Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Sviluppo software in contesti vincolati: analisi, implementazione e ottimizzazione del Battery Sizing Tool

Relatore:
Prof. Angelo Di Iorio

Presentata da:
Thomas Parigi

Correlatrice:
Dott.ssa Letizia Ghirardello

Sessione III
Anno Accademico 2024/2025

Non è il talento a fare la differenza

È quella fame che non ti lascia in pace
Quella roba che senti in petto ogni giorno
Che ti tiene sveglio, che ti spinge a non mollare mai

La dopamina che senti quando arrivi a un traguardo è solo un assaggio
Ti illude per un attimo, poi ti chiede subito di alzare l'asticella

Non basta vincere una volta, serve volerlo ogni giorno
Anche quando non guarda nessuno, (ossessione)

Quando gli altri rallentano, tu spingi
Quando gli altri dormono, tu continui

Indice

Capitolo 1

Introduzione

Il presente elaborato si inserisce all'interno delle attività svolte presso *Vertiv* [9], azienda multinazionale operante nel settore delle infrastrutture per data center e delle soluzioni per la continuità elettrica. Vertiv progetta, produce e commercializza sistemi di alimentazione *UPS* (Uninterruptible Power Supply), apparati di gestione termica e numerosi servizi correlati. Una componente essenziale di tali sistemi è rappresentata dalle batterie, che garantiscono l'autonomia necessaria a mantenere in funzione le apparecchiature critiche in caso di interruzione dell'alimentazione principale.

All'interno di questo contesto industriale, la corretta selezione della tipologia e della capacità delle batterie costituisce un'operazione fondamentale sia per gli ingegneri che per i team commerciali, poiché da essa dipendono le prestazioni complessive della soluzione offerta al cliente. Tradizionalmente, tale attività veniva svolta mediante strumenti esterni, principalmente fogli Excel sviluppati nel corso degli anni, spesso separati tra loro e specifici per i diversi fornitori di batterie. Questo approccio comportava processi manuali ripetitivi, un elevato rischio di errore e tempi di configurazione non ottimali.

Il *Battery Sizing Tool*, oggetto centrale di questo lavoro, è stato progettato e sviluppato con l'obiettivo di affrontare tali problematiche e di integrare l'intero processo all'interno del software aziendale utilizzato per la configurazione e la vendita dei prodotti Vertiv.

Tale piattaforma, adottata come standard per la regione EMEA (Europe, Middle East, and Africa), costituisce l'infrastruttura di base per la creazione e

l'utilizzo di configuratori di prodotto. Questi strumenti guidano ingegneri e commerciali nella definizione delle specifiche delle macchine, permettendo la selezione di opzioni e funzionalità in modo analogo a quanto avviene nei configuratori del settore automotive. Sebbene la struttura standard del software gestisca efficacemente la logica di selezione di base per unificare e rendere compatibili le richieste dei clienti, essa mostra limiti evidenti nella gestione di logiche avanzate non previste dal framework nativo. Per colmare tale lacuna, l'ambiente di sviluppo offre una sezione dedicata alla programmazione personalizzata, vincolata tuttavia all'uso esclusivo del linguaggio JavaScript, senza possibilità di integrazione con altri linguaggi o librerie esterne.
Di conseguenza, l'architettura del *Battery Sizing Tool* risulta ibrida: la raccolta dei requisiti iniziali (input utente) sfrutta i menu a tendina nativi del sistema, mentre il complesso motore algoritmico di dimensionamento è stato interamente implementato tramite script in JavaScript.

Il tool permette infatti di:

- eliminare la dipendenza da strumenti esterni come Excel, evitando la necessità di reinserire manualmente i codici nel software di vendita;
- unificare in un unico applicativo i tool messi a disposizione dai principali fornitori di batterie, riducendo la molteplicità di strumenti da consultare e semplificando il confronto tra soluzioni diverse;
- automatizzare e velocizzare il lavoro dei venditori e degli *Application Engineer*, riducendo il numero di operazioni manuali e migliorando l'efficienza complessiva;
- fornire un algoritmo di calcolo più preciso e coerente rispetto alle valutazioni effettuate manualmente, garantendo risultati immediati e consistenti;
- rendere il processo di selezione delle batterie più *user-friendly*, consentendo anche al personale non strettamente tecnico di identificare la soluzione più appropriata senza dover necessariamente coinvolgere gli ingegneri elettrici.

