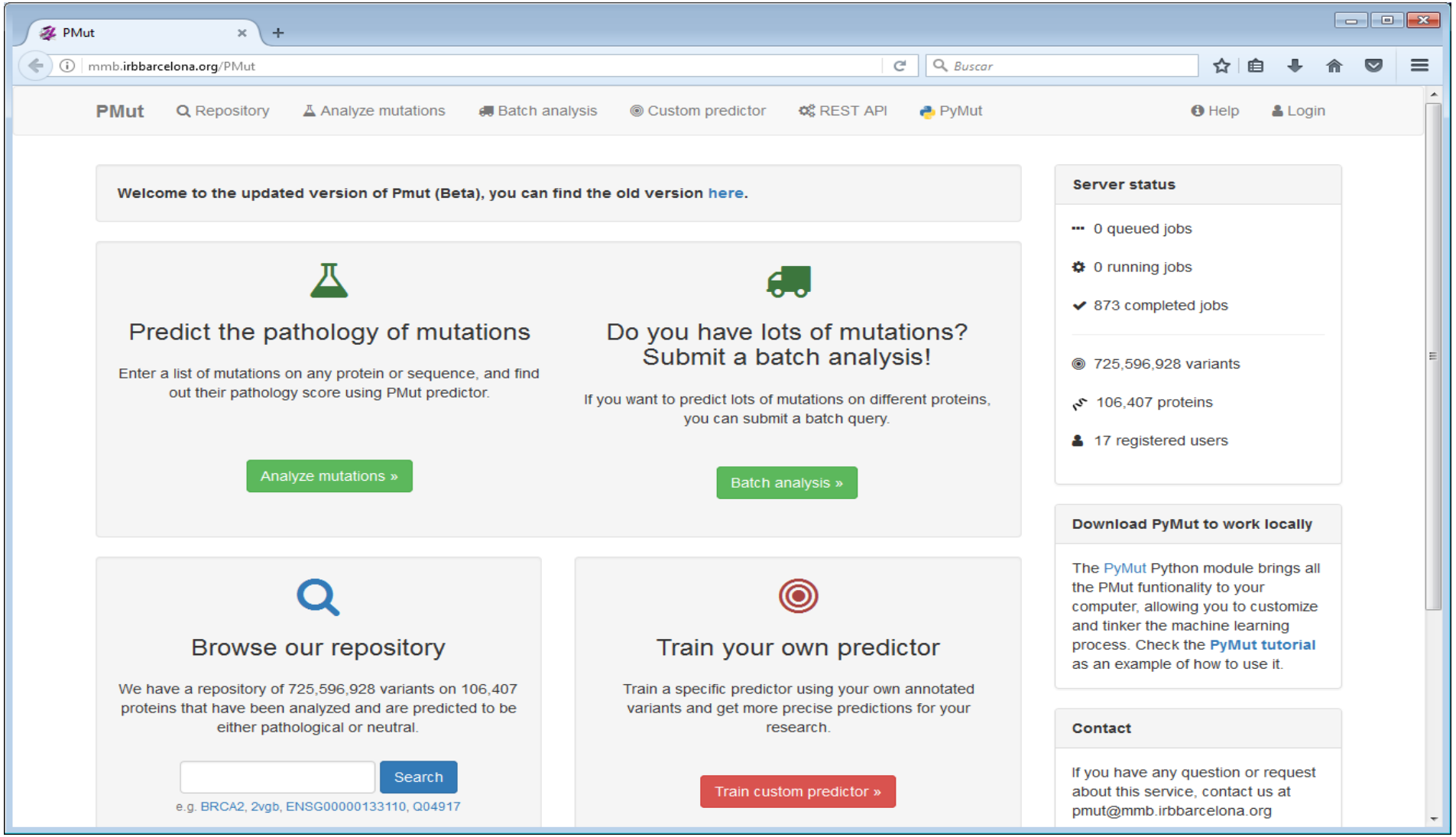
Data Portals



Project Data Portals

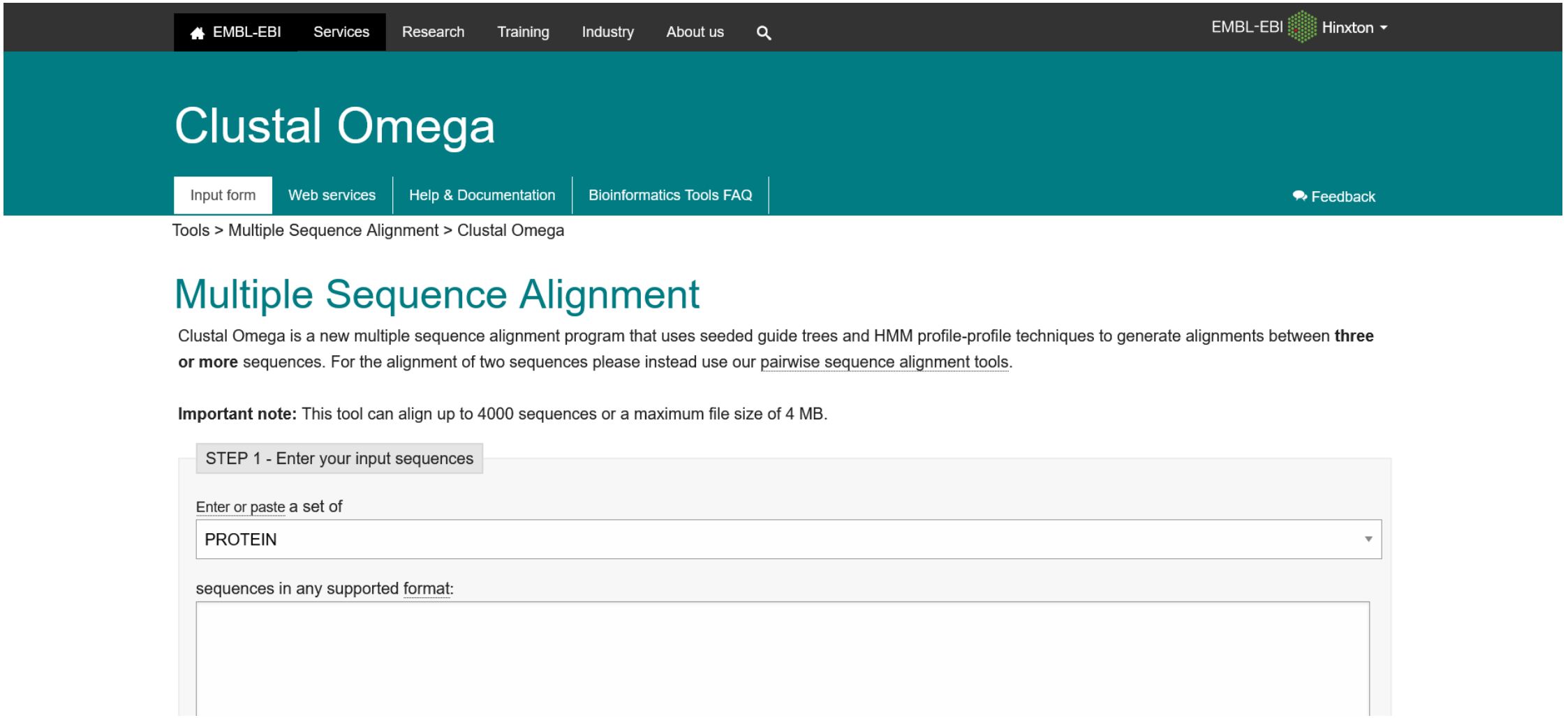


Or both...

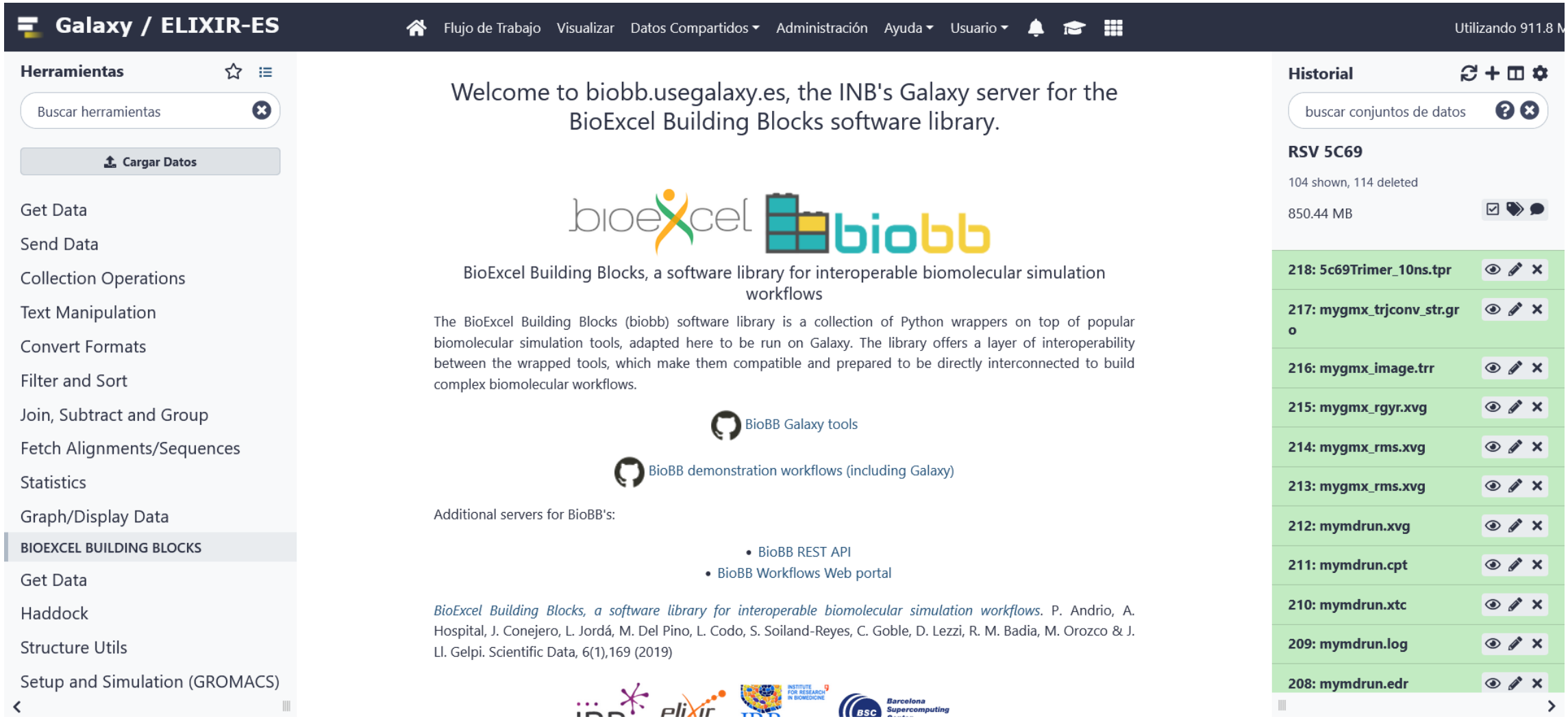


López-Ferrando *et al.* **Nucl. Ac. Res** 2017

Web interfaces to bioinformatics applications



Workbenches



Integrated platforms

Data Catalogue

Access Manager

Data Analysis

User Support and Docs.

