TEMPLATE DI PROJECT WORK = 3 CFU
min 12 pagine - max 20 pagine
**da compilare e caricare in formato pdf*

Cognome e Nome:	PERNA MARIO
Numero di Matricola:	0312201474
Corso di Studio: ◊ L-5 Filosofia ed Etica ◊ L-22 Scienze Motorie **X** L-31 Informatica per le Aziende Digitali	Barrare la casella riferita al proprio corso di studio
Tema n: **Titolo del tema:**	1 La digitalizzazione dell'impresa
Traccia del PW n: **Titolo della traccia:**	1.5 Sviluppo di una dashboard in JavaScript per l'analisi delle prestazioni aziendali nel settore primario
Titolo dell'elaborato:	DemetraStats: dashboard per l'analisi della produzione agricola
PARTE PRIMA – DESCRIZIONE DEL PROCESSO	
Utilizzo delle conoscenze e abilità derivate dal percorso di studio:	Nonostante abbia lavorato nel mondo IT per 5 anni, soprattutto in ambito React e NodeJS, grazie a questo progetto, ho avuto modo di mettere in campo le conoscenze, competenze ed abilità acquisite durante i corsi di studio qui di seguito riportati: 1. **TECNOLOGIE WEB - 0312212INF01IV:** per quanto concerne la parte di programmazione orientata al web, fornendo solide basi relativamente alla parte di HTML e CSS. Sulla base dei macroargomenti presentati, si sono andati ad estendere, facendo ampio uso del linguaggio JavaScript, i framework che sono stati impiegati nel presente progetto come React JS e NodeJS. 2. **INGEGNERIA DEL SOFTWARE - 0312212INGINF05:** per quanto riguarda la parte di progettazione del software, il ciclo di vita, l'organizzazione dei tempi e le modalità operative di realizzazione progettuale, nonché l'importanza delle best practices finalizzate ad una migliore usabilità dello stesso. 3. **CALCOLO DELLE PROBABILITÀ E STATISTICA - 0312209MAT06:** per la parte di generazione delle distribuzioni statistiche applicate ad una base di dati, al fine di generare delle oscillazioni degli stessi, quanto più vicini alla realtà.
Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato:	Il project work ha richiesto un'organizzazione del lavoro in fasi e una definizione dei tempi necessari per implementare tutte le azioni necessarie per la predisposizione dell'elaborato. Si riassumono le principali fasi di lavoro:

1. Analisi della traccia del project work: lettura attenta delle richieste, dei vincoli tecnologici da rispettare e scelta dell'ambientazione progettuale.

   Come ambientazione progettuale, è stato preso in considerazione il tema dell'agricoltura, tema che risulta essere in linea con le indicazioni della traccia selezionata.

   Per lo svolgimento di questa attività, non ho riscontrato particolari difficoltà.

   La durata complessiva di questa fase è stata di circa due giorni.

2. Ricerca in rete di dati afferenti la tematica individuata come: le tipologie di coltivazioni, i periodi di crescita del raccolto.

   Inoltre, sono stati oggetto di ricerca anche le condizioni atmosferiche come: temperatura ambientale, velocità del vento, precipitazioni e percentuale di umidità.

   Infine, si è andato a ricercare l'aspetto economico come: prezzo dell'energia, dell'acqua e delle materie prime da produrre.

   Al fine di contestualizzare le decisioni intraprese e le post-elaborazioni eseguite su tali informazioni, tengo a precisare che, sono stati oggetto di ricerca i dati afferenti al territorio dell' Emilia-Romagna, con particolare riferimento alla città di Bologna.

   I dati sono stati ricavati parzialmente dal sito https://open-meteo.com e successivamente analizzati e riadattati in modo personale.

   Per lo svolgimento di questa attività, ho riscontrato diverse difficoltà nella parte di ricerca ed analisi dei dati di storico, offerti da open-meteo, in quanto, il range preso in considerazione conteneva alcuni valori giornalieri vuoti che ho provveduto a colmare usando valori verosimili.

   La durata complessiva di questa fase è stata di circa 1 settimana.

3. Prototipazione e produzione dei modelli (mock-up grafico) della soluzione informatica elaborata.

   Sono stati implementati questi modelli, ovvero, i render del layout che è stato poi implementato per la realizzazione della dashboard, mediante l'uso di un sito web gratuito (excalidraw.com) che permette la prototipazione di un'idea completamente da zero.