La tesi non si concentra quindi sulla fase di sviluppo del tool, già conclusa, bensì sull'analisi critica dei vincoli tecnici incontrati durante la sua realizzazione e sulle conseguenze che tali limiti hanno comportato nelle scelte progettuali. Verranno inoltre messe a confronto le soluzioni implementate con quelle che sarebbe stato possibile adottare in assenza dei vincoli imposti dagli strumenti software

aziendali e dal contesto legacy. Con tale espressione ci si riferisce, nello specifico, a un ecosistema assimilabile alla definizione accademica di *Legacy System* [8]: un'infrastruttura monolitica che funge da base per i configuratori di prodotto della regione EMEA. Sebbene tale piattaforma sia critica per il business, essa si configura come un ambiente tecnologicamente rigido che limita fortemente lo sviluppo di funzionalità personalizzate. I vincoli che ne derivano, e in particolare l'assenza di connettività verso database esterni, l'impossibilità di integrare librerie di terze parti e la limitata manipolabilità dell'interfaccia utente, definiscono il perimetro tecnico obbligato entro il quale il progetto ha dovuto evolversi. Tali limitazioni hanno richiesto l'adozione di soluzioni ingegneristiche ad hoc per colmare il divario con gli standard di sviluppo attuali e garantire le funzionalità avanzate richieste.

L'obiettivo finale è mostrare come sia stato possibile raggiungere un risultato funzionale, utilizzabile e integrato nei processi aziendali nonostante le restrizioni tecnologiche, evidenziando al contempo le potenzialità future del tool qualora tali vincoli venissero meno.

Capitolo 2

Requisiti e Vincoli del Progetto

2.1 Requisiti iniziali e risultato atteso

Lo sviluppo del *Battery Sizing Tool* nasce da una precisa esigenza aziendale: ottimizzare e rendere affidabile il processo di selezione delle soluzioni di batterie per i sistemi UPS, in linea con le best practice internazionali del settore[1]

È utile ricordare che un UPS (Uninterruptible Power Supply) è un'apparecchiatura complessa posta a protezione dei carichi critici, la quale garantisce continuità elettrica convertendo l'energia immagazzinata nei banchi batterie in corrente alternata in caso di blackout. Poiché ogni UPS può essere accoppiato a numerose combinazioni di batterie differenti per chimica, capacità e numero di cabinet, l'obiettivo del tool non è solo verificare la fattibilità tecnica, bensì permettere all'utente di individuare la migliore soluzione possibile tra tutte quelle compatibili, bilanciando autonomia, ingombri e costi.

Tale approccio mira a ridurre al minimo la possibilità di errore umano e a standardizzare le procedure interne.

2.1.1 Il processo originario e le sue criticità

Prima dell'introduzione del tool, la selezione delle batterie veniva eseguita manualmente dagli ingegneri. Per ogni richiesta del cliente era necessario:

- individuare i parametri elettrici e meccanici specifici del modello UPS;
- convertire i valori presenti nei datasheet dei fornitori in grandezze utilizzabili ai fini del dimensionamento, inteso come il processo di selezione del modello e calcolo del numero esatto di blocchi (o stringhe) necessari per soddisfare l'autonomia richiesta a fronte del carico specifico;

- effettuare calcoli complessi per verificare la compatibilità delle batterie con l'autonomia richiesta;
- confrontare manualmente i possibili modelli per identificare la soluzione più efficiente.

Questo metodo comportava due limitazioni principali:

