|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  | DOCUMENTAZIONE Guida completa all'installazione, configurazione e utilizzo di VertiStock per la gestione automatizzata del magazzino verticale. |
|  |  |

## INDICE:

Introduzione………………………………………………………………………………………………………………………………………...............3

Requisiti di sistema……………………………………………………………………………………………………………………………………….…5

Ambiente di sviluppo……………………………………………………………………………………………………………………………………….6

Librerie e dipendenze…..…………………………………………………………………………………………………………………………………8

## 1. INTRODUZIONE

#### 1.1 Descrizione Generale del Progetto:

VertiStock è un software innovativo per la gestione di magazzini verticali automatizzati, sviluppato per offrire qualità, velocità e precisione nelle operazioni di stoccaggio e prelievo. La piattaforma è progettata per essere accessibile da PC, tablet e dispositivi mobili, consentendo agli operatori di monitorare e controllare l’inventario in tempo reale. Grazie a un’interfaccia semplice e intuitiva, VertiStock rende la gestione del magazzino più efficiente, riducendo errori e ottimizzando il flusso di lavoro.

#### 1.2 Obiettivi e Funzionalità Principali:

L’obiettivo principale di VertiStock è migliorare la gestione dei magazzini verticali attraverso un sistema affidabile e automatizzato, che garantisce:

Ottimizzazione dello spazio di stoccaggio, riducendo ingombri e sprechi.

Automazione dei processi di prelievo e rifornimento, aumentando la velocità delle operazioni.

Monitoraggio in tempo reale dell’inventario, con tracciamento preciso dei prodotti.

Riduzione degli errori umani, grazie a un sistema intelligente e interconnesso.

Analisi avanzata dei dati, con reportistica dettagliata per migliorare la gestione logistica.

#### 1.3 Strumenti e Tecnologie Utilizzate:

VertiStock è sviluppato con tecnologie moderne e affidabili, progettate per garantire elevate prestazioni, scalabilità ed efficienza nella gestione dei magazzini verticali automatizzati.

|  |  |  |
| --- | --- | --- |
|  |  |  |

* Frontend:
  + Interfaccia desktop realizzata con Electron e Laravel Mix, offrendo un’esperienza utente fluida e intuitiva.
  + Applicazioni mobile per iOS e Android, sviluppate con Flutter e Dart, per una gestione cross-platform accessibile ovunque.
* Backend:
  + Il cuore del sistema è Arduino Rev4 WiFi, che gestisce la comunicazione e il controllo dei dispositivi, assicurando stabilità ed efficienza operativa.

|  |
| --- |
|  |

* Database:
  + Archiviazione locale su SD card dell’Arduino, garantendo un sistema sicuro, autonomo e indipendente da connessioni cloud;

|  |
| --- |
|  |

* AI e Automazione:
  + Integrazione con API DeepSeek, sfruttando intelligenza artificiale avanzata per ottimizzare lo stoccaggio, prevedere la domanda e migliorare la gestione dell’inventario;

|  |
| --- |
|  |

* Integrazione IoT:
  + Connessione con ESP32 e M5Stack Core Basic, abilitando il monitoraggio in tempo reale e la gestione automatizzata della movimentazione nel magazzino;

Grazie a questa combinazione di tecnologie, VertiStock offre una soluzione innovativa, scalabile e altamente efficiente, garantendo precisione, velocità e affidabilità nella gestione dei magazzini verticali automatizzati.

## 2. REQUISITI DI SISTEMA

#### 2.1 Requisiti hardware:

Per garantire il corretto funzionamento di VertiStock, è necessario rispettare una serie di requisiti hardware e software per ogni componente del sistema.

VertiStock richiede i seguenti dispositivi per la gestione del magazzino verticale:

* Arduino Rev4 WiFi – Unità centrale di controllo per la gestione delle operazioni di movimentazione e archiviazione dati;
* ESP32 e M5Stack Core Basic – Moduli IoT per il monitoraggio in tempo reale e il controllo degli attuatori;
* PC Desktop o Laptop – Necessario per l’utilizzo dell’interfaccia Web App e il monitoraggio avanzato;
* Smartphone o Tablet (iOS/Android) – Per l’utilizzo dell’App Mobile, consentendo la gestione del magazzino da remoto;
* MicroSD Card – Utilizzata come memoria per l’archiviazione dei dati dell’inventario sull’Arduino;

#### 2.2 Requisiti software per ogni componente:

* Arduino Rev4 WiFi e M5Stack Core Basic:
  + Firmware: Programmazione basata su C++ (Arduino IDE) con librerie per il controllo del magazzino;
  + Comunicazione: Supporto per WiFi (WiFiS3) per la connessione con la Web App e i dispositivi IoT;
  + Archiviazione: Lettura/scrittura dati sulla SD card per il salvataggio locale dell’inventario;
* Web App (Laravel Mix & Node.js):
  + Sistema operativo: Windows, MacOS o Linux;
  + Node.js per il backend e la gestione delle API.
  + Electron per l’applicazione desktop.
  + Bonjour per API di connessione;
* App Mobile (Flutter/Dart):
  + Sistema operativo: iOS (14+) e Android (8+);
  + Framework: Flutter 3.x con linguaggio Dart;
  + Connessione WiFi per la comunicazione con Arduino e i dispositivi IoT;