DAC

Metadata

Cohorts

EURO-BIOIMAGING

EUROPEAN GENOME-PHENOME ARCHIVE

Controlled Data

Tools Repository

Workflow management systems

COMPSs NextFlow

SGE

Resource Management

HPC

Private Cloud

https://eucanshare.bsc.es

euCanSHare

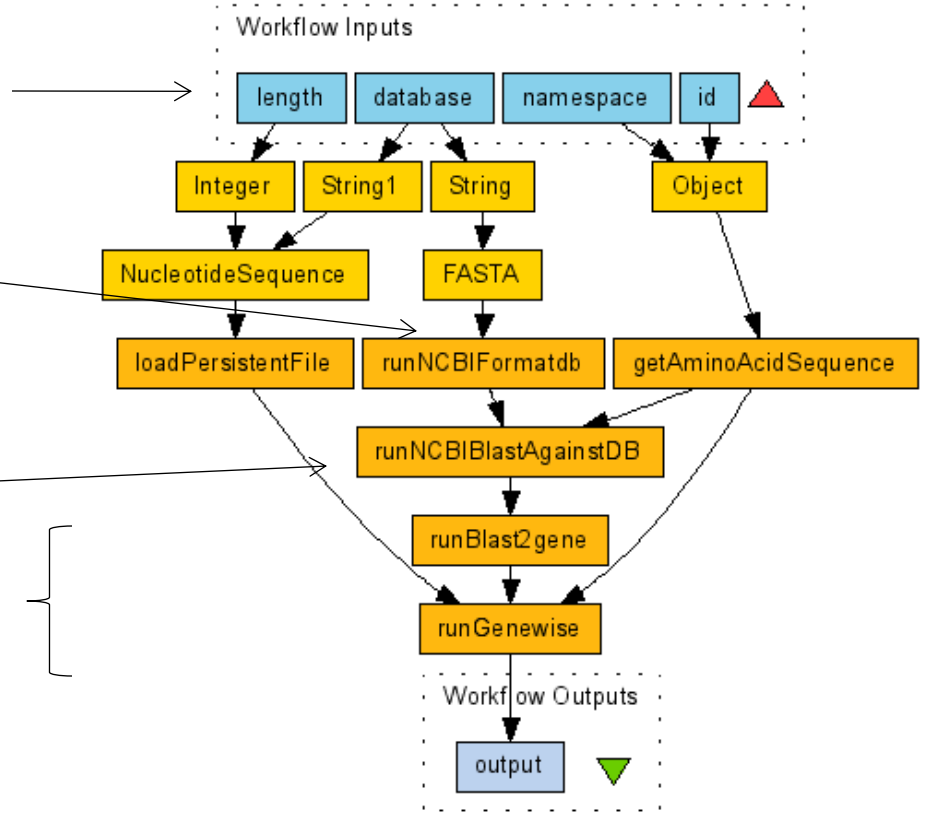
Bioinformatics web-services and workflows

Input: Protein Id and DNA genomic sequence

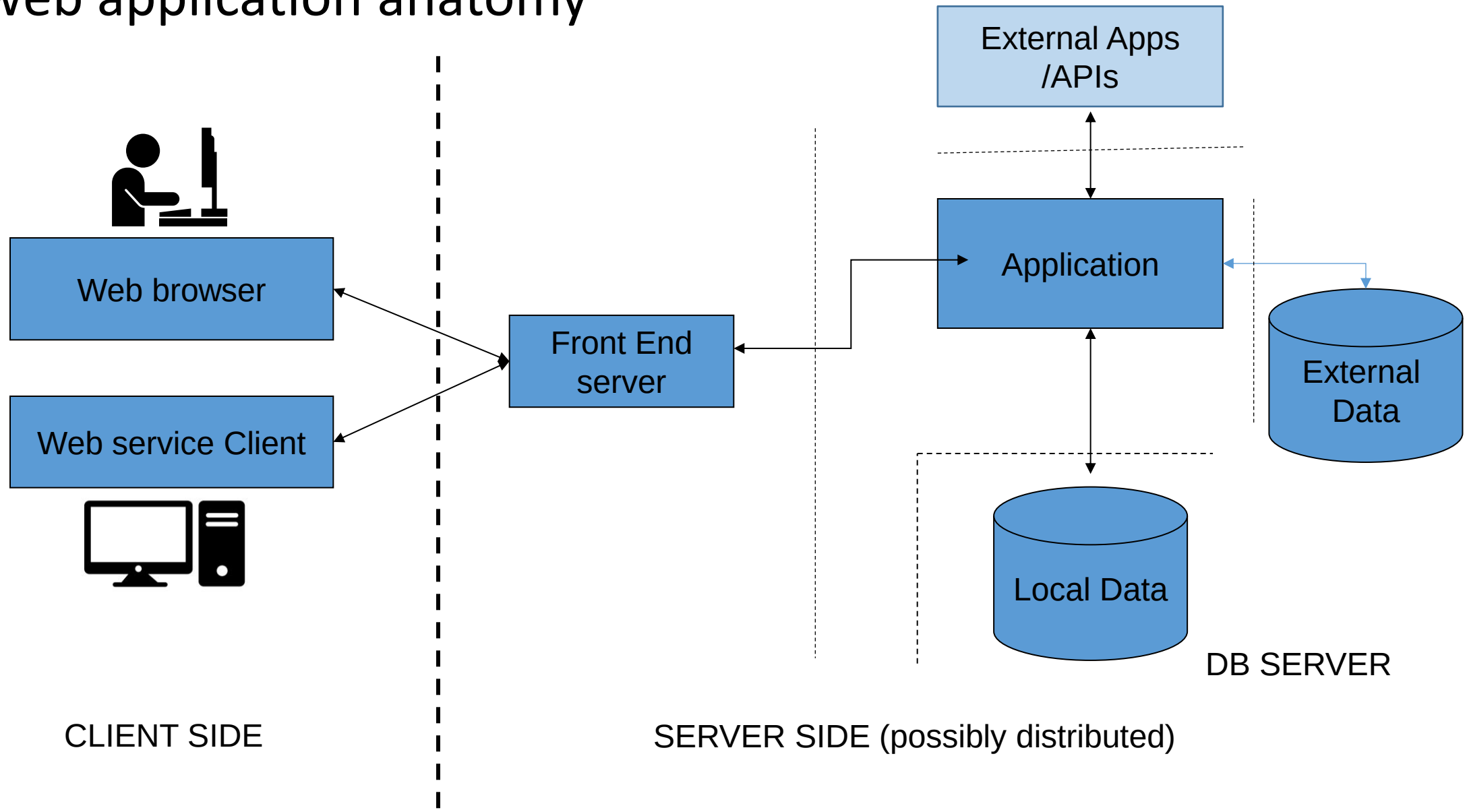
Building of Blast Database from DNA seq.

