

# Internet Oriented Systems

University of Parma - A.Y. 2020-2021



## Collaborative Data Curation Platform

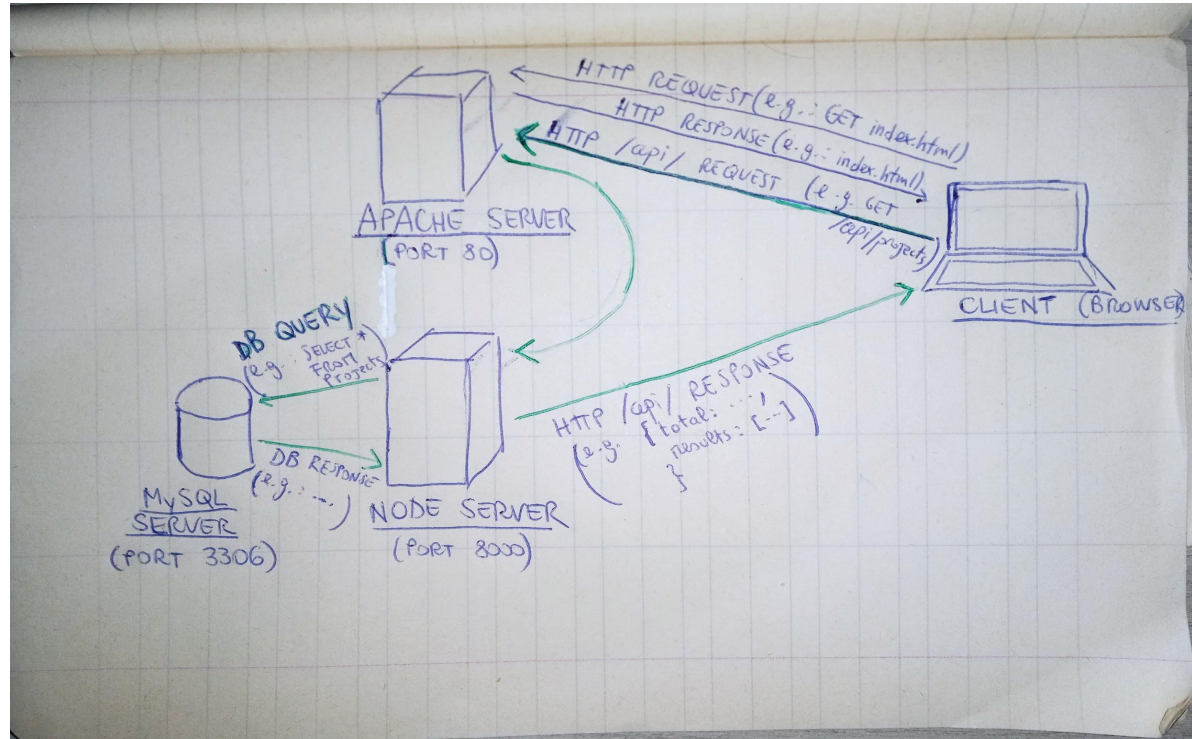
*Architectural design and implementation technologies*

Giuseppe La Gualano [[giuseppe.laqualano@studenti.unipr.it](mailto:giuseppe.laqualano@studenti.unipr.it)]

Francesco Vetere [[francesco.vetere@studenti.unipr.it](mailto:francesco.vetere@studenti.unipr.it)]

# Architectural design

Si potrebbe quasi togliere questa slide... pensiamoci un attimo magari...