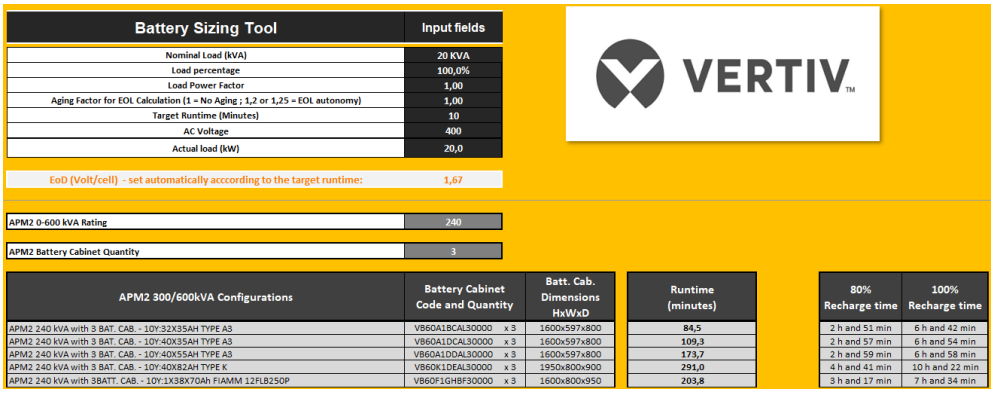
1. **Un elevato carico di lavoro**, dato l'alto numero di calcoli e verifiche richieste.
2. **Una significativa probabilità di errore umano**, dovuta alla ripetitività delle conversioni e alla quantità di dati da gestire.

Per mitigare tali problemi venne introdotto un primo tool basato su Excel: esso automatizzava parte dei calcoli attraverso un insieme articolato di formule collegate ai dati dei datasheet. Tuttavia, anche questa soluzione presentava limitazioni rilevanti:

- richiedeva comunque all'utente di selezionare e verificare manualmente numerosi parametri;
- non presentava in modo immediato tutte le soluzioni disponibili, costringendo a provare diversi modelli uno alla volta;
- non garantiva coerenza dei dati, spesso discordanti o non aggiornati;
- implicava un ulteriore passaggio manuale per copiare la configurazione scelta all'interno del software aziendale di vendita.

2.1.2 Obiettivi richiesti dall'azienda

Per superare queste difficoltà, la direzione tecnica ha richiesto la realizzazione di un unico strumento in grado di unificare l'intero processo di dimensionamento. Le richieste principali erano le seguenti.



Battery Sizing Tool	Input fields
Nominal Load (kVA)	20 KVA
Load percentage	100,0%
Load Power Factor	1,00
Aging Factor for EOL Calculation (1 = No Aging ; 1,2 or 1,25 = EOL autonomy)	1,00
Target Runtime (Minutes)	10
AC Voltage	400
Actual load (kW)	20,0
EoD (Volt/cell) - set automatically acccording to the target runtime:	1,67
APM2 0-600 kVA Rating	240
APM2 Battery Cabinet Quantity	3

APM2 300/600kVA Configurations	Battery Cabinet Code and Quantity	Batt. Cab. Dimensions HxWxD	Runtime (minutes)	80% Recharge time	100% Recharge time
APM2 240 kVA with 3 BAT. CAB. - 10Y:32X35AH TYPE A3	VB60A1BCAL30000 x 3	1600x597x800	84,5	2 h and 51 min	6 h and 42 min
APM2 240 kVA with 3 BAT. CAB. - 10Y:40X35AH TYPE A3	VB60A1DCAL30000 x 3	1600x597x800	109,3	2 h and 57 min	6 h and 54 min
APM2 240 kVA with 3 BAT. CAB. - 10Y:40X55AH TYPE A3	VB60A1DDAL30000 x 3	1600x597x800	173,7	2 h and 59 min	6 h and 58 min
APM2 240 kVA with 3 BAT. CAB. - 10Y:40X82AH TYPE K	VB60K1DEAL30000 x 3	1950x800x900	291,0	4 h and 41 min	10 h and 22 min
APM2 240 kVA with 3BATT. CAB. - 10Y:1X38X70Ah FIAMM 12FLB250P	VB60F1GH8F30000 x 3	1600x800x950	203,8	3 h and 17 min	7 h and 34 min

Figura 2.1: Versione preliminare del tool Excel utilizzato per il dimensionamento delle batterie.

Automatizzare l'intero processo di calcolo

L'azienda desiderava un algoritmo in grado di:

- calcolare automaticamente tutte le configurazioni di batterie compatibili con l'autonomia richiesta;
- generare l'intero ventaglio di soluzioni senza intervento manuale;
- garantire la correttezza matematica dei risultati, riducendo quasi a zero la possibilità di errore.

L'obiettivo era trasformare un processo complesso, manuale e soggetto a variabilità in un flusso standardizzato, rapido e affidabile.