   Di seguito si riportano alcune immagini dei layout realizzati:

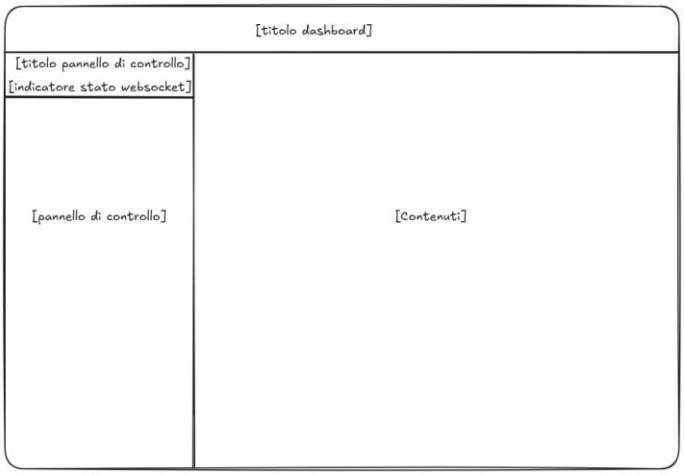

Figure 1 - Bozza layout base.

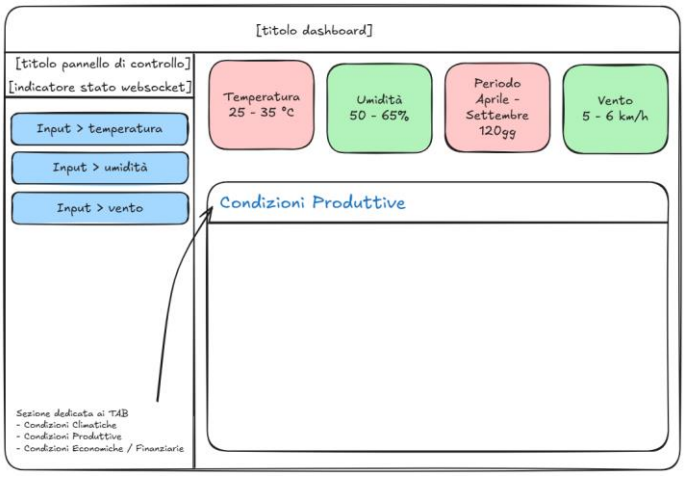

Figure 2 – A sinistra, il layout del pannello di controllo. A destra, il box e le tabs dei dati.

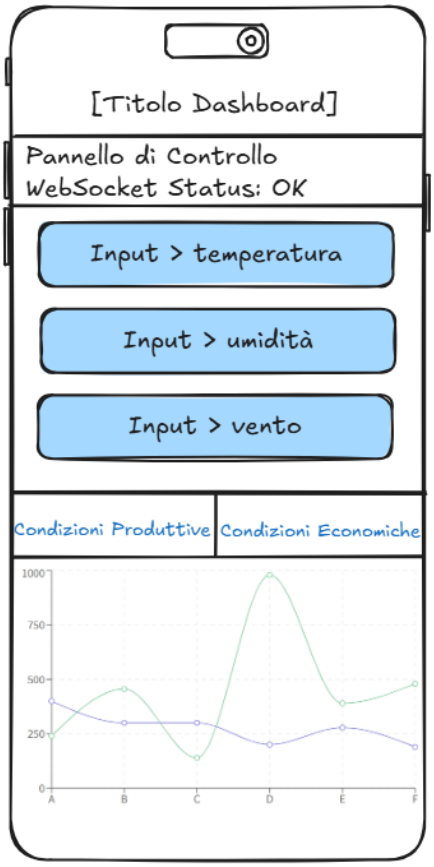

Figure 3 – Layout ottimizzato per i dispositivi mobili.

È stato deciso di svolgere questo passaggio intermedio al fine di agevolare le successive fasi di codifica, in modo tale da far emergere eventuali criticità di progettazione / usabilità permettendo così una correzione preventiva e non in corso d'opera.

Per lo svolgimento di questa attività, non ho riscontrato difficoltà.

La durata complessiva di questa fase è stata di circa due giorni.

4. Scelta delle tecnologie e delle strategie implementative

   Tra le tecnologie proposte sono state prese in considerazione:

   - Per la parte frontend, ovvero, la parte più vicina all'utente finale, l'uso di React JS: framework già noto in ambiente lavorativo, è stato arricchito da un framework di stile CSS denominato TailwindCSS.

     Questo framework permette una scrittura immediata del codice CSS, offrendo molte soluzioni dal design moderno e dalla codifica semplificata.

   - Per la parte di backend, che si occupa della parte di gestione, manipolazione dei dati è stato scelto NodeJS.

