

GWMS: Gas Warehouse Management System

Progetto per gestione impianti refill GAS (cliente iniziale: Pizzaferrri tramite W.I.L.)

Descrizione generale

il sw si occupa di

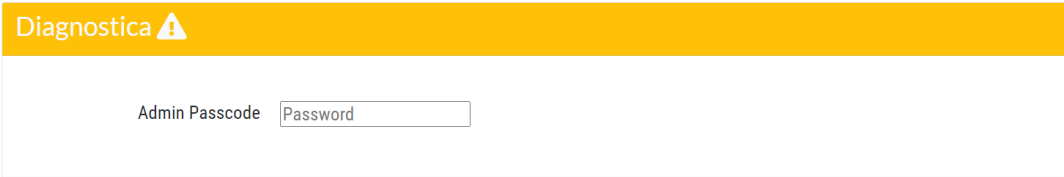
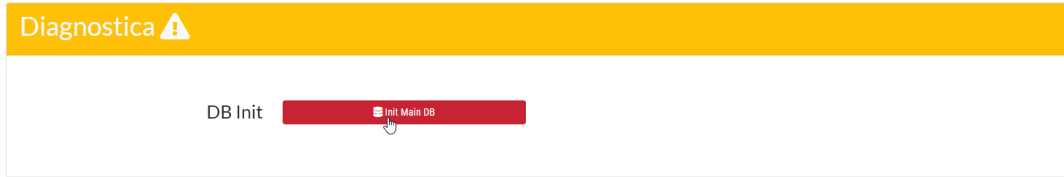
- visualizzare stato impianti distribuzione GNL
- visualizzare storico eventi registrati
- visualizzare storico allarmi
- visualizzare ordini di refill generati in automatico
- configurare parametri ammissibili epr il sistema
- configurare il piano consegne settimanale
- verificare gli ordini aperti
- permettere ai fornitori di verificare gli ordini aperti
- permettere ai trasportatori di mostrare gli ordini in consegna
- permettere ai singoli plant di registrare le consegne (inizio/fine) degli ordini aperti

Setup iniziale

Il sistema, in caso di mancanza dei DB, propone di caricarli (ed eventualmente anche di simulare i dati da msotare ina ttesa della vera acquisizioend al campo)

Per funzionare si procede in questo modo

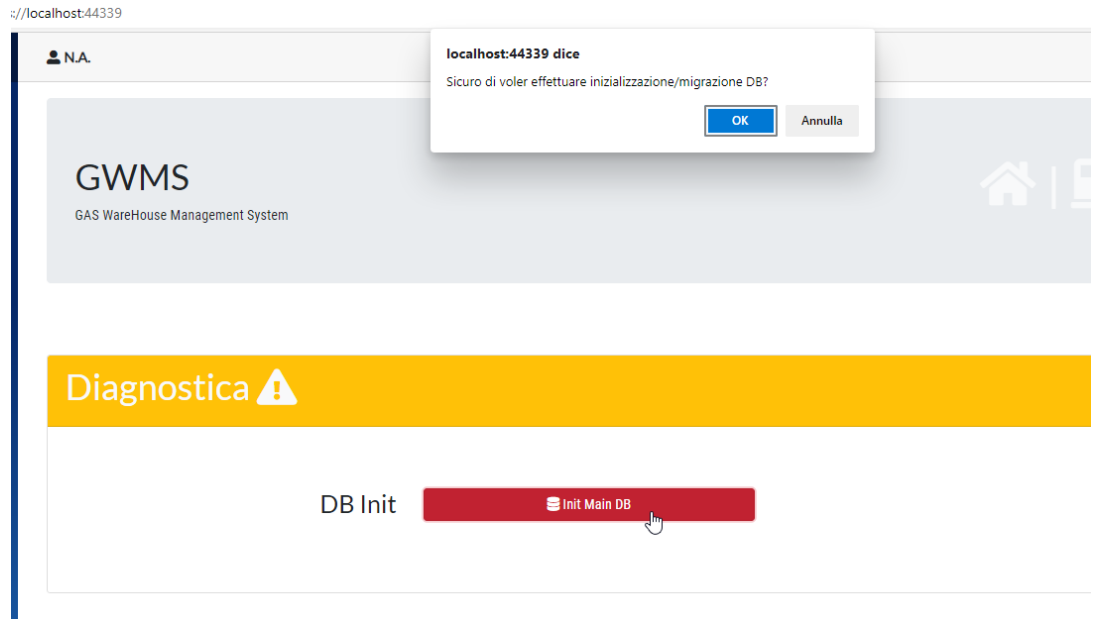
- login sulla pagina iniziale (es: <http://gwms.egalware.com>, oppure sul server test interno <http://10.74.83.98>)
- il sistema trova mancanza DB, richiede una passphrase (NON persistente...) --> inserire **f@mmiEntrare!**
- a questo punto la procedura guidata fa creare da init il DB di base

