

**TEMPLATE
PROJECT WORK**

Corso di Studio	INFORMATICA PER LE AZIENDE DIGITALI (L-31)
Dimensione dell'elaborato	Minimo 6.000 – Massimo 10.000 parole <i>(pari a circa Minimo 12 – Massimo 20 pagine)</i>
Formato del file da caricare in piattaforma	PDF
Nome e Cognome	Alex Filippini
Numero di matricola	0312301747
Tema n. (Indicare il numero del tema scelto):	1
Titolo del tema (Indicare il titolo del tema scelto):	La digitalizzazione dell'impresa
Traccia del PW n. (Indicare il numero della traccia scelta):	11
Titolo della traccia (Indicare il titolo della traccia scelta):	Lotto Economico di Ordinazione (EOQ) e del Livello di Riordino con Scorta di Sicurezza (Safety Stock)
Titolo dell'elaborato (Attribuire un titolo al proprio elaborato progettuale):	Il ruolo del Safety stock nella resilienza aziendale

PARTE PRIMA – DESCRIZIONE DEL PROCESSO

Utilizzo delle conoscenze e abilità derivate dal percorso di studio

(Descrivere quali conoscenze e abilità apprese durante il percorso di studio sono state utilizzate per la redazione dell'elaborato, facendo eventualmente riferimento agli insegnamenti che hanno contribuito a maturarle):

Lo sviluppo di questo progetto ha tratto notevole beneficio da una varietà di insegnamenti universitari che hanno fornito le fondamenta teoriche, metodologiche e tecniche necessarie. Di seguito sono descritti i principali contributi per ogni area di competenza.

Il corso di *Strategia, organizzazione e marketing*, a cura del Prof. Manuel Cavola e del Prof. Stefano Abbate, ha fornito il contesto organizzativo e strategico per comprendere l'importanza della gestione dell'inventario all'interno della catena di approvvigionamento complessiva approfondendo tematiche come:

- La natura e gli obiettivi della gestione dell'inventario
- Le diverse tipologie di inventario (materie prime, semilavorati, prodotti finiti)
- La relazione tra gestione dell'inventario e altri processi della supply chain
- La valutazione delle performance di inventario attraverso KPI appropriati
- L'equilibrio tra costi di inventario e livello di servizio ai clienti

Il corso di *Calcolo delle probabilità e statistica*, a cura del Prof. Sergio Frigeri, ha fornito gli strumenti necessari a modellare serie temporali complesse, identificando stagionalità, trend e cicli, e nello specifico:

- Comprendere e calcolare la deviazione standard come misura di variabilità

- Lavorare con distribuzioni di probabilità, in particolare la distribuzione normale
- Utilizzare i quantili (Z-score) per determinare livelli di servizio
- Analizzare serie storiche di domanda e identificare componenti di trend, stagionalità e noise
- Applicare tecniche di forecasting e analisi predittiva ai dati storici

Il corso di *Ingegneria del software*, a cura del Prof. Fabiano Pecorelli, del Prof. Massimiliano Pirani e del Prof. Roberto Vergallo, ha fornito l'insieme delle teorie, dei metodi e delle tecniche impiegate nello sviluppo industriale del software e in particolare dettagli su come applicare le seguenti conoscenze scientifiche e tecnologiche:

- Ciclo di vita del software (SDLC) come processo strutturato per organizzare il lavoro in fasi ben definite
- Metodi di gestione del progetto sviluppato entro i tempi e i costi preventivati
- Design pattern e loro applicazione al fine di garantire manutenibilità ed espandibilità

Il corso di *Tecnologie web*, a cura del Prof. Stefano D'Urso, ha fornito l'insieme delle teorie, dei metodi e delle tecnologie impiegate nello sviluppo di soluzioni web-based, e nello specifico:

- Modello OSI
- Protocollo HTTP e HTTPS
- Formati standard quali XML e JSON
- Linguaggi per la costruzione di front-end web quali HTML, CSS e Javascript

Il corso di *Programmazione 1*, a cura del Prof. Giuseppe De Pietro, e il corso di *Programmazione 2*, a cura del Prof. Massimo Esposito e del Prof. Claudio Tomazzoli, hanno fornito tutti gli elementi necessari alla progettazione di algoritmi e loro traduzione in linguaggio di programmazione e in particolare:

- Fondamenti di programmazione
- Concetti di programmazione strutturata e relativi costrutti (Sequenza, Selezione e Iterazione)
- Tipi di dati e relativa aritmetica

Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato

(Descrivere le attività svolte in corrispondenza di ciascuna fase di redazione dell'elaborato. Indicare il tempo dedicato alla realizzazione di ciascuna fase, le difficoltà incontrate e come sono state superate):

Il progetto è stato strutturato seguendo le fasi standard del Software Development Life Cycle (SDLC), con particolare attenzione alla pianificazione, progettazione, implementazione e validazione, come descritto di seguito.

Fase 1: Analisi dei Requisiti (Settimana 1-2)

In questa fase iniziale sono stati definiti i requisiti funzionali e non-funzionali dell'applicazione:

- Studio approfondito della letteratura sulla EOQ e sulla gestione della scorta di sicurezza
- Identificazione dei dati di input necessari (domanda annuale, costi, parametri statistici)
- Definizione del formato e della chiarezza dei dati di output
- Specifica dei livelli di servizio supportati (es. 90%, 95%, 99%)
- Valutazione delle performance richieste (velocità di calcolo, capacità di elaborazione)

Fase 2: Progettazione dell'Architettura (Settimana 3)

La fase di progettazione ha definito l'architettura complessiva dell'applicazione:

- Singolo file HTML per facilitare la portabilità (non richiede NodeJS o altra componente lato server)
- Frontend: interfaccia web reattiva basata su HTML5 e CSS3 (nessun utilizzo di framework)
- Business logic in Javascript: classi e funzioni dedicate per il calcolo EOQ, Safety Stock e analisi comparativa.

Fase 3: Implementazione (Settimana 4-6)

La fase di sviluppo ha proceduto secondo un approccio iterativo:

- Implementazione dei moduli di calcolo matematico (EOQ, ROP, Safety Stock)
- Sviluppo delle funzioni per la creazione di un dataset realistico con dati di domanda simulati
- Sviluppo dell'interfaccia web con validazione dei dati di input, esecuzione dei calcoli e della simulazione, rappresentazione dei dati tramite tabella e grafici

Fase 4: Testing e Validazione (Settimana 7)

La fase di testing ha garantito la correttezza e l'affidabilità dell'applicazione:

- Test unitari su singole funzioni (check expected vs actual)
- System test frontend per compatibilità browser Chrome
- Validazione dei risultati comparandoli con letteratura e benchmark esistenti
- Test di performance con dataset di varie dimensioni

Fase 5: Documentazione e Rapporto Finale (Settimana 8)

La fase finale ha completato il progetto attraverso la stesura dell'elaborato con una documentazione comprensiva di:

- Codice completamente commentato e documentato
- Analisi comparativa dei risultati ottenuti
- Discussione dei risultati e delle implicazioni pratiche

Risorse e strumenti impiegati

(Descrivere quali risorse - bibliografia, banche dati, ecc. - e strumenti - software, modelli teorici, ecc. - sono stati individuati ed utilizzati per la redazione dell'elaborato. Descrivere, inoltre, i motivi che hanno orientato la scelta delle risorse e degli strumenti, la modalità di individuazione e reperimento delle risorse e degli strumenti, le eventuali difficoltà affrontate nell'individuazione e nell'utilizzo di risorse e strumenti ed il modo in cui sono state superate):

La redazione di questo project work ha richiesto l'impiego di molteplici risorse di natura differente, suddivisibili in categorie quali bibliografia specializzata, dataset storici, strumenti software, modelli teorici e metodologie di ricerca.

La presente sezione descrive in dettaglio le risorse individuate, i criteri di selezione adottati, le modalità di reperimento e relativo utilizzo.