     Basato su sintassi JavaScript, offre la possibilità di esporre in modo facile e veloce, mediante anche l'utilizzo di Express JS, una REST API per servire il flusso dati elaborato per il frontend.

Inoltre, con l'aggiunta di alcuni pacchetti specifici, tra cui socket.io, è stato possibile esporre il medesimo oggetto come un WebSocket Server.

Questo ha permesso di rispettare il vincolo di simulazione dei dati in real-time.

Per lo svolgimento di questa attività non ho riscontrato particolari difficoltà.

La durata complessiva di questa fase è stata di circa una settimana.

5. Setup del progetto backend e frontend: uso di NPM (Node Package Manager) per la gestione ed installazione delle librerie, tra i quali:

   - Socket.io – per la parte di gestione delle WebSocket per entrambi i progetti (backend che frontend)

   - Concurrently – per la parte di installazione simultanea delle dipendenze necessarie per l'avvio del backend e del frontend.

     Inoltre, quest'ultimo, permette l'avvio simultaneo in locale di entrambi i progetti, semplificando la fase di "primo avvio".

   Per lo svolgimento di quest'attività ho riscontrato piccole difficoltà derivanti dall'uso delle WebSocket, in combinazione con il comando setInterval di JavaScript per l'invio temporizzato dei dati.

   La durata complessiva di questa fase è stata di circa 3 giorni.

6. Coding delle funzionalità operative della dashboard dati (parte backend e frontend).

   Per lo svolgimento di quest'attività ho riscontrato significative difficoltà nel re-mapping dei dati, provenienti dal backend, verso il componente messo a disposizione dalla libreria per la realizzazione dei grafici (recharts).

   La durata complessiva di questa fase è stata di circa 2 settimane (backend + frontend).

7. Deployment, Test e Bug discovery

   Il server è stato configurato usando la tecnologia Docker, tecnologia che ho avuto modo di conoscere e specializzarmi durante i miei cinque anni di permanenza in azienda, prima come sviluppatore software e successivamente come responsabile DevOps.

   Sono state preparate delle CI GitHub che permettono la generazione dell'immagine Docker. Queste immagini vengono salvate sul registry privato e gratuito offerto GitHub.

   Sul server acquistato, è attivo un servizio denominato "watchtower" che permette, ad intervalli di tempo predefiniti, la verifica di nuove immagini presenti sul registry. In caso affermativo, procede effettuando l'aggiornamento automatico dell'applicativo.

   Per lo svolgimento di quest'attività non ho riscontrato particolari difficoltà.

	Per quanto concerne la parte di test, ho avuto modo di svolgere qualche prova circa il funzionamento degli input, la responsività del layout e la verifica approssimativa dei dati prodotti. La durata complessiva di questa fase è stata di circa 3 giorni. Durante la fase di produzione del progetto, relativamente alla parte di codifica è stato intrapreso l'uso del noto sistema di versionamento "Git", coadiuvato dal gestore dei repository "GitHub". L'applicazione di questo sistema ha permesso: - Tracciabilità del codice e delle modifiche nel tempo. - Eventuale attività di revert, ovvero: l'annullamento dei cambiamenti introdotti da un commit precedente, creando un nuovo commit che "inverte" le modifiche senza alterare la cronologia del repository.
Risorse e strumenti impiegati:	**Risorse:** - Digitali: consultazione di alcuni siti web riguardante lo storico meteorologico, sull'agricoltura e sulle coltivazioni come: mais, grano, segale, grano tenero ed orzo. Le difficoltà riscontrate sono quelle dichiarate nella sezione *"Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato"* al punto *"Ricerca in rete di dati afferenti alla tematica individuata".* - Documentazioni ufficiali online per la parte di scrittura del codice JavaScript e delle librerie utilizzate per la parte di integrazione con le WebSocket, la gestione delle date e la realizzazione dei grafici. **Strumenti:** - Uso di editor di testo specifici per la programmazione, distribuiti open source, come Visual Studio Code. - Uso di tool online (Excalidraw) per la realizzazione di render grafici per la gestione dell'layout dell'elaborato. - Uso del CI/CD (Continuous Integration / Continuous Delivery), offerto da GitHub Action, per la gestione del deployment automatico dell'elaborato. - Uso dello strumento di versionamento "Git" unitamente a "GitHub" per la parte di persistenza del repository. **Eventuale bibliografia o link a risorse digitali Utilizzate** **Risorse online:** - Open-Meteo. (n.d.). *Storico dei dati meteorologici*. Open-Meteo. Recuperato il 2 novembre 2024, da https://open-meteo.com/ - Luxon. (n.d.). *Documentazione ufficiale di Luxon*. Recuperato il 3 novembre 2024, da https://moment.github.io/luxon/#/ - Remy Sharp. (n.d.). *Nodemon: Strumento per il riavvio automatico delle applicazioni Node.js*. Recuperato il 3 novembre 2024, da https://nodemon.io/ - React. (n.d.). *Documentazione ufficiale di React*. Recuperato il 4 novembre 2024, da https://react.dev/ - *Open CLI Tools. (n.d.). concurrently. GitHub. Recuperato il 4 novembre 2024, da https://github.com/open-cli-tools/concurrently*