Consentire anche ai venditori di utilizzare il sistema

Un ulteriore requisito fondamentale era rendere il tool utilizzabile non solo dagli ingegneri, ma anche dal team commerciale. I venditori, infatti, non possiedono necessariamente competenze approfondite di ingegneria elettrica o familiarità con la lettura dei datasheet tecnici forniti dai produttori. Allo stesso tempo, devono essere in grado di rispondere rapidamente alle richieste dei clienti, proponendo una soluzione adeguata senza dover interpellare ogni volta un Application Engineer.

Per questo motivo, il nuovo sistema doveva presentare un'interfaccia intuitiva e guidata, capace di mostrare esclusivamente risultati coerenti e già validati, riducendo così la possibilità di commettere errori o selezionare configurazioni non compatibili. L'obiettivo era realizzare uno strumento in grado di supportare anche utenti non tecnici, permettendo loro di individuare autonomamente la soluzione

più adatta in tempi brevi e con la certezza che il risultato fosse corretto dal punto di vista tecnico.

Come mostrato nella Figura 2.2, i tradizionali datasheet contengono tabelle tecniche ricche di valori numerici, difficili da interpretare per un utente non esperto.

DISCHARGE TABLE

Constant Current Discharge (Amperes) at 25°C (77°F)																
F.V/Time	5min	10min	15min	20min	30min	45min	1h	1.5h	2h	3h	4h	5h	6h	8h	10h	20h
1.85V/cell	30.2	20.7	16.1	12.9	9.34	6.72	5.41	3.92	3.08	2.22	1.77	1.51	1.29	1.02	0.832	0.440
1.80V/cell	32.4	21.9	16.9	13.4	9.63	6.90	5.54	4.00	3.14	2.26	1.80	1.53	1.31	1.03	0.843	0.445
1.75V/cell	34.2	22.8	17.4	13.8	9.88	7.05	5.66	4.08	3.19	2.30	1.82	1.55	1.33	1.04	0.851	0.450
1.70V/cell	35.8	23.7	18.0	14.2	10.1	7.21	5.76	4.15	3.24	2.33	1.85	1.57	1.34	1.05	0.860	0.454
1.67V/cell	37.0	24.4	18.5	14.5	10.3	7.32	5.85	4.20	3.28	2.35	1.86	1.58	1.35	1.06	0.866	0.457
1.60V/cell	39.3	25.4	19.1	14.9	10.6	7.50	5.98	4.29	3.34	2.40	1.90	1.61	1.37	1.08	0.877	0.462
Constant Power Discharge (Watts/cell) at 25°C (77°F)																
F.V/Time	5min	10min	15min	20min	30min	45min	1h	1.5h	2h	3h	4h	5h	6h	8h	10h	20h
1.85V/cell	57.1	39.4	30.8	24.8	18.1	13.0	10.5	7.66	6.03	4.37	3.49	2.98	2.55	2.02	1.65	0.880
1.80V/cell	60.8	41.4	32.1	25.7	18.5	13.3	10.8	7.79	6.13	4.43	3.53	3.02	2.59	2.04	1.67	0.890
1.75V/cell	63.3	42.8	33.0	26.2	18.9	13.6	10.9	7.91	6.22	4.49	3.58	3.05	2.62	2.06	1.69	0.900
1.70V/cell	65.7	44.2	33.9	26.9	19.3	13.8	11.1	8.03	6.30	4.55	3.62	3.09	2.64	2.08	1.71	0.907
1.67V/cell	67.4	45.2	34.6	27.4	19.6	14.0	11.2	8.11	6.36	4.59	3.65	3.11	2.67	2.10	1.72	0.914
1.60V/cell	70.1	46.5	35.6	28.0	20.0	14.3	11.4	8.30	6.46	4.66	3.71	3.15	2.70	2.13	1.74	0.924

Figura 2.2: Estratto di una tabella di scarica presente nel datasheet della batteria, contenente valori tecnici non immediatamente interpretabili per un utente non specializzato.

Mostrare tutte le soluzioni contemporaneamente

Un'altra esigenza chiave consisteva nel presentare all'utente l'intero insieme delle soluzioni possibili, ordinate e confrontabili.