# Technologies



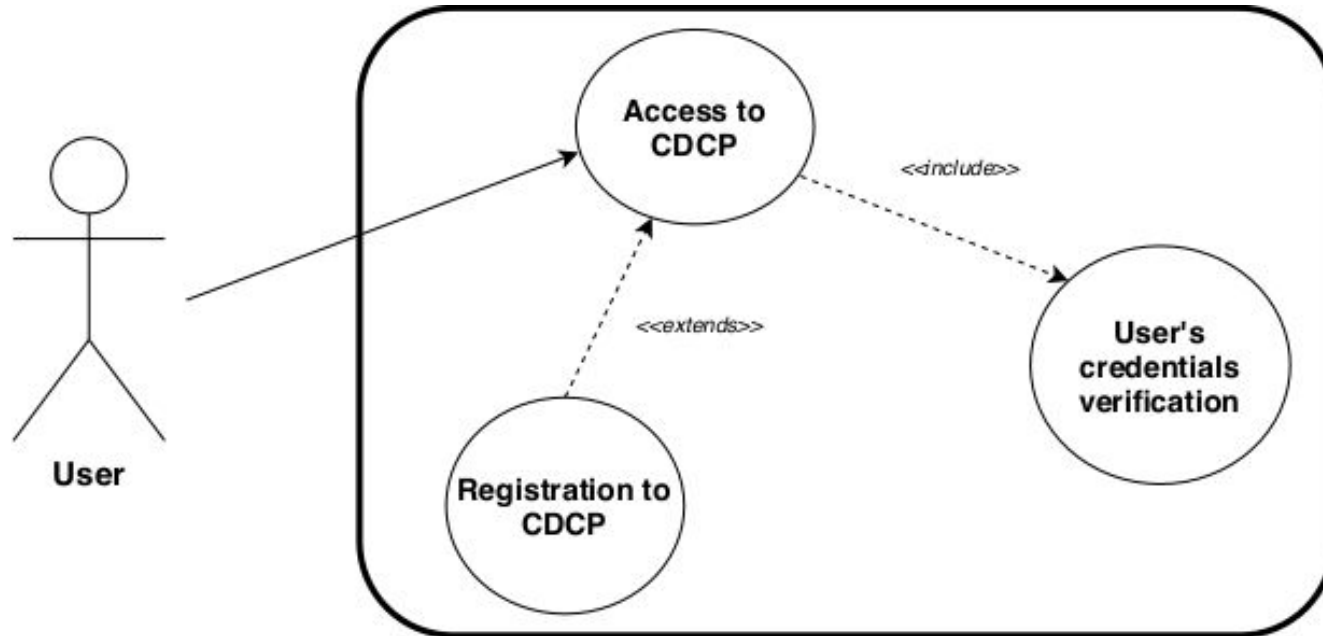
## Frontend

- HTML5
- CSS3 and Bootstrap
- JavaScript and JQuery

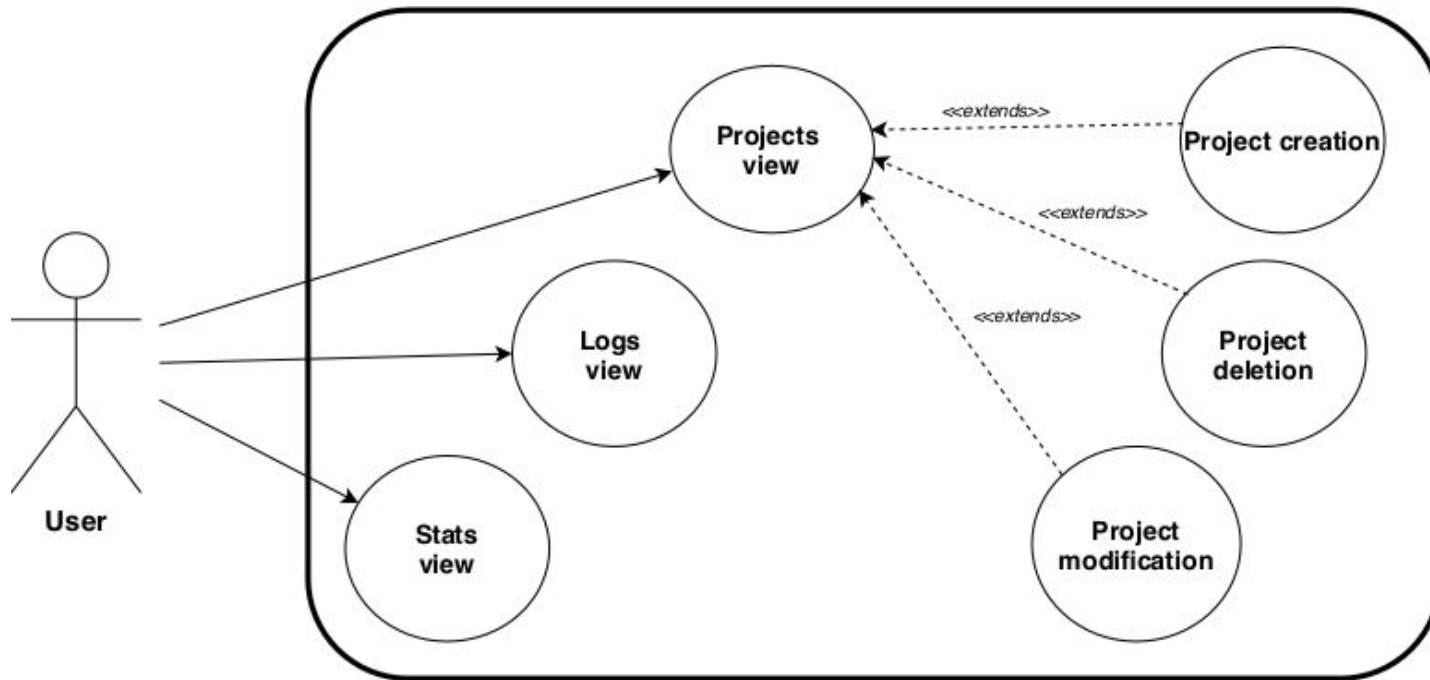
## Backend

- Node.js
- Express.js
- MySQL

# Use case scenarios



# Use case scenarios





# Database logical schema

**Users** (id, nickname, email, password, registrationDate)

**Projects** (id, title, inputType)

**Examples** (id, Projects.id, inputType, inputValue)

**TagNames** (Projects.id, Examples.id, tagName)

**TagValues** (Projects.id, Examples.id, TagNames.tagName, tagValue)

**Logs** (id, Users.Nickname, Projects.id, actionType, details, timeStamp)

**TokenAuth** (id, nickname, token, expirationDate)



# Single Page Application

- Client performs just one requests to the Apache server for all the static contents, at the very beginning (index.html, JS/CSS files, images).
- Client has now its own state and logic, needed in order to display contents on screen: in particular, no other static resource is required from the Apache server.
- All the other requests will be /api/ requests, performed as AJAX calls: form submits will not cause any page reloading!

# APIs



- RESTful APIs are used, and they are implemented with Express.js using JSON as representation format.
- Every API performs a (minimum) check on body/params values, before executing the corresponding query.
- For almost every DB entity, 4 APIs are implemented, in order to perform CRUD operations on them.
- API's do not need to worry about deleting referenced objects: "on delete cascade" logic is already implemented in DDL commands that build up the DB.



# Database interactions

- Queries on the database are performed server-side using promise-mysql.js library (async/await 😊).
- Moreover, prepared statements are used: every query is processed in order to prevent SQL Injection attacks.
- A class DBManager.js has been written, which takes care of opening a connection, performing a query, and closing the connection.

# Storing images

- TODO



# Cookies

- TODO



# Authentication



- HTTP is used as application layer protocol: this implies having an insecure basic authentication scheme.
- However, passwords are not stored in clear in the DB. In fact, before storing them on DB a simple hash is applied, using bcrypt.js library.

# Software implementation



## Frontend

- index.html
- custom.css
- application\_logic.js, <xyz>\_page.js, cookies.js

## Backend

- server.js
- options.js
- routes.js
- DBManager.js

# Security issues/future developments



- (TODO: HTML escaping, visualizzazione compatta se si hanno molti examples, ...)