- *Vite Contributors. (n.d.). create-vite: Setup React/Vite. GitHub. Recuperato il 15 novembre 2024, da https://github.com/vitejs/vite/tree/main/packages/create-vite*

- MUI. (n.d.). *Material UI: React UI library*. MUI. Recuperato il 18 novembre 2024, da https://mui.com/material-ui/

- Socket.IO. (n.d.). *Crea un'applicazione di chat con Socket.IO*. Recuperato il 23 novembre 2024, da https://socket.io/get-started/chat

- Fajib, F. (2021, agosto 17). *Setup proxy in Vite per React*. Medium. Recuperato il 28 novembre 2024, da https://medium.com/@faazfajib7/setup-proxy-in-vite-react-2eb1454bff62

- Tailwind Labs. (n.d.). *Setup Tailwind CSS with Vite*. Documentazione di Tailwind CSS. Recuperato il 5 dicembre 2024, da https://tailwindcss.com/docs/guides/vite

- Tailwind Labs. (n.d.). *Effetto CSS Ping per componente WebSocket*. Documentazione di Tailwind CSS. Recuperato il 5 dicembre 2024, da https://tailwindcss.com/docs/animation#ping

- MUI. (n.d.). *React Tabs: Componenti di navigazione con Material UI*. MUI. Recuperato il 16 dicembre 2024, da https://mui.com/material-ui/react-tabs/

- Recharts. (n.d.). *Libreria di grafici per React: Documentazione ufficiale*. Recuperato il 16 dicembre 2024, da https://recharts.org/en-US/

- i18next. (n.d.). *Utilizzo di i18n con React e hooks*. Documentazione ufficiale di i18next. Recuperato il 18 dicembre 2024, da https://react.i18next.com/latest/using-with-hooks

- Containrrr. (n.d.). *Watchtower: A process for automating Docker container base image updates*. Recuperato il 18 dicembre 2024, da https://github.com/containrrr/watchtower

- Docker, Inc. (n.d.). *Docker documentation*. Docker. Recuperato il 18 dicembre 2024, da https://docs.docker.com

- Bednar. (s.d.). Gestione dei residui colturali. Recuperato il 15 febbraio 2025, da https://www.bednar.com/it/gestione-dei-residui-colturali/

Motivi che hanno orientato la scelta delle risorse e degli strumenti.

Le risorse e gli strumenti sono stati selezionati considerando la loro reputazione nel settore e la loro idoneità agli obiettivi progettuali.

Si è tenuto conto della capacità di ciascuna risorsa o strumento di soddisfare i requisiti tecnici del progetto, come la qualità dei dati usati per le elaborazioni statistiche, l'uso di determinati editor per la segnalazione di eventuali errori/anomalie di scrittura nel codice e le automazioni, offerte dalle Action di GitHub, per la distribuzione del prodotto finale.

La disponibilità di documentazione ufficiale, tutorial e supporto dei forum è stata valutata per garantire una rapida integrazione ed apprendimento.

Modalità di individuazione e reperimento delle risorse e degli strumenti.