Per ogni configurazione, il nuovo tool doveva mostrare:

- autonomia prevista (*Expected Runtime*): valore risultante dall'algoritmo che indica per quanto tempo la specifica configurazione di batterie è in grado di alimentare il carico dell'UPS in assenza di rete elettrica;
- numero totale di cabinet (armadi batteria) richiesti;
- ingombri fisici;
- prezzo stimato secondo la logica SPL, calcolato applicando al costo base della batteria specifici coefficienti moltiplicativi (trimmer) per garantire la corretta marginalità di guadagno sulla vendita.

Ciò permette una valutazione più rapida, trasparente e completa rispetto al tool precedente.

Introduzione di un database centralizzato

Un requisito fondamentale riguardava l'integrazione, all'interno del nuovo tool, di un database unico e centralizzato. L'obiettivo primario era la creazione di un'unica fonte di verità (Single Source of Truth), indispensabile per evitare l'utilizzo parallelo di molteplici strumenti di comparazione e garantire l'univocità del dato tecnico.

Tale archivio avrebbe dovuto contenere l'elenco completo di tutte le batterie presenti a listino, comprese le caratteristiche tecniche necessarie al dimensionamento, come capacità, tensione nominale, ingombri fisici e informazioni di compatibilità con gli UPS. Oltre ai dati principali, il database doveva includere anche i parametri utilizzati per le formule di calcolo e le conversioni richieste dai vari modelli, nonché i valori aggiornati provenienti dai diversi fornitori.

Dal punto di vista operativo, il popolamento di tale struttura non avviene in modo automatico, bensì attraverso un processo di aggregazione e inserimento manuale curato direttamente in fase di sviluppo. I dati necessari vengono raccolti grazie alla collaborazione con diversi stakeholder aziendali: da un lato i referenti tecnici che gestiscono la ricezione delle schede prestazionali dai fornitori esterni, dall'altro i dipartimenti interni responsabili della definizione dei listini prezzi e delle specifiche dimensionali degli armadi (cabinet).

L'introduzione di questa struttura dati aveva lo scopo di superare le difficoltà del processo precedente, nel quale gli utenti erano costretti a consultare manualmente i datasheet, verificare ripetutamente le stesse informazioni e utilizzare strumenti distinti per confrontare le soluzioni. Grazie al database centralizzato, tutte le informazioni diventano immediatamente disponibili, coerenti e aggiornate, eliminando duplicazioni, riducendo il rischio di errori e garantendo un flusso di lavoro più rapido e uniforme.

Unificare i tool dei fornitori

Prima dell'avvento del *Battery Sizing Tool*, il panorama degli strumenti di dimensionamento risultava estremamente frammentato. Ogni produttore di batterie forniva software proprietari, basati su interfacce e logiche di calcolo non omogenee, creando una barriera operativa per gli ingegneri costretti a:

- confrontare dati eterogenei, spesso basati su condizioni al contorno differenti (es. temperature di riferimento o tensioni di fine scarica diverse), rendendo complesso un confronto diretto e imparziale tra i brand;

- replicare calcoli simili più volte, dovendo inserire manualmente i medesimi parametri di progetto su molteplici piattaforme esterne;
- adattare manualmente il risultato al software di vendita interno, con il rischio di introdurre errori di trascrizione dei codici articolo o delle specifiche tecniche.

Il nuovo sistema risolve tale discontinuità permettendo di:

- raccogliere i dati di tutti i fornitori in un'unica piattaforma, agendo da collettore universale che astrae la complessità specifica del singolo produttore;
- ottenere risultati uniformi e immediatamente confrontabili, poiché l'algoritmo applica la medesima logica di calcolo normalizzata a tutte le batterie, garantendo una valutazione tecnica coerente (confronto "apples-to-apples");
- trasferire automaticamente le configurazioni nel sistema interno di vendita, assicurando la perfetta corrispondenza tra il dimensionamento tecnico e l'offerta commerciale.

2.1.3 Struttura, interfaccia e risultato atteso

Il flusso operativo del Battery Sizing Tool è stato strutturato per guidare l'utente in modo sequenziale, rispecchiando la logica di dimensionamento ingegneristico. La prima fase, illustrata in Figura 2.3, avviene nella sezione "Project Parameters".

