# CLONAZIONE DEL REPOSITORY

* Selezionare una directory di destinazione per la clonazione e posizionarsi all’interno tramite terminale che supporti git
* Digitare da terminale il comando 🡪 git clone <https://gitlab.di.unipmn.it/pissir2021-22/gruppo-6.git>
* Inserire username e password di GitLab
* Il progetto (documentazione e applicazione) è ora clonato nella directory scelta

# COSA SI TROVA NELLA CARTELLA PRINCIPALE DEL PROGETTO

* Directory docs: all’interno si trova la documentazione testuale richiesta più la suddivisione dei compiti
* Directory diagrammi: all’interno si trovano tutti i diagrammi richiesti in formato visual paradigm e pdf
* Directory Emulatore: si trovano i sorgenti Java per emulare l’invio di misure e la ricezione di comandi da parte dei dispositivi iot presenti nel db in quel momento. Si spiegherà successivamente come utilizzare l’emulatore
* Directory WebApp: contiene tutti i sorgenti html, js, file del server, etc (frontend e backend) utili a far funzionare le API, la logica del backend e l’interfaccia con l’utente finale
* File Instructions.docx: il file corrente
* I file package.json che servono ad includere le dipendenze che richiede la WebApp all’avvio. Si spiegherà in seguito come eseguirla

# PER AVVIARE LA WEBAPP

* Installare Visual Studio Code: <https://code.visualstudio.com/Download>
* Aprire all’interno di VS Code la directory WebApp (File 🡪 Apri cartella)
* Aprire un nuovo terminale (Terminale 🡪 Nuovo terminale)
* All’interno del terminale, assicurarsi di essere nella directory contenente il file “server.js”
* Installare tramite terminale i package di “nodejs” e “npm”
* Digitare 🡪 npm install all per installare tutte le dipendenze
* Digitare 🡪 npm init e dare sempre l’invio
* Per avviare la WebApp digitare 🡪 node server.js
* VS Code può essere lasciato in esecuzione in background
* Ora è possibile visitare <http://localhost:3000> per utilizzare l’interfaccia grafica
* È inoltre possibile testare separatamente le API, essendo il server in stato di running
* È anche possibile inviare messaggi mqtt al Backend secondo il formato previsto

# UTENTI E SERVER KEYCLOAK

* L’applicazione utilizza un server Keycloak remoto
* Le informazioni per accedere alla console del server remoto sono disponibili nel file “server\_keycloak.txt” contenuto nella directory docs
* Gli utenti previsti sono otto: un agricoltore e un collaboratore per ognuna delle quattro aziende agricole previste
* Per accedere alla WebApp è possibile utilizzare quindi
  + Username: agricoltore\_X oppure collaboratore\_X (dove 1<=X<=4 && X appartiene a N)
  + Password: password

# TESTARE LE API

* Per effettuare chiamate alle API indicando il token nella richiesta è prima necessario richiamare l’API standard Keycloak per ottenere il token sul server il cui indirizzo è specificato nel file sopra citato
* Per ottenere l’access token si fa una API POST specificando nel body i campi che vediamo

Immagine che contiene testo

Descrizione generata automaticamente

* Il formato previsto dalle API è specificato nel file “Progettazione.docx” presente nella cartella docs
* Per chiamare una qualsiasi API è necessario specificare nell’header l’access token. Nel caso di API POST è necessario specificare il body come descritto nel file della progettazione

Immagine che contiene testo

Descrizione generata automaticamente

# TESTARE IL FRONTEND

* È possibile testare il frontend utilizzando l’interfaccia grafica sviluppata <http://localhost:3000>

# TESTARE L’EMULATORE

* Sportarsi con un terminale che possa eseguire il comando “java -jar …” nella cartella Emulatore del progetto
* Per avviare il simulatore di misure digitare quindi “java -jar publisher.jar”. Il programma leggerà dal db gli iot che sono sensori e genererà periodicamente per essi delle misure casuali. Le misure possono essere viste tramite uno strumento come MQTT lens, collegandosi al server di Alessandria e sottoscrivendo i topic specificati nella Progettazione. Inoltre, le misure vengono ricevute dal backend per essere salvate nel db ed eventualmente per far scattare le configurazioni automatiche.
* Per avviare il simulatore di attuatori digitare “java -jar subscriber.jar” e visualizzare lo stato degli attuatori sulla console. Il programma legge periodicamente lo stato degli attuatori dal db e rimane in ascolto sui topic predisposti per eventuali cambi di stato (comunicati dal backend a seguito di scatto configurazione automatica, oppure per via di messaggio pubblicato da MQTT Lens, o ancora sempre dal backend a seguito di gestione manuale attuatori del frontend da parte dell’utente)

**VISUALIZZARE STRUTTURA E DATI CONTENUTI NEL DB**

* Il database è sqlite. Il db è il file “iserra.db” contenuto nella directory WebApp del progetto. Tutti gli altri file “\*.db” sono dei backup
* È possibile installare SQLiteStudio <https://sqlitestudio.pl/> per aprire il file del db e visualizzarne contenuto e struttura, anche durante l’esecuzione della WebApp
* Forniamo inoltre il dump per la creazione del db nel file “iserra\_db\_dump.db” nella directory WebApp. Il dump non contiene le misure (si può utilizzare l’emulatore per popolare il db). Inoltre, vi forniamo un db già funzionante e completo, quindi il dump non dovrebbe essere necessario.

NB: Il server Keycloak (Lemur) e il server mqtt (Alessandria) sono microservizi esterni.