	Eventuali difficoltà affrontate e modo in cui sono state superate. Durante la realizzazione del progetto, si sono dovute affrontare una serie di criticità, tra le quali: - Integrazione di dati eterogenei: inviati in modalità real-time alla dashboard e con possibilità di alterazione degli stessi secondo alcuni parametri, controllabili e sovrascrivibili dalla UI da parte dell'utente. Questa parte è stata superata effettuando una post-elaborazione, lato frontend, dei dati provenienti dal backend, attraverso il canale di connessione client-server (WebSocket). L'utente, quindi, inserendo i valori negli opportuni campi di input ha modo di alterare i dati in ingresso. Questo meccanismo, fa si che tutti i dati provenienti dal backend, siano inalterati per ogni client che si collegherà al server. Sarà dunque l'utente finale, attraverso l'attribuzione di parametri variabili, a visualizzare in modo personalizzato i dati a lui significativi. - Problemi di CORS Origin in fase di deployment: questo problema è stato riscontrato in seguito all'esposizione del backend sotto un dominio differente da quello usato per il frontend. L'operazione di correzione che si è andata ad apportare è stata la modifica delle configurazioni di deployment affinchè l'esposizione del backend e del frontend fossero sotto un unico dominio. - Mapping dei dati da fornire ai componenti React per la generazione dei grafici (recharts). Questo è stato risolto attraverso la ricerca di esempi pratici estratti dalla documentazione ufficiale della libreria per la generazione dei grafici.
PARTE SECONDA – PREDISPOSIZIONE DELL'ELABORATO	
Obiettivi dell'elaborato/progetto/artefatto:	La soluzione informatica proposta, si pone l'obiettivo di permettere una gestione efficiente delle coltivazioni agricole, permettendo di intraprendere, ai decisori aziendali, azioni correttive per garantire quantità e qualità dei raccolti, nonché il perseguimento dei principî di massimizzazione finanziaria in termini di profitti e la riduzione dei costi di produzione. Il progetto aderisce interamente alle richieste tecnologiche e tematiche della traccia. Come già anticipato nella sezione ***"Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato"*** il progetto fa uso di React e NodeJS - entrambi originati dallo stack JavaScript. Gli obiettivi perseguiti dal seguente elaborato sono: - Produzione e manipolazione di dati che possano avvicinarsi al contesto reale. - Saper usare i dati prodotti per alimentare la dashboard. - Gestire il flusso dati prodotto in condizione di istantaneità (real-time).

	- Far interagire l'utente finale con la dashboard affinchè possa visualizzare la variazione dei dati in funzione del cambiamento di determinati parametri di input. - Visualizzare dati produttivi, condizioni ambientali e performance finanziaria. - Best practices in termini di usabilità da parte dell'utente finale e predisposizione ed adattamento ai multi-dispositivi (principio di responsività) – garantito dal framework TailwindCSS.
Contestualizzazione:	Il contesto operativo è quello di un'azienda agricola medio-piccola, produttrice di grano, mais, segale ed orzo nella città di Bologna. L'azienda opera in regime di B2B (Business to Business), ovvero, rivende i cereali ad altre aziende come pastifici e biscottifici i quali necessitano di forniture continue per evitare fermi produttivi. L'azienda, pertanto, necessita di ottimizzare i raccolti per incrementare il proprio fatturato e per farlo, si è dotata di **sensoristiche IOT** di ultima generazione applicate ai terreni, ai mezzi impiegati per la lavorazione della terra oltre che **sistemi di copertura del terreno domotico** e **stazioni meteo** per la rilevazione dei valori atmosferici. Nel caso in cui si verifichino delle condizioni avverse (temperatura / vento, rappresentate dai box di colorazione rossa), potrà attivarsi l'impianto automatico di copertura del terreno ed evitare l'irrigazione del campo in tarda serata per evitare il congelamento. Alternativamente, l'agricoltore, sulla base dell'analisi dei parametri riscontrati, potrà implementare tecniche di agricoltura come ad esempio il **mulching** che consiste nel coprire il terreno con uno strato di materiale organico o inorganico (paglia, compost, foglie secche) per aumentarne la temperatura ed abbassare il punto di congelamento. Invece, per quanto riguarda i dati raccolti dai sensori posti sui trattori verranno impiegati per collezionare le quantità di raccolto prelevato e generare varie metriche finanziarie come **costi** e **ricavi** che saranno trasmesse in real-time alla dashboard.
Descrizione dei principali aspetti progettuali:	**Riferimenti al Progetto** Al fine di rendere il codice disponibile per le valutazioni finali, è stato pubblicato, sul mio account personale GitHub, un repository pubblico denominato "**pw-l31-unipegaso**" e consultabile al seguente link: https://github.com/marioperna/pw-l31-unipegaso Inoltre, con la volontà di agevolare le operazioni di visualizzazione, test e valutazione dell'applicativo, ed evitare setup in locale dell'elaborato prodotto, ho provveduto all'acquisto di un server in cloud sul quale è stato effettuato il deployment dell'artefatto. Pertanto, il progetto è disponibile per la visualizzazione e la prova al seguente indirizzo web: https://pw-l31-unipegaso.marioperna.com Nel caso in cui si preferisse procedere con l'installazione in locale, la guida completa è disponibile alla sezione "**Installazione locale"** del presente documento. **Guida utente all'uso dell'applicativo** L'applicativo DemetraStats si avvia automaticamente alla prima visita.