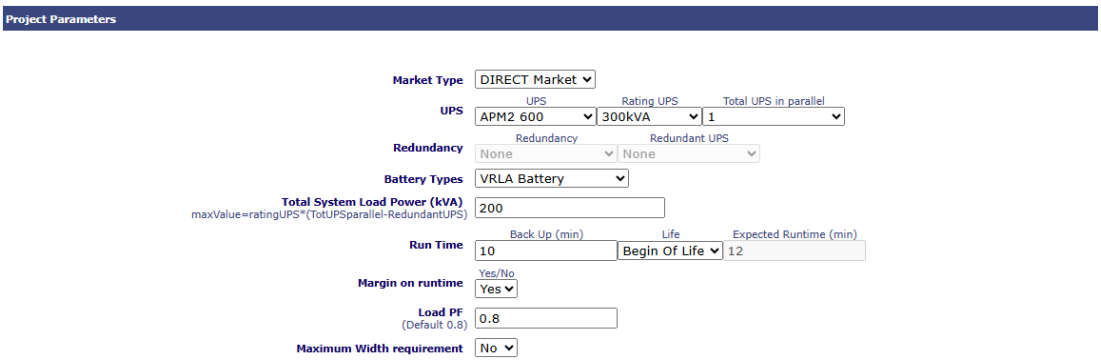


Figura 2.3: Interfaccia di inserimento dei parametri di progetto (Project Parameters).

Qui l'utente definisce i vincoli di ingresso (boundary conditions) fondamentali per l'algoritmo: il modello di UPS e la sua configurazione (singolo o parallelo), la

tipologia di batteria richiesta (es. *VRLA o Litio*), il carico effettivo del sistema (*Total System Load Power*) e l'autonomia target (*Run Time*). Il sistema permette inoltre di raffinare la ricerca applicando margini di sicurezza sull'autonomia o specifici vincoli dimensionali (*Maximum Width requirement*).

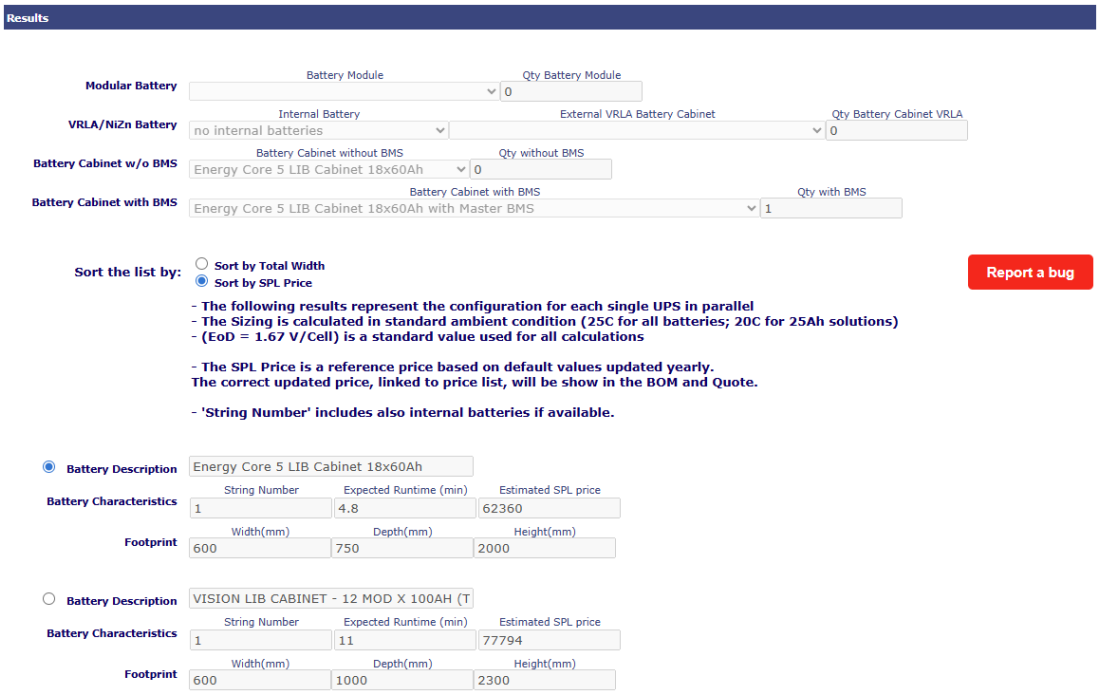


Figura 2.4: Esempio di output del Battery Sizing Tool con le soluzioni calcolate.