BLAST Search

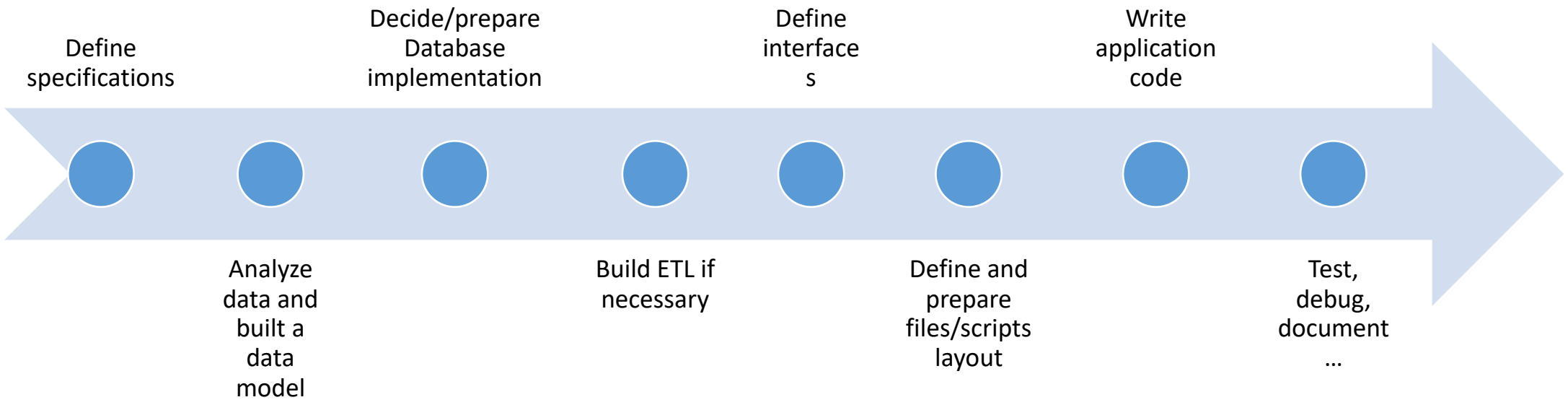
Run GeneWise to detect gene structure



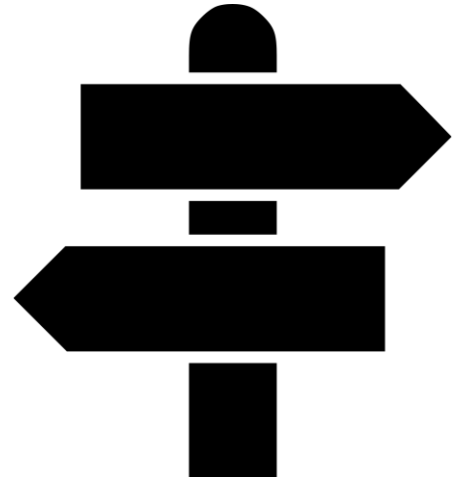
General web application anatomy



Building a (web) application, usual steps

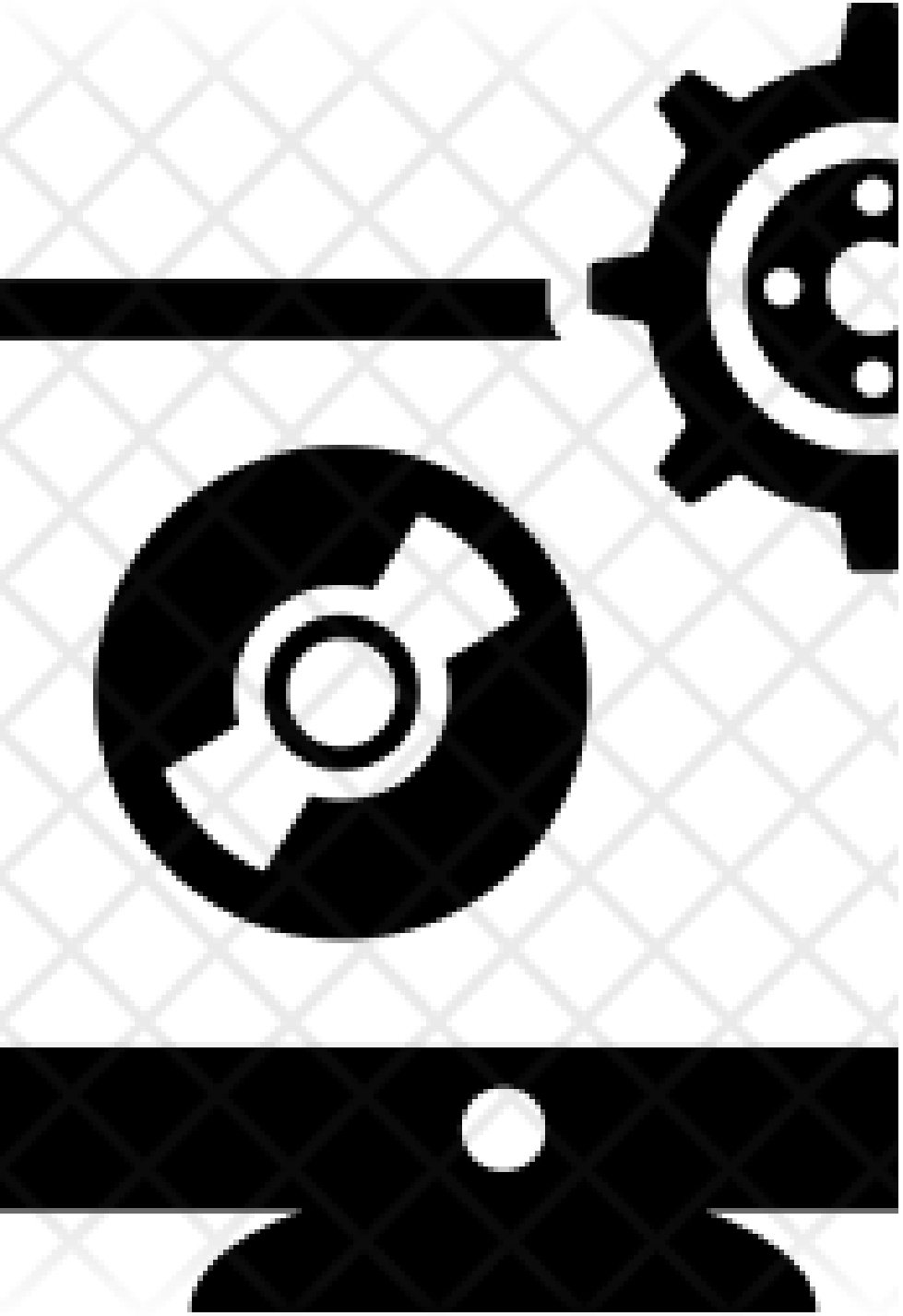



Logistics



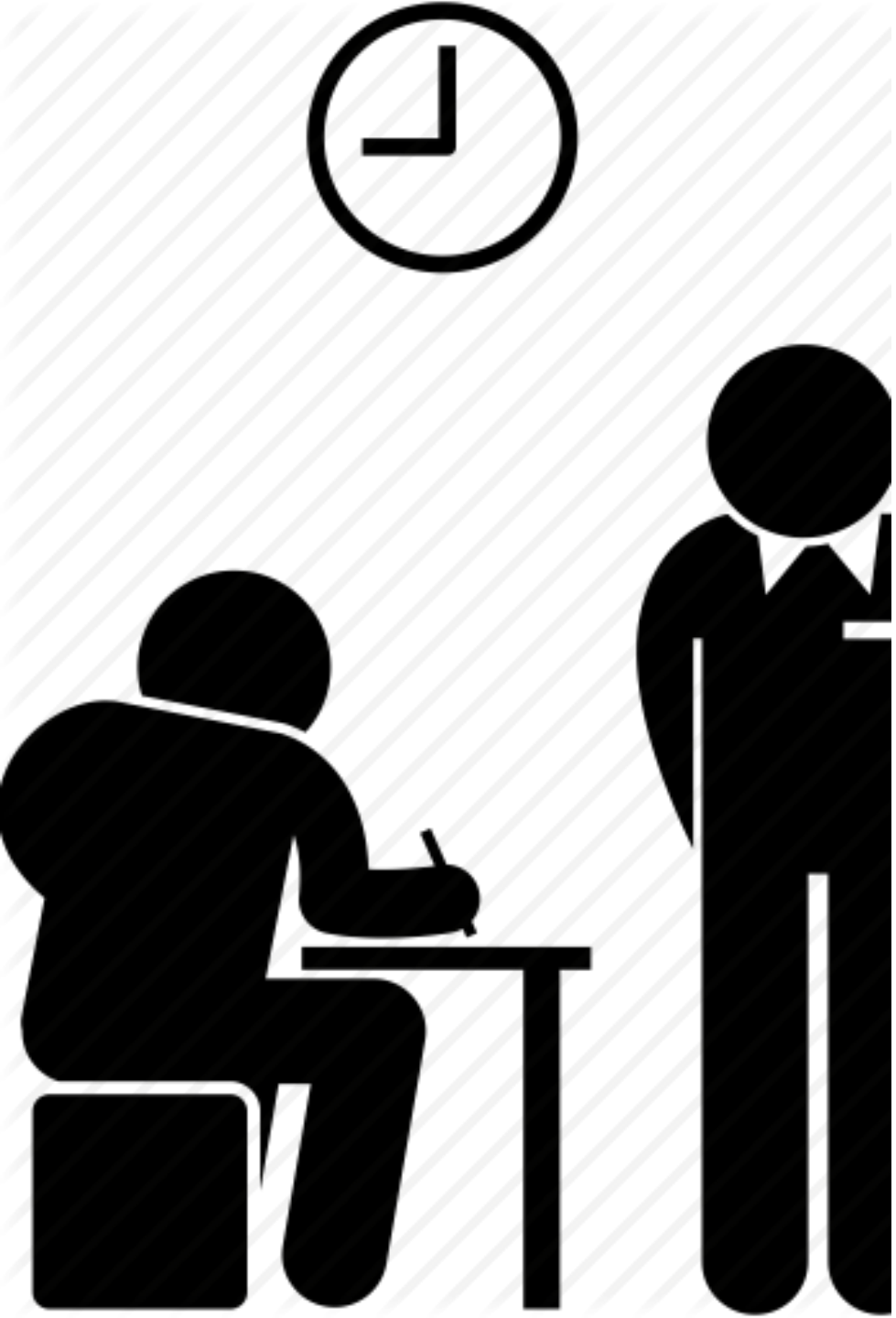
- Web site(s)
  - Course materials:
    - https://formacio.bq.ub.edu/
  - Personal sites:
    - https://formacio.bq.ub.edu/~uXXXXXX

- Server
  - SSH Access
    - ssh formacio.bq.ub.edu –l uXXXXXX
    - Password dbw_uXXXXXX
  - SCP
    - scp uXXXXXX@formacio.bq.ub.edu …
  - MySQL/MongoDB Access
    - Localhost only
    - DBs on demand