Figure 4 - Dashboard

La dashboard è caratterizzata, sulla sinistra, dalla presenza di un pannello di controllo che permette di:

- Selezionare la coltivazione da analizzare
- Modificare i parametri di input, permettendo la generazione di dati personalizzati in funzione dei valori inseriti.
- Resettare i valori inseriti, ripristinando, di fatto, i valori originari, provenienti dal backend.
- Disconnettere la WebSocket per interrompere il flusso dati in regime di real-time.

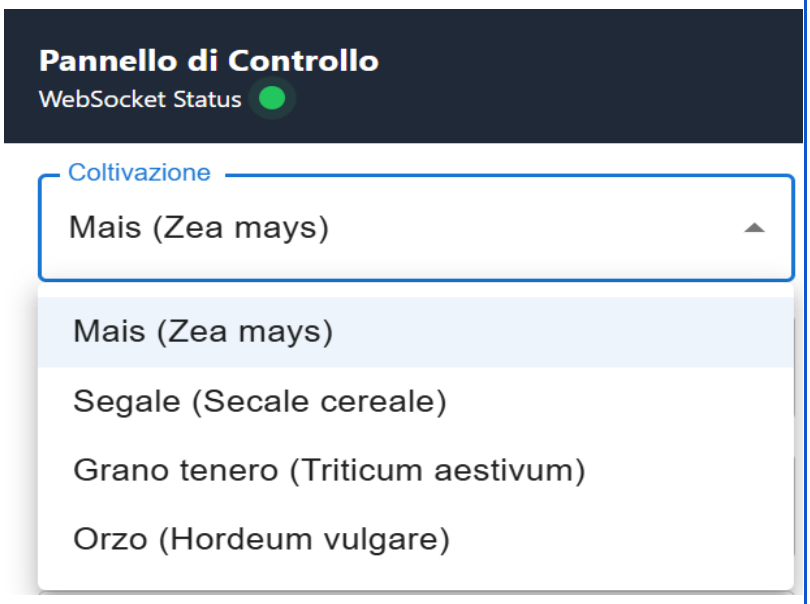

Figure 5 - Pannello di controllo: selezione della coltivazione.

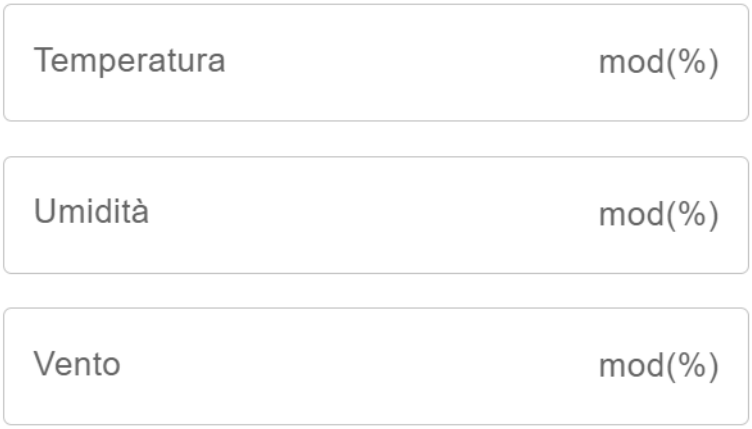

Figure 6 - Pannello di controllo, parametri per l'alterazione dei dati.

Sulla destra, invece, troviamo una parte superiore, formata da 4 box che indicano rispettivamente:

- Periodo consigliato per la semina ed il raccolto
- Temperatura
- Umidità
- Velocità del vento

Questi box, inoltre, assumono colorazioni differenti, tra cui:

- Rosso - quando la condizione non è verificata
- Verde - quando la condizione è verificata

Figure 7 - Box contenente i range di riferimento per la "buona crescita" del raccolto.

Le informazioni raccolte dai box sono la rappresentazione istantanea dei dati, ovvero, rappresentano l'ultima rilevazione inviata dal backend alla dashboard.

Nella sezione sottostante i box, troviamo tre tabs, che sono:

- **Condizioni Climatiche**: forniscono grafici e raccolgono le rilevazioni istantanee al fine di realizzare un tracciato.
  Come si può notare, sull'asse delle "x", troviamo l'indicazione temporale (t) della rilevazione effettuata, mentre, sull'asse delle "y", troviamo il valore registrato.

Figure 8 - Dashboard Tab: Condizioni climatiche.