## 3. AMBIENTE DI SVILUPPO

### 3.1 Configurazione dell'Ambiente Arduino IDE e VS Code:

Installazione di Arduino IDE per Arduino Rev4 Wifi:

* Scaricare e installare Arduino IDE da [Arduino Download](https://www.arduino.cc/en/software);
* Aprire Arduino IDE e installare il supporto per Arduino Rev4 WiFi;
* Gestore schede → Cercare Arduino UNO R4 WiFi → Installare;
* File → Preferenze → URL aggiuntive per il gestore di schede → https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json;
* Installare le librerie necessarie:
  + WiFi (per la connessione WiFi);
  + SD e SPI (per la gestione della scheda SD);
  + ArduinoJson (per la comunicazione dati);
  + ArduinoMDNS e WifiUDP (configurazione comunicazione DNS);

Installazione di Arduino IDE per M5Stack Core Basic:

* Aprire Arduino IDE e installare il supporto per M5Stack;
* Gestore schede → Cercare M5Stack → Installare;
* File → Preferenze → URL aggiuntive per il gestore di schede → https://static-cdn.m5stack.com/resource/arduino/package\_m5stack\_index.json;
* Installare le librerie necessarie:
  + M5Stack (pacchetto per programmare controllori M5Stack);
  + ArduinoMDNS e (configurazione comunicazione DNS);

Configurazione di VS Code:

* Installare Visual Studio Code da [VS Code Download](https://code.visualstudio.com/);

Installazione e Configurazione di Node.js, npm e Laravel Mix:

* Necessario installare e configurare Node.js, npm e Laravel Mix:
  + Scaricare l’ultima versione LTS di Node.js da [Node.js Download](https://nodejs.org/);
  + Verificare l’installazione con i comandi;

node -v

npm -v

* Configurazione del progetto con Laravel Mix:
  + Creare una cartella di progetto ed eseguire;

npm init -y

npm install laravel-mix –-save-dev

* Configurare il file webpack.mix.js per gestire gli asset del frontend;
* Eseguire il comando per compilare le risorse:

npx mix watch

Configurazione dell’Ambiente di Sviluppo Flutter:

* Scaricare e installare Flutter SDK da [Flutter Download](https://flutter.dev/);
* Aggiungere Flutter al PATH alle variabili di sistema;
* Controllare l’installazione con:

flutter –-version

* Installazione di Android Studio per Android;
* Installazione di Xcode per iOS;
* Controllare l’installazione con:

flutter doctor

## 4. LIBRERIE E DIPENDENZE

### 4.1 Librerie per Arduino Rev4 WIFI:

Librerie base per Arduino Rev4 WIFI:

#include <Arduino.h>

Questa è la libreria principale di Arduino che fornisce le funzioni di base per la programmazione di microcontrollori, come digitalWrite(), analogRead(), delay(), ecc.

Librerie per la comunicazione di rete e WiFi:

#include <WiFi.h>

Permette la connessione WiFi per schede basate su ESP32 e altre con supporto WiFi. Include funzioni per la gestione della rete, come la connessione a un SSID, l'assegnazione di un IP e la gestione delle connessioni client-server.

#include <WiFiUDP.h>

Consente la comunicazione tramite il protocollo UDP (User Datagram Protocol) su reti WiFi. È utile per applicazioni che richiedono trasmissione veloce di pacchetti dati, come la telemetria o il controllo remoto.

#include <ArduinoMDNS.h>

Implementa il supporto per mDNS (Multicast DNS), permettendo ai dispositivi di essere raggiunti tramite un hostname locale invece di un indirizzo IP statico. Ad esempio, un dispositivo con mydevice.local può essere trovato nella rete senza conoscere il suo IP.

Librerie per la comunicazione con periferiche:

#include <Wire.h>

Gestisce la comunicazione I2C (Inter-Integrated Circuit), utilizzata per collegare dispositivi come sensori, display e altri componenti elettronici.

#include <SPI.h>

Gestisce la comunicazione tramite il protocollo SPI (Serial Peripheral Interface), utilizzato per connettere il microcontrollore con dispositivi come memorie SD, display e sensori.

Librerie per gestione di dati e archiviazione:

#include <SD.h>

Fornisce il supporto per schede SD, consentendo di leggere e scrivere file su memorie SD tramite interfaccia SPI.

#include <ArduinoJson.h>

Libreria per la manipolazione di dati in formato JSON (JavaScript Object Notation). Utile per strutturare e scambiare dati in modo leggibile e compatto, ad esempio in comunicazioni IoT.

Librerie per gestione del display OLED:

#include <Adafruit\_GFX.h>

Libreria grafica di Adafruit per disegnare testi, forme e immagini su display grafici, come quelli OLED e TFT.

#include <Adafruit\_SSD1306.h>

Libreria specifica per i display OLED basati sul driver SSD1306. Fornisce funzionalità per disegnare testi, immagini e aggiornare lo schermo in modo efficiente tramite I2C o SPI.

