**PROGETTO SOFTWARE – GeoControl**

**GeoControl** is a **WebApp** composed of:

* WEB SERVER exposes HTTP/rest/JSON APIs
* WEB CLIENT interacts with the server through HTTP/Rest/JSON calls
* DB saves app data; we will use SQLite as DBMS in the mandatory part (in **Task4** [***Optional***] we will **docker**ize all components and replace SQLite with **MySQL** to support multi-component deployment)

The **Front-End** will be provided fully implemented in **Task2**

**Task1** will be only on the **Requirements** (theory part)

The **Back-End** must be implemented in **Node.js** using **Express** (as Applicazioni Web 1) and **TypeORM**; the language used must be **TypeScript**. It’s available in project repository; instructions and architecture are described in **README.md** (that includes provided development and execution scripts, usage instructions and useful technical details).

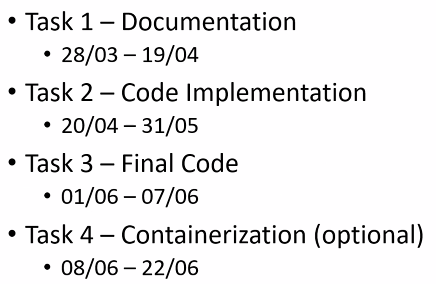
The Back-End provide the implementation of the **HTTP**/**Rest**/**JSON APIs** defined in the **OpenAPI** (**Swagger** [a document from which start to define the requirements]) specification. In:

* **Task1**: only the Swagger UI is available
* **Task2**: complete architecture + some example features as reference for our work

**Task3** (the last mandatory) will be only fix the code basing on the professor feedback

GeoControl Front-End & Back-End use the **npm** package manager (Node.js as Applicazioni Web 1), which build 2 important files:

* **package.json**  project metadata, dependencies, scripts to automate tasks
* **package-lock.json** exact versions of all installed packages to ensure consistent installations

**TIMELINE**:

⚠️ NON MODIFICARE (nemmeno con spazi) IL FILE **swagger\_v1.yaml** !!!!!!!

**TASK1 – Documentation** **analayze the system** based on the provided OpenAPI (Swagger) specification + produce 2 documents: **Requirement Document + Effort Estimation**

**TASK2 – Implementation & Testing**:

* **implement the system** according to the OpenAPI (Swagger) specification
* **extend the provide codebase**, which will include:
  + complete execution flow (routes 🡪 controllers 🡪 services 🡪 repository 🡪 model) as an example
  + example tests at all levels (unit, integration and end-to-end)
* **complete the missing features** based on the Swagger definition
* **write your own tests to ensure full coverage** (target = coverage as close as possible to 100%)

**TASK3 – Final Code Verification**:

* a complete e2e suite will be provided: these tests will verify that all features defined in the Swagger specification have been correctly implemented
* we have to: **run the test suite** against our code + **fix any issues** as needed + **submit a final version of the** **code that passes 100% of the provided tests**

**TASK4 – Containerization**:

* **containerize** the entire system (backend + frontend + db)
* **replace SQLite with MySQL**, using the official Docker image (an init.sql script will be provided to initialize the db)
* containerize the backend:
  + create a Docker file for the backed app
  + create-root script must be executed before the backend starts, to create an admin in the db
* create a **docker-compose.yml** file to orchestrate all components
* no changes to the backend code will be necessary

⚠️ Se faccio **npm install** (che riscarica tutte le dipendenze e crea l’ambiente per Node.js) e poi faccio **npm start** (starta un web server su localhost:5000), poi posso andare all’indirizzo **localhost:5000/api/v1/doc** e accedo allo Swagger con la documentazione