- **Condizioni Produttive**: permette l'analisi in tempo reale delle quantità raccolte dai macchinari agricoli / risorse umane, il consumo di acqua e quello energetico (stimato) per la produzione di tale quantità, indicate dai box alla sinistra.

  Sulla destra, invece, vengono riportati, in forma grafica, i valori evidenziati dai box, permettendone una facile lettura visiva.

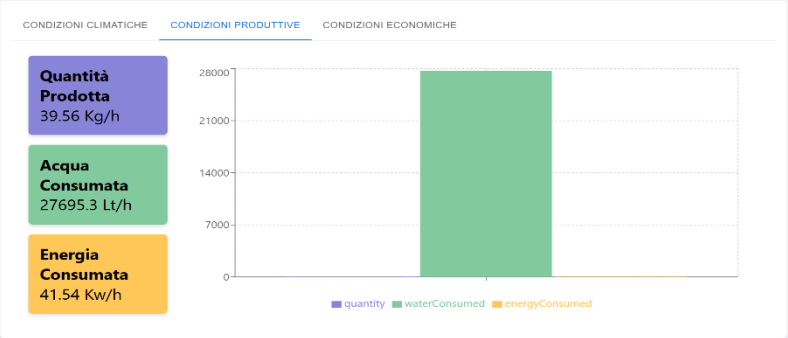

Figure 9 - Dashboard: Tab Condizioni Produttive.

- **Condizioni Economiche / Finanziarie:** permette di avere sotto controllo le performance finanziarie della produzione in tempo reale, sulla base dei dati presentati nel tab "Condizioni Produttive".

  L'etichetta "Mostra Saldi", se attivata, permette di visualizzare i saldi di ogni singola voce ed adattare il grafico presente.

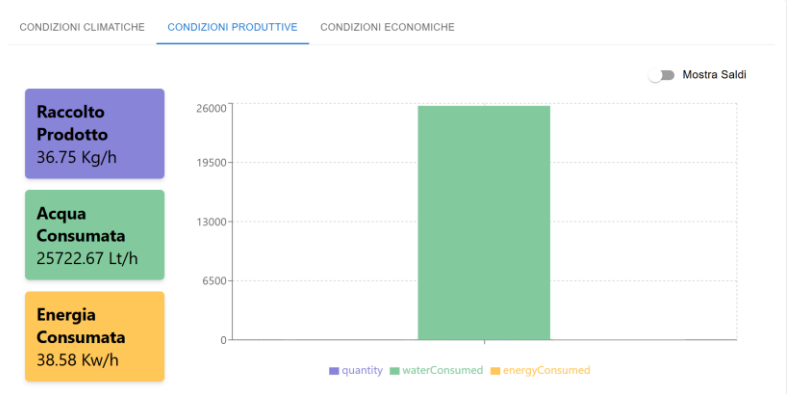

Figure 10 - Dashboard: Tab Condizioni Produttive.

Figure 11 - Dashboard: Tab Condizioni Produttive con l'applicazione della funzione "Mostra Saldi".

Installazione locale

Requisiti software
Node*: v22.11.0 – si consiglia di installare il software “nvm” (Node Version Manager) per gestire versioni differenti di Node su un medesimo dispositivo.

Procedura di installazione

1. Clonare il progetto da https://github.com/marioperna/pw-l31-unipegaso/tree/main
2. Nel caso si utilizzi nvm, eseguire il comando “**nvm use**”
3. Installare le dipendenze usando il comando: “**npm run install:all**”
4. Avviare il progetto usando il comando: “**npm run start:all**”

È possibile verificare che l’avvio sia avvenuto con successo, visitando:
- http://localhost:5173 per la dashboard frontend.
- http://localhost:3000 per il backend, con relativo messaggio di conferma “Backend successfully started”.

Approfondimento sulla struttura del progetto

La struttura del progetto è così composta:

```
frontend
├ public
│  ├ locales
│  │  ├ en
│  │  │  └ translation.json
│  │  └ it
│  │  │  └ translation.json
│  └ vite.svg
├ src
│  ├ assets
│  │  └ react.svg
│  ├ components
│  │  ├ ConnectionState.tsx
│  │  ├ ControlPanel.tsx
│  │  ├ CultivationStats.tsx
│  │  ├ CustomBarChart.tsx
│  │  ├ CustomLineChart.tsx
│  │  ├ CustomPosAndNegBarChart.tsx
│  │  ├ CustomTabPanel.tsx
│  │  └ DashboardTabs.tsx
│  ├ interfaces
│  │  └ tabpanel.interface.ts
│  ├ types
│  │  ├ common.d.ts
│  │  ├ cultivation.d.ts
│  │  └ line-chart.d.ts
│  ├ app.env.ts
│  ├ App.tsx
│  ├ i18n.tsx
│  ├ index.css
│  ├ main.tsx
│  ├ socket.tsx
│  ├ utilities.tsx
│  └ vite-env.d.ts
├ .gitignore
├ Dockerfile
├ eslint.config.js
```