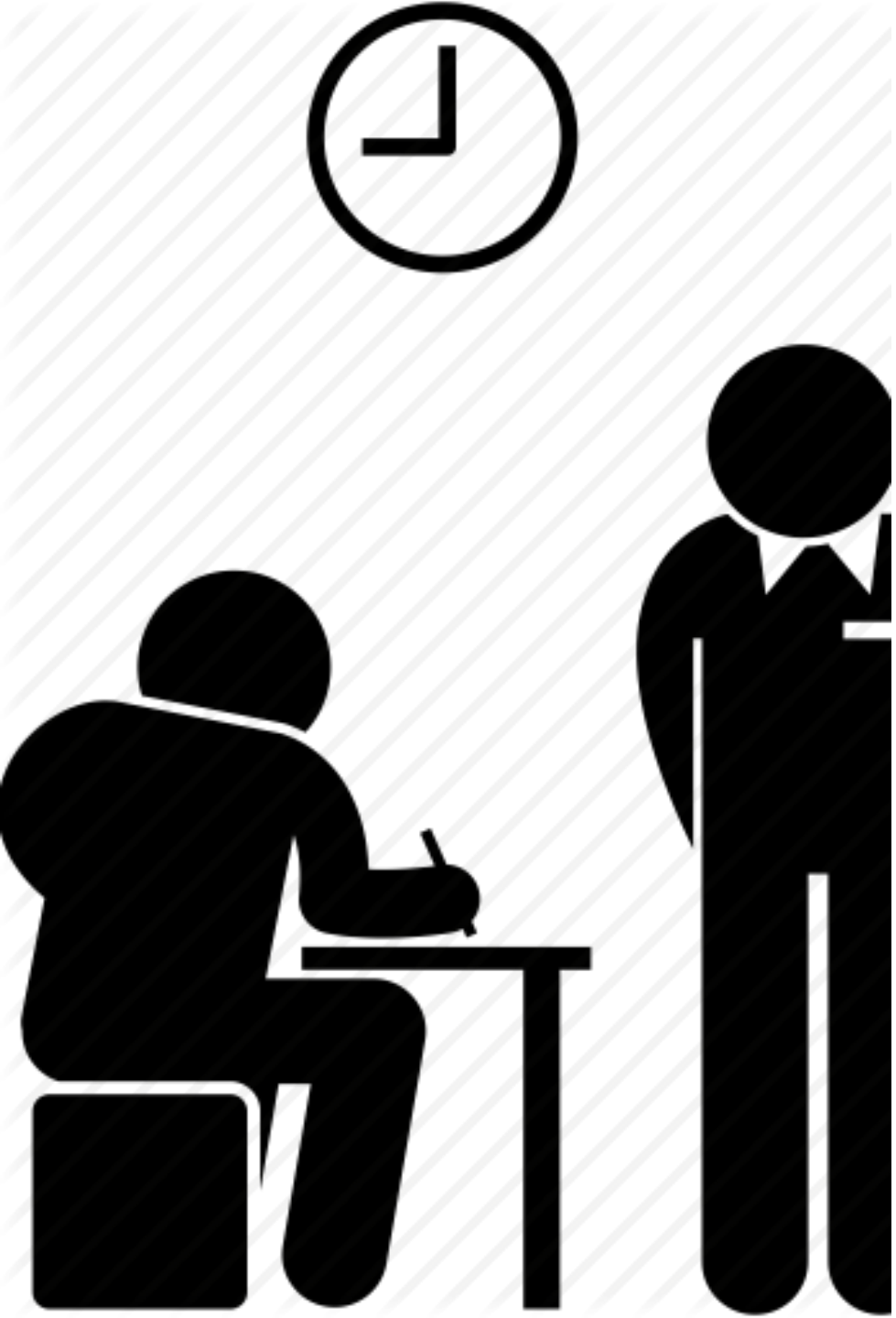
Software to install

- Ideally Linux
  - Also Windows WLS or Mac
- From Linux distributions
  - A Web server (one of)
    - Apache (with PHP 7.x)
    - Nginx (better for Python apps)
  - MYSQL (or MARIADB) server
  - MYSQL Workbench or phpMyAdmin
- Your preferred software editor
- MongoDB (optional)
  - Install drivers for PHP/python if needed