RISORSE BIBLIOGRAFICHE

Criteri di Selezione

La selezione bibliografica è stata effettuata seguendo criteri rigorosi di qualità e rilevanza:

- Rilevanza direttamente connessa ai temi di EOQ e Safety Stock
- Affidabilità della fonte (riviste peer-reviewed, case studies aziendali, testi accademici riconosciuti)
- Attualità: priorità a pubblicazioni recenti (ultimi 15 anni) con inclusione di classici storici (Harris 1913)
- Equilibrio teorico-pratico: combinazione di fondamenti matematici con applicazioni industriali concrete
- Diversità di prospettive: economia aziendale, operations research, supply chain management

Fonti Primarie Individuate

Le risorse bibliografiche sono state reperite attraverso diversi canali accademici e professionali.

Tra i database accademici sono stati utilizzati Google Scholar e ResearchGate.

Il reperimento di case studies è stato effettuato tra le pubblicazioni di società di consulting internazionali (McKinsey, BCG, Deloitte)

DATASET E BANCHE DATI

Per la validazione empirica dei modelli teorici è stato necessario disporre di dati storici di domanda. Il dataset utilizzato è stato generato sinteticamente ma basato su pattern realistici osservati in contesti aziendali.

Parametri utilizzati per la generazione del Dataset

Durata in giorni del mese	30
Numero mesi simulazione	Configurabile, default: 36
Domanda media mensile	Configurabile, default: 5.200
Trend mensile (variazione domanda mensile)	Configurabile, default: +20
Rumore casuale	Configurabile, default: 3.000
Mese picco (picco gestione domanda stagionale)	Configurabile, default: 12
Coefficiente amplificazione mese picco stagionale	Configurabile, default: 1,28

Modalità di Generazione

Il dataset sintetico è generato mediante la funzione “`generateMonthlyDemand`”.

Calibrazione Basata su Fonti Reali

I parametri del modello di generazione dati sono stati calibrati sulla base di:

- Dati pubblicamente disponibili da case studies di supply chain industriale (pubblicazioni accademiche)
- Coefficienti di variazione tipici per categorie di prodotti a domanda indipendente (ricerche empiriche)
- Fattori di stagionalità osservati in letteratura per segmenti specifici (retail, manifattura, consumer goods)

STRUMENTI SOFTWARE E TECNOLOGIE

Lo sviluppo dell'applicazione ha richiesto l'impiego di molteplici strumenti software, selezionati sulla base di criteri di maturità, diffusione industriale, e adeguatezza al problema specifico.

Stack Tecnologico Completo

Tecnologia	Versione	Utilizzo
VIM	-	Editor testuale
HTML	5	Interfaccia web frontend
CSS	3	Stili ed estetica frontend
Javascript	ES6+	Business logic e gestione user interaction frontend
ChartJS	-	Libreria grafici
Microsoft Excel	-	Verifica e analisi numerica risultati
Microsoft Word	-	Documento project work

Criteri di Selezione delle Tecnologie

- Open-source: Garantisce trasparenza, sostenibilità e assenza di costi di licenza
- Comunità attiva: Supporto consolidato, documentazione completa, forum online di riferimento
- Compatibilità: Verifica esplicita della compatibilità versioni
- Diffusione industriale: Competenze richieste dal mercato del lavoro contemporaneo

MODELLO TEORICI E METODOLOGIE

Oltre alle risorse computazionali, il progetto ha fatto uso intensivo di modelli teorici consolidati nella letteratura di operations research e supply chain management.

Il modello **EOQ** di Harris-Wilson fornisce la base teorica per l'ottimizzazione analitica dei costi attraverso la formula classica:

$$EOQ = \sqrt{\frac{2 \cdot D \cdot S}{H}}$$

Dove:

- **D** rappresenta la domanda annua
- **S** è il costo di setup per ciascun ordine mensile (es. costi amministrativi, trasporto)
- **H** è il costo di mantenimento delle scorte per unità per anno (es. magazzino, obsolescenza, capitale immobilizzato)

Il modello di **Safety Stock** (Supply chain management, Chopra & Meindl, 2016) propone un approccio statistico che utilizza la distribuzione normale per legare il livello di servizio alla quantità di scorta cuscinetto:

$$SS = Z \times \sigma \times \sqrt{L}$$

Dove:

- **Z** rappresenta il fattore di servizio (Z-score dalla distribuzione normale), che riflette il livello di servizio desiderato (es. Z=1.65 per 95% di probabilità di non stockout).
- **L** rappresenta il lead time (in giorni o unità di tempo) e che scala la variabilità con la durata del periodo di approvvigionamento
- **σ** è Deviazione standard della domanda giornaliera (o per unità di tempo), che misura l'incertezza della domanda.

Il modello di **Reorder Point** fornisce trigger per attivazione di nuovi ordini e viene calcolato tramite la formula:

$$ROP = (d \times L) + SS$$

Dove:

- **d** è la domanda media
- **L** è il lead time
- **SS** rappresenta la scorta di sicurezza

METODOLOGIE DI RICERCA APPLICATE

Il progetto ha integrato diverse metodologie di ricerca complementari:

- Modellizzazione matematica:
- Simulazione e Validazione Numerica: Implementazione computazionale dei modelli, test su dataset sintetici, benchmarking contro letteratura. Output: software funzionante, validazione della correttezza.
- Analisi Comparativa: Confronto sistematico tra modello EOQ classico e integrato con Safety Stock su metriche multiple. Output: quantificazione di trade-off, raccomandazioni decisionali.

CONCLUSIONI

La selezione accurata delle risorse, unita a una gestione proattiva e metodica, ha permesso di realizzare un elaborato che coniuga solidità accademica con applicabilità industriale concreta.

Il progetto integra elementi di teoria (operations research), pratica (implementazione software), e validazione empirica (dataset realistico), creando una soluzione completa e riproducibile.

BIBLIOGRAFIA

1. Harris, F. W. (1913). "How Many Parts to Make at Once". *The Magazine of Management*, 10(2), 135-136. [Articolo seminale che ha introdotto il modello EOQ]
2. Silver, E. A., Pyke, D. F., & Peterson, R. (2017). *Inventory Management and Production Planning and Scheduling* (3rd ed.). John Wiley & Sons. [Riferimento classico in inventory management]
3. Williams, T. M. (1984). "Stock Control with Uncertain Demand". *International Journal of Production Research*, 22(4), 555-567. [Gestione dell'inventario con incertezza]

PARTE SECONDA – PREDISPOSIZIONE DELL’ELABORATO

Obiettivi del progetto

(Descrivere gli obiettivi raggiunti dall’elaborato, indicando in che modo esso risponde a quanto richiesto dalla traccia):

L’obiettivo dell’elaborato è lo sviluppo di un’applicazione software che implementi:

1. il calcolo del Lotto Economico di Ordinazione (EOQ) utilizzando la formula classica EOQ
$$= \sqrt{(2 \times D \times S) / H}$$
2. il calcolo della Scorta di Sicurezza (Safety Stock), calcolata in funzione della variabilità della domanda, del lead time e del livello di servizio desiderato
3. il calcolo del Punto di Riordino (ROP) come $ROP = (d \times L) + SS$, dove d è la domanda media, L il lead time e SS la scorta di sicurezza

Si procede quindi a progettare e implementare un’interfaccia utente intuitiva, che consenta agli utenti di inserire facilmente i parametri necessari (domanda annua, costo di setup, costo di mantenimento, domanda media, lead time, variabilità della domanda) e visualizzare i risultati calcolati in modo chiaro e significativo.

Sarà quindi infine possibile realizzare un’analisi comparativa tra il modello EOQ classico e il modello EOQ con integrazione della Scorta di Sicurezza, evidenziando i vantaggi tangibili nella gestione delle scorte in scenari di domanda variabile e i relativi costi associati.