	├ index.html ├ nginx.conf ├ package-lock.json ├ package.json ├ postcss.config.js ├ README.md ├ tailwind.config.js ├ tsconfig.app.json ├ tsconfig.json ├ tsconfig.node.json └ vite.config.ts Si descrivono le principali cartelle / files che meritano menzione per il corretto funzionamento dell'elaborato: - public > locales: troviamo i file per le traduzioni usati in combinazione con il pacchetto i18n. - src > components: sono presenti i componenti utilizzati nella pagina App.tsx che permettono la riusabilità del codice. - src > interfaces: contiene i file per la gestione della tipizzazione delle interfacce applicate ai componenti. - src > types: costituisce l'insieme dei "tipi" generati ed implementati nei vari componenti e nella pagina di progetto. - App.tsx: è il cuore pulsante dell'applicazione. Ingloba tutti i componenti necessari per funzionare ed è il punto di collegamento del backend e la WebSocket. - package.json / package.lock: sono i descrittori delle dipendenze contenute nel presente progetto. Sono responsabili dell'installazione delle versioni corrette e dello scaricamento delle dipendenze di progetto. - Dockerfile: è un file che descrive la creazione di immagini Docker, definendo istruzioni per configurare un ambiente containerizzato. backend ├ cultivation-indicators.json ├ Dockerfile ├ historical-data.json ├ index.js ├ package-lock.json ├ package.json ├ utilities.js └ websocket-cmd.js Come per il frontend, si descrivono le principali cartelle / files del backend, che meritano menzione per il corretto funzionamento dell'elaborato: - cultivation-indicators.json: contiene il set di indicatori ottimali per ogni coltivazione. - Dockerfile: si rimanda alla definizione enunciata nella sezione frontend del medesimo punto. - historical-data.json: sono i dati statistici recuperati da open-meteo.com che poi vengono alterati dal backend con un delta di oscillazione. - package.json / package.lock: si rimanda alla definizione enunciata nella sezione frontend del medesimo punto. - utilities.js: contiene funzioni di utilità generate ad hoc per l'intero progetto.

	- websocket-cmd.js: contiene i nomi degli eventi che il WebSocket server può emettere.
Campi di applicazione:	Il presente elaborato trova il suo punto di applicazione nel **settore primario** e nello specifico, nel **settore agricolo**. L'implementazione del presente prodotto apporta notevoli vantaggi riscontrabili nella possibilità di **ottimizzare in modo efficace le coltivazioni**, **eseguire previsioni su costi e ricavi** ed **offrire un set di dati comparativi** per poter interpretare le **opportunità di crescita aziendale** nel breve e lungo periodo.
Valutazione dei risultati (potenzialità e criticità):	Tra le potenzialità offerte da questo elaborato è possibile riscontrare, oltre ai punti già dichiarati nella sezione "campi di applicazione", l'**adattabilità al mercato** con previsioni della domanda e ottimizzazione della produzione oltre ad una **maggiore sostenibilità ambientale** grazie a una gestione efficiente delle risorse. Tra gli aspetti critici, invece, è possibile notare come non vengano prese in considerazione le **spese accessorie**, come: i costi da sostenere per la **manutenzione dei macchinari, i concimi e gli stipendi del personale**. Inoltre, la **qualità delle elaborazioni statistiche è limitata** dall'esiguo campione di dati, circoscritto a una sola città e a un ridotto periodo di tempo, non garantendo l'affidabilità delle proiezioni climatiche ottenute. Dal punto di vista previsionale, è possibile riscontrare come il prodotto consideri solo alcune variabili atmosferiche (temperatura, vento ed umidità), mentre un'analisi più accurata richiederebbe **ulteriori parametri ambientali**, come pH del suolo, radiazione solare e drenaggio. Infine, i **costi elevati di installazione** circa l'acquisto di attrezzature utili per la rilevazione dei dati o la protezione del raccolto (sensoristiche IOT, tensostrutture domotiche e stazioni meteo) potrebbero rappresentare un ostacolo all'adozione del prodotto.