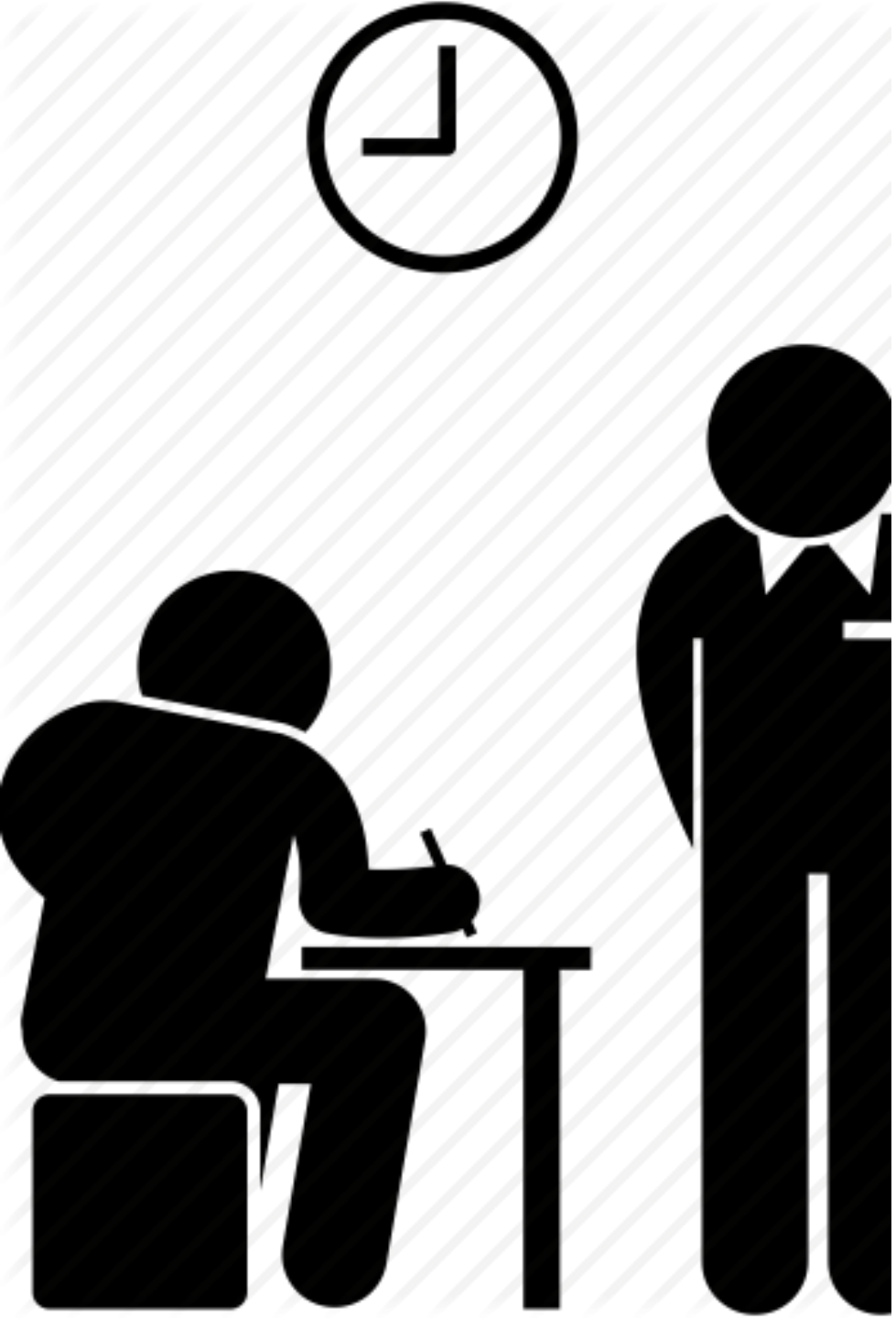
Evaluation

- Personal web site (20%)
- Exercices, in-class projects (20%)
- Web application project (60%)
  - Progress presentations
  - A fully operative web application using a local Database



Evaluation

- Web application project
  - 3-4 people / group
  - Free subject (bioinformatics preferred)
  - **Should include DB management, web interface, users' management** (Mysql or MongoDB)
  - May use **fake data if necessary**
  - Available at the personal web sites of all team members
  - Preferred languages: PHP, Python, …
  - Source code at github or equivalent



Evaluation

- Web application project
  - Steps (and Deadlines):
    - Initial specification (Presentation 20th Jan)
    - Data analysis & Database design (Presentation 27th Jan)
    - Project prototype Demo (Presentation 3th Feb)
    - Mid development review (meeting around 20th Feb)
    - Final application (End of Term)

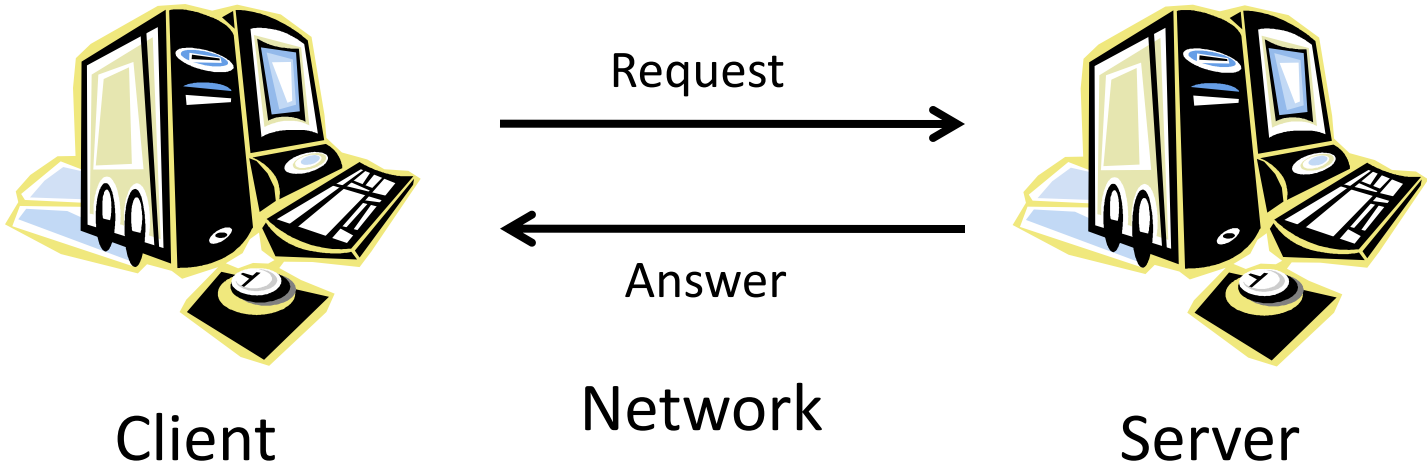
- Fully Installed and functional **on course server**
  - PHP projects will use Apache
  - Python projects will use uwsgi/nginx, uvicorn,… (dev servers not acceptable)

Basic computer communication protocols

Aim & Outline

- Understand the basic components of computer communication protocols
  - Concepts of client and server
  - Addressing servers and data
    - Computer addresses (MAC Address, IP Address, DNS)
    - Ports
    - Resource identification: URL/URI concepts
  - Client/server transactions
    - HTTP protocol

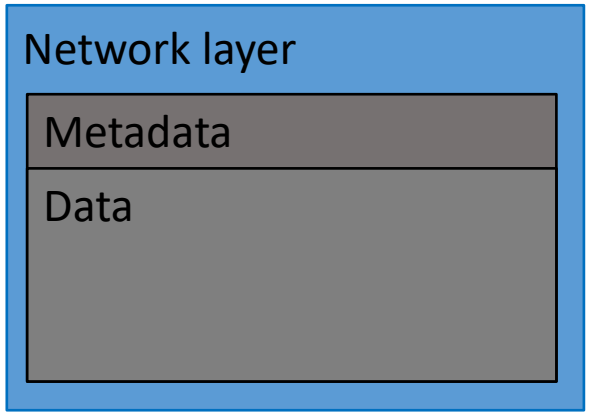
Some definitions …



- Clients and Servers require logic and physical addresses
- Clients and servers refer both to computers and to software components
- Data transferred requires MetaData (meta-information)
  - Nature of data
  - Request (what to do)
  - Applications involved (email, web, etc.)