Contestualizzazione

(Descrivere il contesto teorico e quello applicativo dell’elaborato realizzato):

La gestione dell’inventario rappresenta una delle sfide critiche della moderna gestione aziendale, in particolare per quelle imprese che si confrontano quotidianamente con la necessità di equilibrare due forze contrastanti: l’esigenza di mantenere scorte sufficienti per soddisfare la domanda dei clienti e il rischio di immobilizzare eccessivamente capitale in giacenze di magazzino. Il presente progetto si propone di affrontare questa problematica complessa attraverso l’implementazione di un software che integri due modelli fondamentali della operations research: il Lotto Economico di Ordinazione (Economic Order Quantity, EOQ) e la gestione della Scorta di Sicurezza con il relativo Punto di Riordino (Reorder Point).

Il modello EOQ, sviluppato per la prima volta da Harris Ford W. nel 1913 e successivamente approfondito da R. H. Wilson, rimane un pilastro della teoria dell’inventory management. Questo modello fornisce una soluzione analitica elegante al problema di determinare la quantità ottimale da ordinare, minimizzando il costo totale derivante dalla somma del costo di ordinazione e del costo di mantenimento delle scorte. Tuttavia, la formulazione classica di Harris-Wilson assume una domanda deterministica e costante nel tempo, un’ipotesi che raramente si verifica nella pratica aziendale contemporanea.

La realtà operativa è caratterizzata da una domanda soggetta a variabilità, dovuta a fattori quali la stagionalità, le fluttuazioni del mercato, e l’incertezza nei tempi di consegna (lead time). Per affrontare questa variabilità, è necessario integrare al modello EOQ il concetto di Scorta di Sicurezza (Safety Stock), che rappresenta un cuscinetto di inventario aggiuntivo mantenuto per

proteggere l'azienda dal rischio di stockout (esaurimento di magazzino) quando la domanda supera le aspettative o i tempi di consegna si allungano.

La Scorta di Sicurezza è calcolata sulla base di considerazioni statistiche e probabilistiche, incorporando il concetto di Livello di Servizio (Service Level), che rappresenta la probabilità desiderata di non incorrere in una situazione di stockout. Il Punto di Riordino (ROP) è quindi calcolato come la somma della domanda attesa durante il lead time più la Scorta di Sicurezza, fornendo un'indicazione precisa del livello di inventario al quale è opportuno attivare un nuovo ordine.

Descrizione dei principali aspetti progettuali
(Sviluppare l'elaborato richiesto dalla traccia prescelta):

Tutto il codice sorgente del progetto è disponibile tramite repository github pubblico all'indirizzo:

https://github.com/alexfilippini85/project_work_0312301747

L'applicazione è stata pubblicata tramite github pages ed è possibile provarla all'indirizzo:

https://alexfilippini85.github.io/project_work_0312301747/

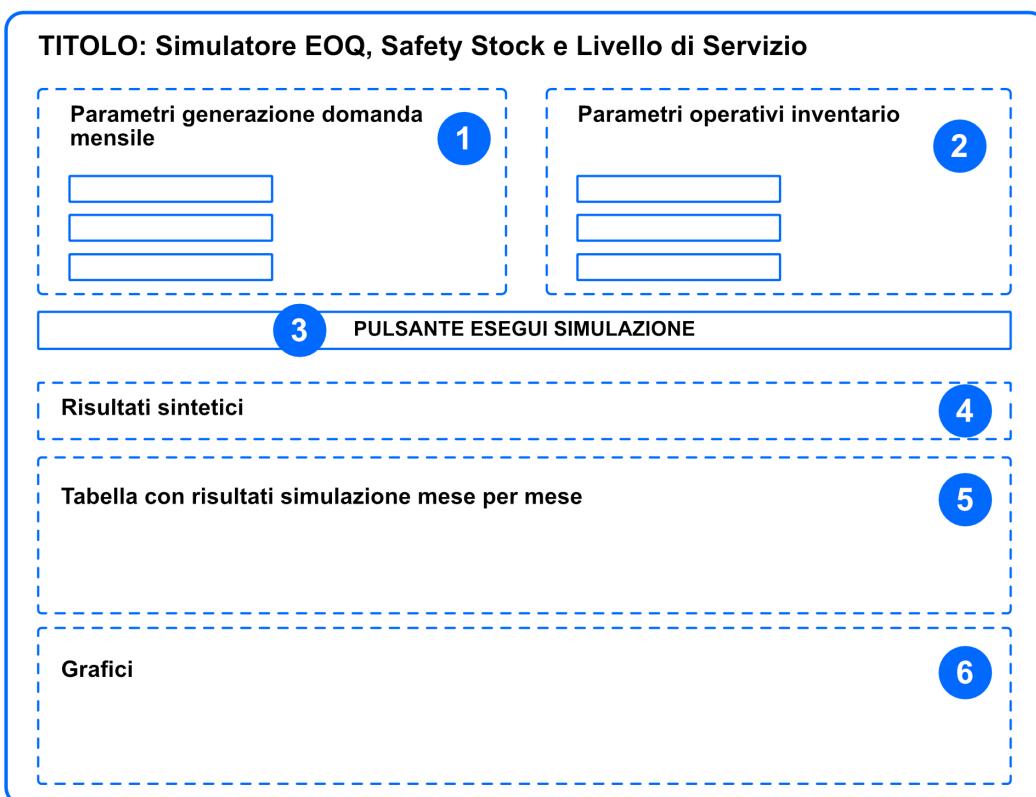
Il progetto consta di un'applicazione web, sviluppata secondo il modello SPA (Single Page Application).

I file che compongono il progetto sono elencati di seguito:

- **index.html**: è il file principale dell'applicazione
- **style.css**: contiene gli stili estetici, il posizionamento e le dimensioni dei componenti della user interface
- **random.js**: contiene l'implementazione di un generatore di numeri pseudo-casuali (PRNG) che consente la ripetibilità della simulazione attraverso l'uso di un seed
- **demand_generator.js**: contiene il codice javascript per la modellazione e la generazione della domanda
- **simulation_engine.js**: contiene le funzioni di calcolo di EOQ, SS e ROP, oltre ad un simulatore di approvvigionamento e consumo che utilizzi la domanda generata e i parametri calcolati
- **chart.umd.min.js**: è la libreria opensource chart.js per il rendering di grafici all'interno di pagine web
- **simulation_chart.js**: contiene le funzioni per la modellazione dei dati e la configurazione dei grafici
- **simulation_table.js**: contiene le funzioni per la generazione della tabella della simulazione all'interno della pagina web

STRUTTURA DELL'APPLICAZIONE

L'interfaccia utente è strutturata in sei parti principali mostrate nella figura sottostante e di seguito descritte.



1. Parametri generazione domanda mensile:

Nella prima sezione sono contenuti tutti i parametri di input utilizzati per la generazione sintetica dei dati della domanda. Il primo parametro è il SEED, utilizzato per consentire la ripetibilità della simulazione. A parità di SEED, il sistema genererà la medesima sequenza di numeri casuali.

A seguire troviamo:

- la domanda media mensile
- il trend da applicare ad ogni mese
- un rumore casuale, utilizzato per rendere più realistica la serie sintetica attraverso l'introduzione di maggiore variabilità tra i periodi
- il mese di picco e il relativo fattore di crescita, utili per rappresentare picchi della domanda in particolari periodi dell'anno
- il numero di mesi della simulazione

Tutti i campi vengono preimpostati con valori di default, utilizzati nell'analisi di questo elaborato.

Di seguito il relativo codice HTML (index.html, righe 19-43)

```
<!-- Sezione 1: generazione serie domanda -->
<fieldset>
    <legend>Parametri generazione domanda mensile</legend>

    <label for="seed">Seed (per ripetibilità simulazione)</label>
    <input type="number" id="seed" value="50" min="0" step="1" />

    <label for="baseLevel">baseLevel (domanda media mensile di base)</label>
    <input type="number" id="baseLevel" value="5200" min="0" step="1" />

    <label for="trendPerPeriod">trendPerPeriod (variazione per mese)</label>
    <input type="number" id="trendPerPeriod" value="20" step="1" />

    <label for="noiseStd">noiseStd (rumore casuale)</label>
    <input type="number" id="noiseStd" value="3000" step="1" />

    <label for="peakMonth">peakMonth (1-12)</label>
    <input type="number" id="peakMonth" value="12" min="1" max="12" step="1" />

    <label for="peakFactor">peakFactor (moltiplicatore per mese di picco)</label>
    <input type="number" id="peakFactor" value="1.28" step="0.1" />

    <label for="months">Numero mesi di simulazione</label>
    <input type="number" id="months" value="36" min="1" max="60" step="1" />
</fieldset>
```

2. Parametri operativi inventario

Nella seconda sezione sono contenuti tutti i parametri di input utilizzati per il calcolo dell'EOQ, del Safety Stock e del punto di riordino.