### 4.2 Librerie per M5Stack Core Basic:

Librerie base per M5Stack Core Basic:

#include <M5Stack.h>

Questa è la libreria principale di M5Stack che fornisce le funzioni di base per la programmazione del microcontrollore.

#include <ArduinoMDNS.h>

Implementa il supporto per mDNS (Multicast DNS), permettendo ai dispositivi di essere raggiunti tramite un hostname locale invece di un indirizzo IP statico. Ad esempio, un dispositivo con mydevice.local può essere trovato nella rete senza conoscere il suo IP.

### 4.3 Dipendenze di Node.js:

DevDependencies (Dipendenze di sviluppo):

laravel-mix (^6.0.49)

Strumento per compilare e ottimizzare risorse web in Laravel, basato su Webpack.

resolve-url-loader (^5.0.0)

Aiuta Webpack a risolvere correttamente i percorsi relativi nelle risorse CSS e Sass.

sass (^1.77.5)

Compilatore per il preprocessore CSS Sass/SCSS.

sass-loader (^12.1.0)

Loader per Webpack che permette di compilare file .scss e .sass in CSS.

Dependencies (Dipendenze di produzione):

bootstrap (^5.3.3)

Framework CSS per la creazione di interfacce web responsive e moderne.

fs(^0.0.1-security)

Pacchetto obsoleto che probabilmente non è necessario, dato che Node.js ha un modulo nativo fs per la gestione dei file.

lottie-web (^5.12.2)

Libreria per animazioni JSON-based create con Adobe After Effects e Bodymovin.

path-browserify (^1.0.1)

Versione per browser del modulo path di Node.js, utile per gestire percorsi di file.

three (^0.171.0)

Libreria JavaScript per la creazione e gestione di grafica 3D basata su WebGL.

altre:

axios (^1.3.5)

Client HTTP per effettuare richieste API in modo semplice ed efficace.

express (^4.18.2)

Framework minimalista per creare server web e API REST in Node.js.

jsonwebtoken (^9.0.0)

Libreria per la creazione e la verifica di token JWT (JSON Web Token), utile per l'autenticazione.

moment (^2.29.4)

Libreria per la gestione e la manipolazione delle date.

mongoose (^7.1.0)

ODM (Object Data Modeling) per MongoDB, semplifica la gestione del database.

nodemon (^2.0.22)

Strumento che riavvia automaticamente l'app quando vengono rilevate modifiche ai file.

socket.io (^4.7.1)

Libreria per la comunicazione in tempo reale basata su WebSockets.

uuid (^9.0.0)

Generatore di UUID (Universally Unique Identifier), utile per identificatori unici.

### 4.4 Dipendenze di Flutter:

Dipendenze principali:

flutter

SDK ufficiale per lo sviluppo di app multipiattaforma con Flutter.

cupertino\_icons (^1.0.8)

Set di icone in stile iOS utilizzabili nei progetti Flutter.

UI e Grafica:

flutter\_svg (^2.0.17)

Supporta il rendering di immagini in formato SVG in Flutter.

lottie (^3.3.1)

Permette di visualizzare animazioni Lottie create con Adobe After Effects.

Scanner e Permessi:

mobile\_scanner (^6.0.3)

Libreria per la scansione di codici a barre e QR Code in tempo reale.

permission\_handler (^11.3.1)

Gestisce le richieste di permessi di sistema per fotocamera, memoria, GPS, ecc.

Rete e Comunicazione:

http (^1.3.0)

Permette di inviare richieste HTTP (GET, POST, ecc.) all'interno dell'app.

multicast\_dns (^0.3.2+7)

Implementa il protocollo mDNS per individuare dispositivi nella rete locale.

Calcolo e Risoluzione di Equazioni:

dnsolve (^1.0.0)

Libreria per la risoluzione di equazioni differenziali.

**Configurazione e Personalizzazione:**

flutter\_launcher\_icons **(**^0.14.3**)**

Permette di generare automaticamente le icone dell'app per Android e iOS.

**DevDependencies (Solo per sviluppo e testing):**

flutter\_test

SDK di Flutter per scrivere test automatici.

flutter\_lints **(**^5.0.0**)**

Regole di linting per mantenere il codice più pulito e conforme agli standard Flutter.

## 5. REPOSITORY

Contenuto software (<https://github.com/norcenw/Software-VS>):

* Codice Arduino;
* Codice M5stack Core Basic;
* Codice App Desktop;
* Codice App Mobile;
* Schema a blocchi delle chiamate http;

Contenuto Elettrica (<https://github.com/norcenw/Elettrica-VS>):

* Schema a Blocchi;
* Schema elettrico;
* Normative elettriche;

Contenuto Meccanica (<https://github.com/norcenw/Meccanica-VS>):

* Dimensionamenti di componenti a rischio;
* Assiemi 3D in CAD;
* Messe in tavola;
* File word di test di sforzo simulati con SolidWork;

Tutta la documentazione del progetto: <https://github.com/norcenw/Documentazione-VS>

Versionamento app desktop e mobile: <https://github.com/norcenw/VertiStock-Setup-1.0.0>