How it works: TCP/IP

- Packet switching
  - Packet switching breaks the signal in small fragments
  - Users have the idea of a dedicated line but, in fact, it is not.
- Computers connected to internet should have addresses/ports
  - MAC Address: Address of the physical interface
  - IP Address: Unique address of the computer
  - Unique Host name
  - Ports to point to specific applications



Identifiying computers

IP Addresses

- Allow to find destination irrespective of the nature of the network media.
- Each device has a **“unique” IP address**
- IPv4: 32 bits (4 x 1 byte (0-255) numbers)
  - Max: 2^{32} : aprox 4.3 x 10^9
  - P. ex. 161.116.72.181 (formacio.bq.ub.edu)
  - The 4 levels are hierarchical
- Some addresses are reserved, and some networks are “local”
- (Coming but still not used) IPv6: 128 bits (16 bytes). Max: 2^{128} (3.4 x 10^{38})

Name addresses

- IP addresses are not easy. Most hosts have also a “name”:

  f. ex. www.ncbi.nlm.nih.edu
- Host names have a structure similar to IP addresses:
  - Top domains (.es, .edu, correspond to full class domains and subnets are indicated by prefixes).
  - ub.edu (161.116.x.x)
  - bq.ub.edu (161.116.72.x)
  - Formacio.bq.ub.edu (161.116.72.181)

Identifying applications within servers: **Ports**

- Each host has **one (at least) IP address** but has **several ports to identify services within**

- Ports are 2-Byte numbers.
  - 0-1023 "**Well known ports**" (Telnet: 23, FTP: 21, HTTP: 80, HTTPS: 443, ..) (only root)
  - 1024-49151 "**Registered ports**", usually managed by applications (MySQL: 3306) (only root)
  - 49,152-65,535 "**Dynamic and/or private ports**" freely usable.

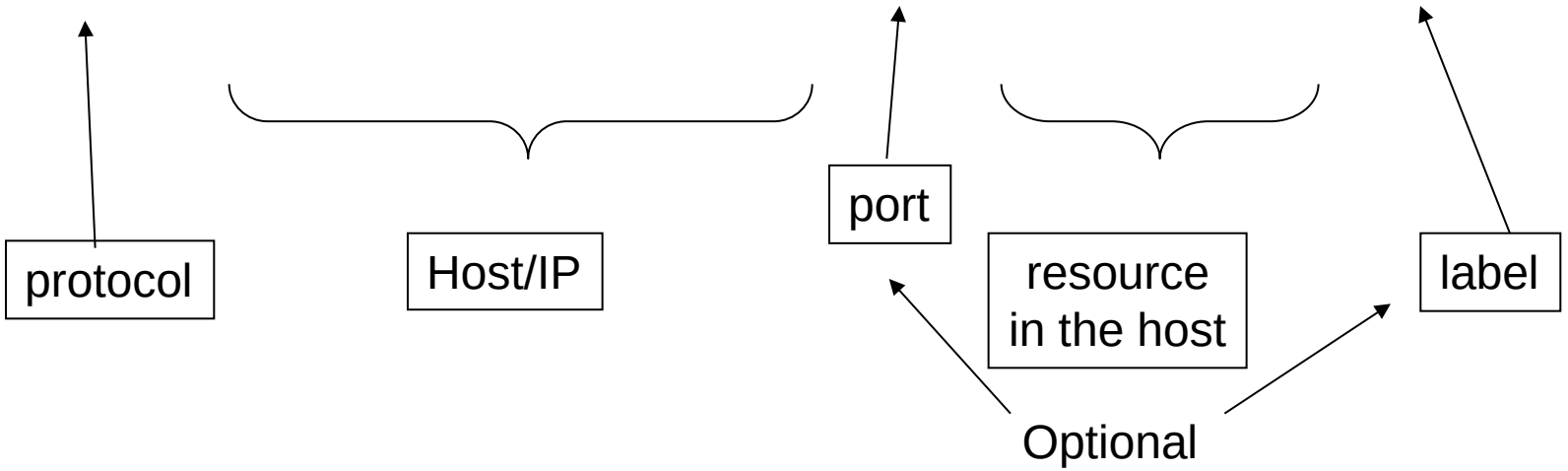
- **Communication to ports triggers the specific application** to deal with the data

- However, different ports from the official ones can be used to:
  - Hide applications, Have more then one server in the same IP address, Hide servers in internal networks.

URI/URLs

- Resources must be identified in a way that includes all the necessary details:

https://formacio.bq.ub.edu:443/index.htm#top



Missing parts of the URL are added by the client by default!!

Client – server communication (HTTP)

- Most Web Applications use HTTP (hypertext transfer protocol), although sometime FTP, SMTP

- HTTP is a client-server protocol
  - Link between client and server is dynamic
  - Usually **limited to a single transaction**
  - Requests composed by a **query** operation and a variable set of headers (Metadata)
  - Answers: headers + data

- Relevant Operations: GET, POST
  - GET: Simple retrieval, all information included in the URL
    - Simple queries, static information
    - Usable from as hypertext links
  - POST: Upload and retrieval
    Query defines the resource, and input data follows
  - PUT: Similar to POST. Used in APIs

- Relevant HTTP headers
  - **Content-type** (POST): input data format
  - **Content-type** (Answer): Data MIME type (text/html, image/jpg, …)
  - **Location**: Redirects browser
  - **Set-cookie**: Set a “cookie” on users’ software.