I parametri disponibili sono:

- **Checkbox safety stock:** grazie a questo parametro è possibile confrontare i risultati delle simulazioni con e senza l'applicazione del safety stock nel calcolo del punto di riordino
- **Costo di setup:** identifica il costo di ogni singolo ordine di approvvigionamento
- **Costo di mantenimento:** identifica il costo annuale di mantenimento di ogni singola unità (es. costo di magazzino, costo immobilizzazione finanziaria, gestione e amministrazione)
- **Lead time:** identifica il tempo di approvvigionamento a seguito di ordine espresso in giorni
- **Livello di servizio (Z):** identifica il livello di servizio target. I valori sono espressi in percentuale in modo da risultare più comprensibili all'utente

Di seguito il relativo codice HTML (index.html, righe 45-67)

```
<!-- Sezione 2: parametri inventario -->
<fieldset>
    <legend>Parametri EOQ / Inventario</legend>

    <label for="applySafetyStock">Safety Stock</label>
    <input type="checkbox" id="applySafetyStock" value="1" checked />

    <label for="setupCost">Costo di setup / ordine (S)</label>
    <input type="number" id="setupCost" value="500" min="0" step="0.01" />

    <label for="holdingCost">Costo di mantenimento per unità/anno (H)</label>
    <input type="number" id="holdingCost" value="5" min="0" step="0.01" />

    <label for="leadTime">Lead time (giorni)</label>
    <input type="number" id="leadTime" value="15" min="0" step="1" />

    <label for="serviceZ">Z per livello di servizio (es. 1.65 ≈ 95%)</label>
    <select id="serviceZ" required>
        <option value="1.2815517783164978">90%</option>
        <option value="1.64485365152359" selected>95%</option>
        <option value="2.326347053050995">99%</option>
    </select>
</fieldset>
```

3. Pulsante per l'esecuzione della simulazione

Per eseguire la simulazione, dopo aver compilato i parametri di input richiesti, è possibile premere sul pulsante “Esegui simulazione”.

Di seguito il relativo codice HTML (index.html, righe 70-71)

```
<!-- Sezione 3: pulsante per avviare simulazione -->
<button id="runBtn" onclick="startSimulation()">Esegui simulazione</button>
```

Al click del pulsante è stata collegata la funzione *startSimulation* che verrà approfondita di seguito.

startSimulation - FASE 1

Nella prima fase della funzione vengono reperiti i parametri utente necessari alla generazione dei dati sintetici della domanda.

La chiamata al costruttore dell'istanza *syntheticDemand* (appartenente alla classe Demand) determina la generazione dei dati simulati.

Di seguito il relativo codice Javascript (index.html, righe 120-139)

```
// Lettura parametri serie domanda
const seed = parseFloat(document.getElementById("seed").value) || 0;
const baseLevel = parseFloat(document.getElementById("baseLevel").value) || 0;
const trendPerPeriod = parseFloat(document.getElementById("trendPerPeriod").value) || 0;
const noiseStd = parseFloat(document.getElementById("noiseStd").value) || 0;
const peakMonth = parseInt(document.getElementById("peakMonth").value) || 1;
const peakFactor = parseFloat(document.getElementById("peakFactor").value) || 1;
const months = parseInt(document.getElementById("months").value) || 12;

// Generazione dati sintetici domanda
const syntheticDemand = new Demand(
  seed,
  months,
  baseLevel,
  trendPerPeriod,
  noiseStd,
  peakMonth,
  peakFactor
);
```

startSimulation - FASE 2

Nella seconda fase della funzione vengono reperiti i parametri operativi da utilizzarsi nella simulazione.

A seguire viene calcolato EOQ, SS e ROP attraverso la chiamata a funzioni dedicate della classe *SimulationEngine*.

Il safety stock viene azzerato qualora la checkbox *applySafetyStock* non sia selezionata.

Di seguito il relativo codice Javascript (index.html, righe 142-159)

```
// Lettura parametri servizio
const applySafetyStock = document.getElementById("applySafetyStock").checked;
const setupCost = parseFloat(document.getElementById("setupCost").value) || 0;
const holdingCost = parseFloat(document.getElementById("holdingCost").value) || 0;
const leadTime = parseFloat(document.getElementById("leadTime").value) || 0;
const serviceZ = parseFloat(document.getElementById("serviceZ").value) || 0;

// Calcolo EOQ, SS, ROP
const EOQ = simulationEngine.computeEOQ(syntheticDemand.annualDemand, setupCost, holdingCost);
const safetyStock = applySafetyStock
  ? simulationEngine.computeSafetyStock(syntheticDemand.dailyStdDeviation, leadTime, serviceZ)
  : 0;
const reorderPoint = simulationEngine.computeReorderPoint(
  syntheticDemand.dailyAvgDemand,
  leadTime,
  safetyStock
);
```

startSimulation - FASE 3

Nella terza fase della funzione si procede ad eseguire la simulazione attraverso la chiamata al metodo *simulateInventory* della classe *SimulationEngine*.

La funzione ritorna un oggetto con due proprietà:

- *simulationMonths*: è un array in cui ogni occorrenza rappresenta i dati simulati di un mese
- *overallServiceLevel*: è un valore numerico che rappresenta il livello di servizio totale raggiunto

Di seguito il relativo codice Javascript (index.html, righe 162-169)

```
// Avvio simulazione
const { simulationMonths, overallServiceLevel } = simulationEngine.simulateInventory(
    syntheticDemand,
    EOQ,
    reorderPoint,
    safetyStock,
    leadTime
);
```

startSimulation - FASE 4

Nella quarta e ultima fase della funzione, si procede ad aggiornare l’interfaccia utente con i risultati della simulazione:

- Aggiornamento dei dati sintetici (domanda annua, domanda media giornaliera, deviazione standard della domanda giornaliera, EOQ, Safety stock, Reorder point e livello di servizio complessivo).
- Aggiornamento della tabella con i dati “mese per mese”
- Aggiornamento dei grafici

Di seguito il relativo codice Javascript (index.html, righe 173-182)

```
// Render risultati sintetici
const summary = document.getElementById("summary");
summary.innerHTML = `
    Domanda annua (D, unità/anno): <strong>${syntheticDemand.annualDemand.toFixed(0)}</strong>, <br>
    Domanda media giornaliera: <strong>${syntheticDemand.dailyAvgDemand.toFixed(2)}</strong>, <br>
    Variabilità domanda giornaliera (std. deviation): <strong>${syntheticDemand.dailyStdDeviation.toFixed(2)}</strong>, <br>
    EOQ: <strong>${EOQ}</strong> unità,<br>
    Safety stock: <strong>${safetyStock}</strong> unità,<br>
    Reorder point: <strong>${reorderPoint}</strong> unità.<br>
    Livello di servizio complessivo: <strong>${(overallServiceLevel * 100).toFixed(2)}%</strong>.`;
```

Di seguito il relativo codice Javascript (index.html, righe 185-186)

```
// Render tabella simulazione con dati mese per mese
simulationTable.render(document.querySelector("#resultsTable tbody"), simulationMonths);
```

Di seguito il relativo codice Javascript (index.html, righe 189-201)

```
// Render grafico simulazione con dati mese per mese
simulationChart.updateSimulation(
    "simulationChart",
    syntheticDemand,
    simulationMonths
);

// Render grafico service level con dati mese per mese
simulationChart.updateServiceLevel(
    "serviceLevelChart",
    simulationMonths
);
```

4. Risultati sintetici

In questa sezione, all'interno del paragrafo *summary*, vengono mostrati i risultati sintetici tra cui:

- **Domanda annua:** espressa in unità, rappresenta **D** nel calcolo dell'EOQ
- **Domanda media giornaliera:** espressa in unità, rappresenta **d** nel calcolo dell'ROP
- **Variabilità domanda giornaliera:** espressa in unità, rappresenta **σ** nel calcolo del Safety Stock e misura l'incertezza della domanda
- **EOQ:** espresso in unità, rappresenta il lotto minimo di ordinazione
- **Safety stock:** espresso in unità, rappresenta la quota di sicurezza di riordino
- **Reorder point:** espresso in unità, rappresenta la soglia di riordino
- **Livello di servizio complessivo:** espresso in percentuale, rappresenta l'efficacia nel soddisfare la domanda, ottenuta tramite applicazione di EOQ, Safety stock e reorder point

Di seguito il relativo codice HTML (index.html, righe 75-77)

```
<!-- Sezione 4: Risultati sintetici -->
<h2>Risultati sintetici</h2>
<p id="summary"></p>
```

5. Tabella con dettaglio mese per mese esito simulazione

In questa sezione viene mostrata una tabella con il dettaglio della simulazione. Ogni record rappresenta un mese della simulazione.