step	Image
passphrase: f@mmiEntrare!	
Init Main DB	

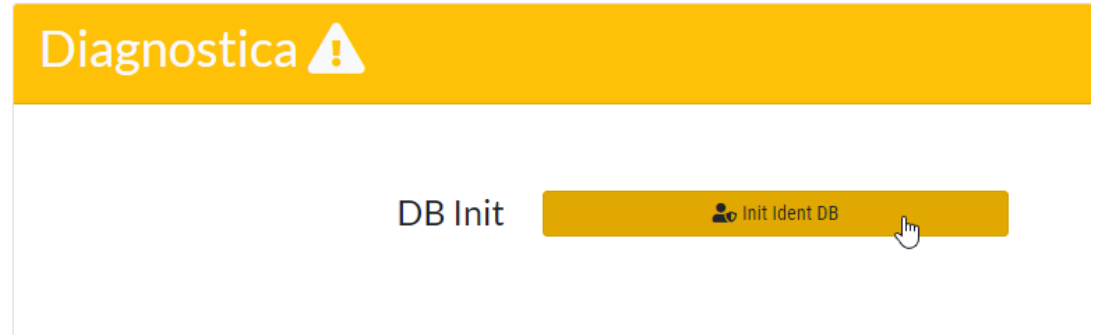
step

Image

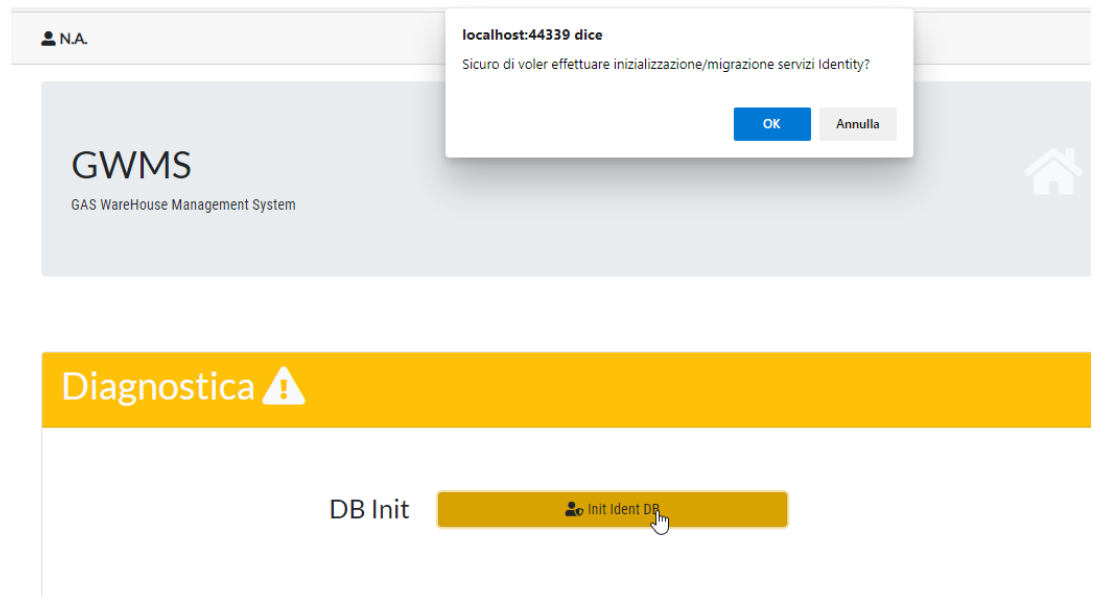
Conferma Init DB



Init Auth DB



Conferma Init DB



step	Image
Init dati SIM (opzionale)	

Note / integrazioni

Nel progetto è stato integrato un set di funzionalità standard/di abse di asp.net core tra cui

- auth management
- identity management
- role management
- librerie client

Identity management

Integrazione librerie font-awesome

Con riferimento a questo link <https://www.kevinwilliams.dev/adding-fontawesome-to-blazor/>

SI è usato

- add client side library
- modifica file _host.cshtml

```
<head>
...
  <link rel="stylesheet" href="font-awesome/css/fontawesome.css" />
  <link href="css/site.css" rel="stylesheet" />
...
</head>
<body>
...

  <script src="font-awesome/js/all.js"></script>
  <script src="_framework/blazor.server.js"></script>
</body>
```

Flusso Operativo

...

Acquisizione dati dal campo

...

Gestione Logistica

...

Modbus

Il sistema prevede di dover leggere i dati tramite MODBUS-TCP dagli impianti remoti

- ipotesi ideale: impiegare un cliente dotnetcore, richiamabile da cron o su richiesta interattiva, per chiamate remote e poi pubblicare sulla app un update dei valori rilevati (stile MP-IO)
- implementazione ideale C# dotnet core (crossplatform), accettabile c# windows only o python/java

link da approfondire

- <https://www.nuget.org/packages/Modbus.Net.Core/>
- <https://github.com/parallelbgl/Modbus.Net>
- <http://easymodbustcp.net/en/>
- <https://sourceforge.net/p/easymodbustcp/wiki/Home/>
- <https://techtimore.com/dotnet/modbus-tcp-communication-in-net-c/>

Appunti

Linux debug

Questa applicazione viene distribuita (localmente in ufficio ed in ovh x cliente finale) su macchina Linux, tramite dotnetcore.

In particolare su linux x debug dei messaggi di log usare il comando

```
journalctl -fu kestrel-app-gwms.service
```

Versioni