A valle dell'elaborazione, i risultati vengono presentati nella sezione "Results" (Figura 2.4), progettata per integrarsi visivamente nel framework legacy del software di vendita. Questa schermata si divide in due macro-aree funzionali:

- **Area di Integrazione (parte superiore):** contiene i campi necessari (es. *Modular Battery, Battery Cabinet*) per la compilazione automatica della distinta base (BOM). La selezione di una soluzione popola istantaneamente questi campi, garantendo il trasferimento dei dati al carrello di vendita senza trascrizioni manuali.

- **Lista delle Soluzioni (parte inferiore):** mostra l'output dell'algoritmo JavaScript. Ogni riga rappresenta una configurazione validata, dettagliando parametri critici quali il codice articolo, l'autonomia effettiva calcolata (*Expected Runtime*), il prezzo stimato secondo logica SPL e l'ingombro a terra (*Footprint*). Le opzioni di ordinamento (*Sort by*) consentono infine di riorganizzare le proposte in base alle priorità del cliente (es. minor prezzo o minor ingombro).

Il risultato finale risponde quindi puntualmente ai requisiti aziendali, offrendo uno strumento:

- veloce e centralizzato;
- affidabile e privo di errori;
- utilizzabile da tutto il personale;
- integrato con il software aziendale di vendita;
- capace di migliorare la qualità e la rapidità delle offerte.

Il tool non deve essere inteso soltanto come un esecutore di calcoli, ma come un elemento abilitante che trasforma l'intero processo di gestione delle richieste, introducendo un metodo più efficiente, accurato e scalabile rispetto agli strumenti precedenti.

2.2 Vincoli riscontrati durante lo sviluppo

Durante la progettazione e l'implementazione del tool sono emerse numerose limitazioni tecnologiche, metodologiche e operative che hanno influenzato in modo significativo le scelte progettuali. Questi vincoli hanno determinato la struttura interna del software, l'organizzazione dei dati, le funzionalità disponibili e la qualità dell'interfaccia.

Per offrire una visione d'insieme delle problematiche affrontate, la Tabella 2.1 riassume le cinque macro-categorie di vincoli che verranno analizzate nei paragrafi successivi, associando a ciascuna una breve descrizione dell'impatto sul progetto.

Tipologia di Vincolo	**Descrizione Sintetica**
Gestione dei Dati	Impossibilità di connettersi a database esterni (SQL/NoSQL); necessità di incorporare i dati staticamente (hardcoded) nel codice sorgente JavaScript.
Software e Tecnologici	Ambiente legacy chiuso che impedisce l'integrazione di librerie esterne, framework moderni o API di terze parti; utilizzo esclusivo di JavaScript nativo [2].
Architetturali e Strutturali	Struttura della pagina HTML (DOM) predefinita e immodificabile dal codice, che vincola rigidamente il layout e la personalizzazione grafica.
Computazionali e Interoperabilità	Impossibilità di eseguire elaborazioni pesanti lato server o utilizzare linguaggi di analisi dati (es. Python) integrati nel runtime del browser.
Navigazione e Usabilità	Vincolo di architettura "Single Page" imposto dal sistema ospitante: impossibilità di creare nuove pagine, popup o percorsi di navigazione multi-step.

Tabella 2.1: Sintesi tassonomica dei principali vincoli tecnici affrontati nello sviluppo.

Prima di analizzare le singole categorie di vincoli, è doveroso definire il conte-

sto tecnologico in cui il progetto è stato sviluppato. L'infrastruttura di riferimento è costituita da una piattaforma software di terze parti, adottata da Vertiv come standard aziendale per i processi di vendita. Si tratta, nello specifico, di un sistema CPQ (*Configure, Price, and Quote*) progettato per centralizzare la gestione del complesso catalogo prodotti dell'azienda[3].

La funzione primaria di tale ambiente è consentire ai team di vendita e agli ingegneri di elaborare configurazioni tecnicamente validate, verificando le compatibilità tra i componenti e generando automaticamente la documentazione ufficiale di offerta (quotazioni in formato PDF) da inviare al cliente. Sebbene il sistema offra strumenti visuali per la gestione delle logiche di prodotto standard, esso prevede una sezione dedicata allo sviluppo personalizzato (scripting) per far fronte a esigenze avanzate. Tuttavia, tale ambiente di sviluppo è fortemente vincolato: l'unico linguaggio di programmazione supportato è JavaScript, affiancato da limitate possibilità di marcatura HTML, il tutto confinato all'interno di una struttura grafica (layout) predefinita e largamente immutabile.