Le informazioni disponibili sono:

- **Mese**
- **Stock iniziale:** rappresenta lo stock all'inizio di ogni mese
- **Domanda:** rappresenta la domanda complessiva del mese
- **Incoming:** rappresenta gli ingressi previsti nel mese, frutto di ordini eseguiti precedentemente
- **Ordine (se lanciato):** rappresenta il numero di ordini eseguiti nel mese che genereranno ingressi in futuro, in funzione del lead time
- **Domanda soddisfatta:** rappresenta il numero totale di unità per cui si è soddisfatta la domanda
- **Giorni stockout:** rappresenta il numero di giorni nel mese per cui non si è riusciti a soddisfare totalmente la domanda
- **Backorder:** rappresenta il numero di unità per cui non si è soddisfatta la domanda e che verrà sommato alla domanda del mese successivo
- **Stock finale:** rappresenta il numero di unità rimanenti a fine mese
- **Livello di servizio:** rappresenta il livello di servizio effettivamente ottenuto nel mese

Di seguito il relativo codice HTML (index.html, righe 79-99)

```
<!-- Sezione 5: Tabella con dettaglio mese per mese esito simulazione -->
<h2>Dettaglio mese per mese</h2>
<div style="overflow-x: auto">
  <table id="resultsTable">
    <thead>
      <tr>
        <th>Mese</th>
        <th>Stock iniziale</th>
        <th>Domanda</th>
        <th>Incoming</th>
        <th>Ordine (se lanciato)</th>
        <th>Domanda soddisfatta</th>
        <th>Giorni stockout</th>
        <th>Backorder</th>
        <th>Stock finale</th>
        <th>Liv. servizio</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>
</div>
```

6. Grafici

In questa sezione vengono mostrati tre grafici:

- **Andamento mensile:** il grafico mostra:
 - l'andamento dello stock con riferimento l'inizio di ogni mese
 - l'andamento della quota di ingresso di nuovo materiale per ogni mese
 - l'andamento della domanda totale di ogni mese
 - i mesi in cui non si è riusciti a soddisfare a pieno la domanda (stockout)
- **Livello di servizio:** il grafico mostra per ogni mese:
 - Il numero di giorni in cui si è riusciti a soddisfare la domanda
 - Il numero di giorni in cui non si è riusciti a soddisfare la domanda (stockout)

Di seguito il relativo codice HTML (index.html, righe 102-110)

```
<div class="chartContainer">
  <h2>Andamento mensile</h2>
  <canvas id="simulationChart" height="120"></canvas>
</div>

<div class="chartContainer">
  <h2>Livello di servizio</h2>
  <canvas id="serviceLevelChart" height="80"></canvas>
</div>
```

GENERATORE DATI SINTETICI DELLA DOMANDA

Al fine di poter simulare diversi scenari di domanda, si è proceduto con lo sviluppo di un generatore di dati sintetici.

Il generatore utilizza diversi parametri definite dall'utente tra cui: numero di mesi della simulazione, domanda media mensile, trend mensile, mese di picco e relativo fattore di crescita (utili per rappresentare picchi della domanda in particolari periodi dell'anno).

Per rendere la serie generata statisticamente affidabile, è stato introdotto anche del rumore casuale, utile a generare maggiore variabilità tra i periodi.

Si è inoltre ritenuto fondamentale che la simulazione potesse essere ripetibile, ovvero che a parità di parametri di input, potesse generare gli stessi valori di domanda.

La classe *Demand* utilizza un generatore di numeri pseudo-casuale configurato con seed deciso dall'utente.

Di seguito il relativo codice Javascript (demand_generator.js, righe 7-13)

```
class Demand {
  random;
  months = [];
  annualDemand;
  dailyAvgDemand;
  dailyStdDeviation;
```

La funzione *generateMonthlyDemand*

- Genera una domanda mensile (*monthDemand*) partendo da un livello base e applicando trend mensile, effetto stagionale di picco e rumore gaussiano approssimato (metodo Box-Muller)
- Genera una domanda giornaliera tramite la chiamata alla funzione *generateDailyDemand*
- Restituisce un array *demand* dove ogni occorrenza rappresenta un mese tramite un oggetto composto dalle proprietà *tot* (domanda mensile) e *daily* (array della domanda giornaliera)

Di seguito il relativo codice Javascript (demand_generator.js, righe 26-53)

```
// Generazione domanda mensile con trend + stagionalità + rumore
#generateMonthlyDemand(months, baseLevel, trendPerPeriod, noiseStd, peakMonth, peakFactor) {
    const demand = [];

    for (let t = 0; t < months; t++) {
        const monthIndex = t + 1; // 1..12, 13.. ecc.
        const monthInYear = ((monthIndex - 1) % 12) + 1;

        const trend = trendPerPeriod * t;
        const peakMult = monthInYear === peakMonth ? peakFactor : 1;

        // Rumore gaussiano approssimato (Box-Muller)
        const u1 = this.random.generate();
        const u2 = this.random.generate();
        const randStdNorm = Math.sqrt(-2 * Math.log(u1)) * Math.cos(2 * Math.PI * u2);
        const noise = randStdNorm * noiseStd;

        const monthDemand = Math.max(0, Math.round((baseLevel + trend) * peakMult + noise));
        const domandaGiornaliera = this.#generateDailyDemand(monthDemand, 30);
        demand.push({
            tot: monthDemand,
            daily: domandaGiornaliera
        });
    }
    return demand;
}
```

La funzione *generateDailyDemand*

- Genera una domanda giornaliera per un mese, con distribuzione esponenziale usando seeded random. Simula vendite realistiche: molti giorni bassi, pochi picchi alti, somma ESATTA=totalMensile

Di seguito il relativo codice Javascript (demand_generator.js, righe 56-94)

```
/*
 * Genera domanda giornaliera mensile con distribuzione esponenziale usando seeded random.
 * Simula vendite realistiche: molti giorni bassi, pochi picchi alti, somma ESATTA=totalMensile.
 */
#generateDailyDemand(totalMensile, giorniMese) {

    /**
     * Tasso esponenziale adattivo: λ = giorni/total
     * Media giornaliera = 1/λ = totalMensile/giorniMese
     */
    const lambda = giorniMese / totalMensile;

    /**
     * FASE 1: Serie esponenziale raw con Inverse Transform Sampling
     * X_i = -ln(1-U_i) / λ   dove U_i ~ Uniforme[0,1] da PRNG
     */
    const serie = [];
    for (let i = 0; i < giorniMese; i++) {
        const u = this.random.generate(); // Usa SeededRandom
        serie.push(-Math.log(1 - u) / lambda);
    }

    /**
     * FASE 2: Normalizzazione proporzionale
     * x_norm[i] = round( (serie[i] / somma_serie) * totalMensile )
     */
    const somma = serie.reduce((a, b) => a + b, 0);
    const normalizzata = serie.map(x => Math.round((x / somma) * totalMensile));
    let sommaAttuale = normalizzata.reduce((a, b) => a + b, 0);

    /**
     * FASE 3: Correzione finale per somma ESATTA
     * Regola ULTIMO giorno per compensare errori arrotondamento
     * Garantisce: sum(normalizzata) === totalMensile
     */
    normalizzata[giorniMese - 1] += totalMensile - sommaAttuale;

    return normalizzata;
}
```

CALCOLO EOQ, SS, ROP

I calcoli di EOQ, Safety Stock e ROP sono stati implementati in tre funzioni indipendenti e pure (prive di effetti collaterali), in modo da garantirne il funzionamento tramite test di unità.

Di seguito il relativo codice Javascript (simulation_engine.js, righe 9-24)

```
// EOQ = sqrt(2DS/H)
computeEOQ(annualDemand, setupCost, holdingCost) {
    return Math.round(Math.sqrt((2 * annualDemand * setupCost) / holdingCost), 0);
}

// Safety stock = Z * sigma_d * sqrt(L)
computeSafetyStock(demandStd, leadTime, serviceZ) {
    return Math.round(serviceZ * demandStd * Math.sqrt(leadTime), 0);
}

// ROP = avgDailyDemand * leadTime + safetyStock
computeReorderPoint(avgDemand, leadTime, safetyStock) {
    return Math.round(avgDemand * leadTime + safetyStock, 0);
}
```

SIMULATORE SCENARIO

La funzione *simulateInventory* simula il comportamento di un sistema di gestione inventario (sistema ROP/EOQ) utilizzando domanda giornaliera sintetica, tracciando metriche di servizio e stockout su scala giornaliero e mensile.

Lo scopo principale è valutare l'efficacia dei parametri EOQ (Economic Order Quantity), reorder point, safety stock e lead time rispetto a una domanda stocastica realistica, assumendo mesi di 30 giorni uniformi.

Gli argomenti in input della funzione sono:

- **syntheticDemand**: oggetto con domanda mensile totale (tot) e array giornaliero (daily) per ogni mese
- **EOQ**: quantità economica d'ordine fissa
- **reorderPoint**: soglia ($ROP = \text{demand} \text{ durante lead time} + \text{safety stock}$)
- **safetyStock**: buffer di sicurezza
- **leadTimeDays**: tempo di approvvigionamento, espresso in giorni

La logica di funzionamento prevede di iniziare con $\text{stock} = ROP + EOQ/2$ (inventario medio target, ottimizzato per ridurre l'impiego di capitale).

Per ogni mese:

- si simulano 30 giorni consecutivi accumulando gli arrivi di ordini pendenti,
- si triggerano nuovi ordini quando $\text{stock} + \text{ordini in corso} < ROP$
 - questo permette ordini multipli outstanding durante Lead Time, evitando stockout con domanda variabile
- si soddisfa la domanda giornaliera accumulando l'eventuale backorder dei giorni precedenti
- si verifica presenza di eventuale domanda non soddisfatta valorizzando il backorder
- si contano i giorni di stockout

Di seguito il relativo codice Javascript (simulation_engine.js, righe 56-79)

```
simulateInventory(syntheticDemand, EOQ, reorderPoint, safetyStock, leadTimeDays) {  
    // tutta la simulazione assume mesi equivalenti di 30 giorni  
    const DAYS_PER_MONTH = 30;  
  
    // Valore stock iniziale: ROP + EOQ/2 --> Inventario medio iniziale, riduce immobilizzo capitale.  
    let stock = reorderPoint + EOQ / 2;  
  
    // oggetto (mappa) dove la chiave indica la data di arrivo.  
    // Il formato della chiave è il progressivo mese + il progressivo giorno  
    // (es: 5_23 significa mese 5, giorno 23)  
    let onOrder = {};  
  
    // array contenente i risultati della simulazione. Ogni occorrenza rappresenta un mese  
    let simulationMonths = [];  
  
    // domanda non servita accumulata  
    let backorder = 0;  
  
    // numero giorni totali della simulazione (numero totale mesi * numero giorno per mese)  
    const totalDays = syntheticDemand.months.length * DAYS_PER_MONTH;  
  
    // numero di giorni totali in stockout  
    let t_stockout_days = 0;
```

Di seguito il relativo codice Javascript (simulation_engine.js, righe 82-175)

```
// simulazione mensile  
// il consumo e i nuovi ordini vengono simulati giornalmente assumendo 30gg per ogni mese  
syntheticDemand.months.forEach((month, idx) => {  
    const m_label = idx + 1;  
    const m_demand = month.tot;  
    const d_demand = month.daily;  
    const m_initial_stock = stock;  
  
    // domanda servita nel mese (in unità)  
    let m_served = 0;  
  
    // numero di giorni del mese in stockout  
    let m_stockout_days = 0;  
  
    // quantità ordinata nel mese  
    let m_orderPlacedQty = 0;  
  
    // numero ordini nel mese  
    let m_ordersPlaced = 0;  
  
    // quantità in ingresso nel mese  
    let m_incomingQty = 0;  
  
    // simulazione giornaliera  
    for (let day = 1; day <= DAYS_PER_MONTH; day++) {  
        // fine simulazione giornaliera  
  
        const waitingArrivals = Object.values(onOrder).reduce((tot, v) => tot + v, 0);  
        const finalBackorder = Math.max(0, m_demand - m_served);  
  
        const serviceLevel = (DAYS_PER_MONTH - m_stockout_days) / DAYS_PER_MONTH;  
  
        simulationMonths.push({  
            month: m_label,  
            demand: m_demand,  
            stockStart: Math.round(m_initial_stock),  
            incomingQty: m_incomingQty,  
            orders: m_ordersPlaced,  
            orderQty: m_orderPlacedQty,  
            served: Math.round(m_served),  
            servedDays: DAYS_PER_MONTH - m_stockout_days,  
            stockoutDays: m_stockout_days,  
            backorder: Math.round(backorder),  
            stockEnd: Math.round(stock),  
            waitingArrivals: Math.round(waitingArrivals),  
            serviceLevel  
        });  
    };
```

Al termine, la funzione restituisce oggetto con:

- **simulationMonths**: contiene un array mensile con stock iniziale/finale, quantità in ingresso, ordini piazzati (quantità e conteggio), domanda servita, giorni stockout/servizio, backorder finale, arrivi pendenti
- **overallServiceLevel**: valorizzato con la percentuale dei giorni senza stockout sull'intera simulazione

Campi di applicazione

(Descrivere gli ambiti di applicazione dell'elaborato progettuale e i vantaggi derivanti della sua applicazione):

L'elaborato integra i modelli EOQ (Economic Order Quantity), Safety Stock e ROP (Reorder Point) in un framework per la gestione ottimale delle scorte, tipicamente sviluppato in progetti industriali per ottimizzare inventari in supply chain. Questi elementi combinati definiscono quantità, momento e buffer di sicurezza per gli ordini, riducendo costi e rischi.

AMBITI DI APPLICAZIONE

Questo modello trova applicazione primaria nella gestione delle scorte per aziende retail e manifatturiere.

Nel retail, come per elettronica o abbigliamento ad alto turnover, EOQ calcola l'ordine economico per bilanciare costi di setup e detenzione, mentre ROP attiva riordini preventivi e Safety Stock protegge da picchi stagionali.

In manifattura, gestisce materie prime e componenti: EOQ minimizza sprechi, ROP sincronizza le azioni di riordino con il lead time, Safety Stock mitiga ritardi dei fornitori.

Nel settore logistico e supply chain, l'integrazione supporta e-commerce e catene globali, ottimizzando magazzini multipli contro disruption (es. Brexit o meteo).

Aziende come Walmart, Costco e grossi gruppi di distribuzione, usano varianti per turnover elevati, integrando il modello con sistemi ERP per automazione.

VANTAGGI DELL'APPLICAZIONE

L'uso combinato riduce costi totali del 20-30% bilanciando ordering (setup) e holding (stoccaggio e obsolescenza), liberando capitale (working capital).

Inoltre, previene eventi di stockout, con conseguente impatto sulle vendite e sulla soddisfazione cliente, ed eventi di overstock, che generano sprechi e danni per materie deperibili.

Attraverso l'uso del Safety Stock, si assicura la continuità aziendale mantenendo un elevato livello di efficienza operativa.

A livello strategico, aumenta competitività: customer satisfaction (disponibilità), produzione stabile, supply chain resiliente.

Valutazione dei risultati

(Descrivere le potenzialità e i limiti ai quali i risultati dell'elaborato sono potenzialmente esposti)

SIMULAZIONE COMPARATIVA CON e SENZA SAFETY STOCK

Nel contesto della gestione ottimale delle scorte, l'integrazione dei modelli EOQ (Economic Order Quantity), Safety Stock e ROP (Reorder Point) rappresenta un pilastro fondamentale per bilanciare costi e servizio al cliente.

Questo capitolo presenta i risultati di una simulazione comparativa condotta tramite l'applicazione software sviluppata nell'elaborato, confrontando due scenari operativi distinti: uno senza l'impiego di Safety Stock e uno con la sua inclusione sistematica.

Senza Safety Stock, il sistema si basa esclusivamente su EOQ per la quantità d'ordine e ROP per il punto di riordino, esponendosi a rischi elevati di stockout dovuti a fluttuazioni della domanda o ritardi nei lead time.

Al contrario, l'integrazione del Safety Stock introduce un buffer protettivo, calcolato in base alla variabilità storica (ad esempio, tramite deviazione standard della domanda e servizio target al 95%), mitigando tali incertezze.

Questa analisi quantitativa, supportata da dati simulati su un orizzonte temporale di 36 mesi, evidenzia metriche chiave come tasso di servizio, rotazione delle scorte e frequenza di shortage. I risultati dimostrano tangibilmente i vantaggi del modello integrato, fornendo evidenze empiriche per decisioni data-driven in contesti industriali reali.

PARAMETRI DELLA SIMULAZIONE

Entrambe le simulazioni (con e senza Safety Stock) sono state effettuate con i medesimi parametri, di seguito riportati:

PARAMETRO	VALORE
Seed	50
Domanda mensile media di base	5.200
Trend per periodo	20
Rumore	3.000
Mese di picco	12
Fattore di picco	1,28
Numero di mesi	36
Costo di setup di ogni ordine	500
Costo di mantenimento per unità/anno	5
Lead Time	15
Livello di servizio	95%

SIMULAZIONE SCENARIO SENZA SAFETY STOCK

I risultati della simulazione mostrano chiaramente l'inefficacia del modello senza l'utilizzo del Safety Stock.

Il livello di servizio totale ottenuto è 66,11%, contro il 95% di obiettivo, con un totale di 366 giorni in stockout.

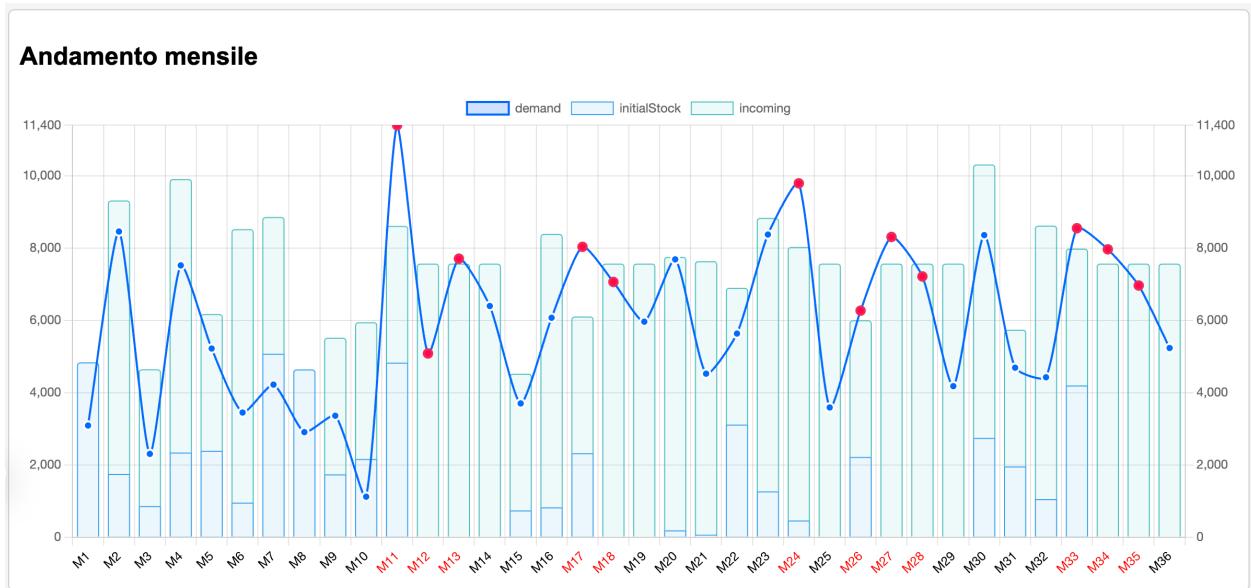
L'EOQ e il calcolo del Reorder Point standard risultano inadeguati nel gestire scenari con elevata variabilità della domanda, dove picchi improvvisi possono non essere coperti dalle scorte disponibili.

KPI	VALORE
Domanda annua	71.568
Domanda media giornaliera	196
Variabilità domanda giornaliera (std.deviation)	231
EOQ	3.783
Reorder Point	2.941
Livello di servizio complessivo	66,11%
Numero di giorni in stockout	366
Numero di ordini	57

Il grafico sottostante mostra:

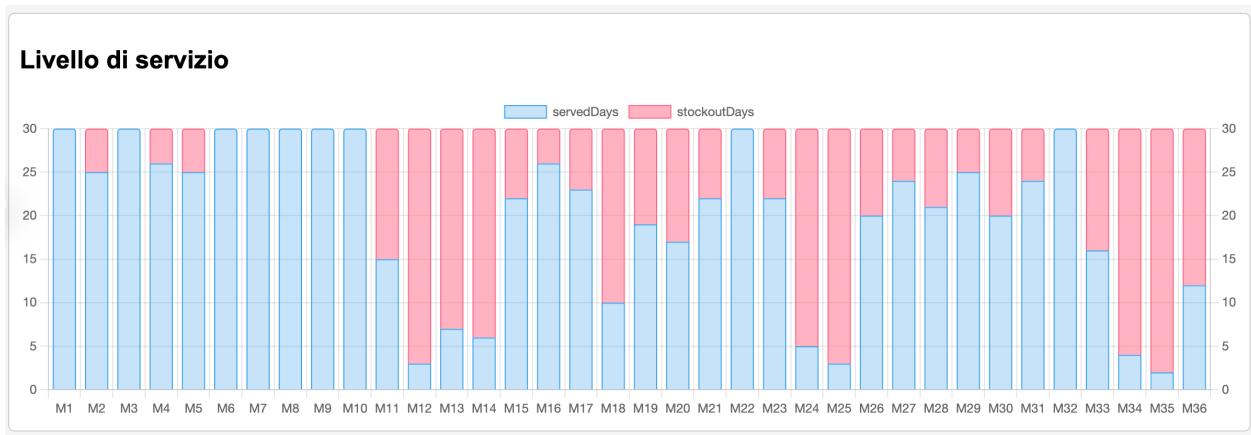
- **initialStock**: lo stock ad inizio di ogni periodo
- **incoming**: la quantità di nuovi ingressi nel mese, frutto di ordini precedenti
- **demand**: la domanda mensile

I marker in rosso sulla linea della domanda, corrispondenti ai mesi in rossi nella legenda sull'asse x, mostrano i mesi di completo stockout, ovvero dove il totale della domanda del mese risulta maggiore della disponibilità di stock cumulato.



Il grafico sottostante mostra:

- **servedDays**: i giorni per cui è stato possibile soddisfare a pieno la domanda
- **stockoutDays**: i giorni per i quali non è stato possibile soddisfare a pieno la domanda, generando ritardi nella consegna, e la generazione di backorder per i giorni successivi



SIMULAZIONE SCENARIO CON SAFETY STOCK

I risultati della simulazione mostrano chiaramente l'efficace contributo dell'introduzione del Safety Stock, che agisce come buffer utile a fronteggiare picchi improvvisi della domanda. Il livello di servizio totale ottenuto è 95,56%, soddisfando a pieno il 95% impostato come obiettivo, con un totale di soli 48 giorni in stockout.

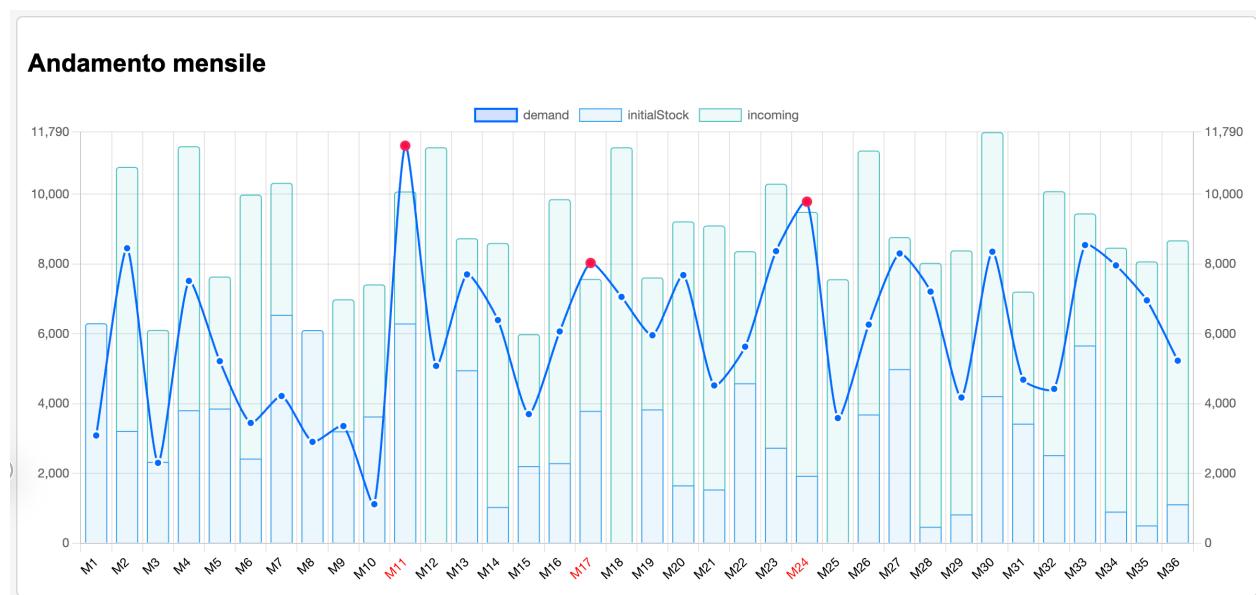
KPI	VALORE
Domanda annua	71.568
Domanda media giornaliera	196
Variabilità domanda giornaliera (std.deviation)	231
EOQ	3.783
Safety stock	1.472
Reorder Point	4.413
Livello di servizio complessivo	95,56%
Numero di giorni in stockout	48
Numero di ordini	57

Il grafico sottostante mostra:

- **initialStock**: lo stock ad inizio di ogni periodo
- **incoming**: la quantità di nuovi ingressi nel mese, frutto di ordini precedenti
- **demand**: la domanda mensile

I marker in rosso sulla linea della domanda, corrispondenti ai mesi in rossi nella legenda sull'asse x, mostrano i mesi di completo stockout, ovvero dove il totale della domanda del mese risulta maggiore della disponibilità di stock cumulato.

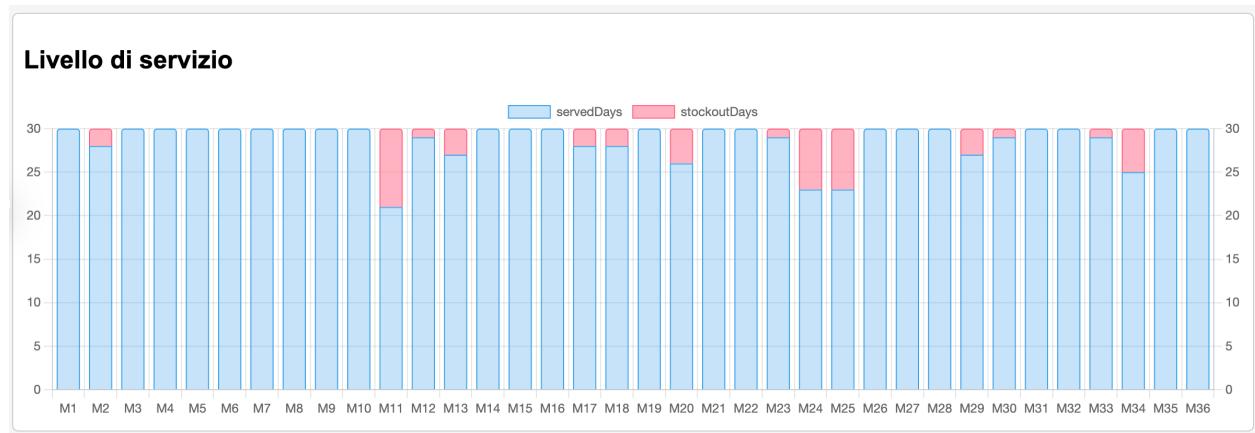
Nel modello con Safety Stock, si evidenziano solo 3 mesi in stockout, contro i 12 del modello standard senza Safety Stock.



Il grafico sottostante mostra:

- **servedDays**: i giorni per cui è stato possibile soddisfare a pieno la domanda
- **stockoutDays**: i giorni per i quali non è stato possibile soddisfare a pieno la domanda, generando ritardi nella consegna, e la generazione di backorder per i giorni successivi

Nel modello con Safety Stock, si evidenzia un numero estremamente contenuto di giorni in stockout (48), contro i 366 del modello standard senza Safety Stock.



LIMITAZIONI E FUTURI SVILUPPI

Assunzione di Normalità: Il modello assume una distribuzione normale della domanda mensile; ulteriori sviluppi potrebbero supportare distribuzioni non normali tramite tecniche di bootstrap o simulazione Monte Carlo.

Lead Time Costante: L'attuale implementazione assume lead time deterministico; si potrebbero integrare variabilità nel lead time utilizzando il framework della doppia incertezza.

Domanda Indipendente: Il modello tratta singoli SKU (Stock Keeping Unit) in isolamento; un'estensione naturale sarebbe la gestione di più prodotti con domande correlate e vincoli di capacità.

CONCLUSIONI

Questo progetto ha dimostrato con successo come i modelli classici di operations research, come il Lotto Economico di Ordinazione, possono essere efficacemente integrati con tecniche statistiche moderne per affrontare le sfide reali della gestione dell'inventario in ambienti incerti. La realizzazione del software ha confermato che:

1. Il modello EOQ rimane un pilastro della teoria di inventario, fornendo una soluzione elegante e efficiente al problema di bilanciamento tra costi di ordinazione e mantenimento.
2. L'integrazione della Scorta di Sicurezza è essenziale per operare in ambienti reali dove la domanda è soggetta a variabilità, poiché fornisce una protezione quantificabile contro il rischio di stockout.
3. Il trade-off tra costi totali e livello di servizio è trasparente e misurabile: un'azienda può scegliere consapevolmente quanto 'pagare' (in termini di costo di inventario aggiuntivo) per ottenere un determinato grado di protezione.
4. La strumentazione software appropriata è cruciale per l'implementazione pratica di questi modelli, poiché permette di processare dati reali, eseguire analisi di sensibilità e supportare decisioni gestionali consapevoli.
5. L'integrazione di competenze da multiple discipline (teoria dell'ottimizzazione, statistica, software engineering) è necessaria per creare soluzioni pratiche che aggiungono valore reale alle organizzazioni.