In questo scenario, il *Battery Sizing Tool* è stato architettato come una soluzione ibrida: esso sfrutta il configuratore nativo del software CPQ come contenitore e interfaccia di input, ma delega l'intera logica computazionale, il filtraggio avanzato e la gestione dei dati (database interno) a un motore JavaScript custom integrato. Tale approccio è stato necessario per colmare il divario tra le funzionalità standard del CPQ e i complessi requisiti matematici richiesti per il dimensionamento delle batterie.

Nei seguenti sottoparagrafi vengono illustrati nel dettaglio i principali vincoli affrontati, classificati per tipologia.

2.2.1 Vincoli legati alla gestione dei dati

Uno dei vincoli più rilevanti ha riguardato l'impossibilità di utilizzare database esterni per la memorizzazione e la consultazione dei dati tecnici relativi alle batterie. Il software aziendale che ospita il Battery Sizing Tool non consente infatti l'accesso a risorse esterne, né locali né in cloud, impedendo qualsiasi forma di integrazione con database relazionali o non relazionali.

Questa limitazione ha imposto la necessità di incorporare internamente, direttamente nel codice, l'intero set di dati estratto dai datasheet dei diversi modelli di batterie. L'intero processo ha richiesto numerose operazioni manuali e semi-automatiche:

- raccolta dei dati da ogni datasheet tecnico in formato Excel;
- conversione dei file Excel in formato JSON;
- caricamento dei dati JSON all'interno dell'algoritmo tramite array di oggetti JavaScript;
- gestione dei calcoli direttamente su dati statici, anziché interrogare dinamicamente un database.

L'assenza di un database esterno ha comportato un notevole appesantimento dell'algoritmo, sia in termini di manutenzione del codice sia in termini di consumo di memoria. L'inserimento manuale e centralizzato di grandi quantità di dati modifica sostanzialmente il paradigma di sviluppo: il tool non può aggiornarsi automaticamente né può attingere selettivamente solo alle informazioni necessarie, ma deve caricare in memoria l'intero dataset in modo statico.

2.2.2 Vincoli software e tecnologici

Il software di riferimento, basato su un'infrastruttura legacy poco flessibile, non consente l'integrazione di librerie esterne, API o estensioni aggiuntive rispetto a quelle standard già incluse nell'ambiente JavaScript fornito nativamente[6]. Questo vincolo ha limitato profondamente le possibilità di implementare funzionalità avanzate, sia dal punto di vista dell'interattività sia dell'elaborazione dei dati.

La mancanza di librerie grafiche moderne, ad esempio, impedisce la generazione di grafici dinamici o visualizzazioni avanzate come curve di scarica, confronti visivi tra batterie o integrazione di elementi UI più evoluti. L'interfaccia utente è così rimasta necessariamente essenziale, penalizzando la comprensione immediata dei risultati tecnici e riducendo il grado di user-friendliness del tool.

Analogamente, l'impossibilità di importare API di calcolo o servizi esterni riduce l'estensibilità futura dello strumento e ne condiziona fortemente la modularità. Qualsiasi miglioramento strutturale deve essere sviluppato manualmente e senza il supporto di componenti esterne consolidate.

2.2.3 Vincoli architetturali e strutturali del software

Un ulteriore vincolo significativo riguarda la rigidità dell'architettura del software utilizzato dall'azienda per il comparto vendite. L'ambiente mette a disposi-

zione un Document Object Model (DOM) [4] predefinito e non modificabile, che non consente interventi rilevanti sulla struttura grafica delle pagine, né l'aggiunta di nuovi layout, stili o componenti non previsti dal template originale.

In particolare, non è possibile:

- creare nuove pagine di navigazione interne;
- aggiungere tabelle con formattazioni personalizzate;
- modificare font, dimensioni o stili;
- introdurre elementi grafici propri o componenti UI aggiuntivi.

Questo limita fortemente la progettazione dell'interfaccia del Battery Sizing Tool, che non può evolversi verso soluzioni più moderne o intuitive. Persino modifiche apparentemente semplici, come l'aggiunta di una riga in una tabella o la definizione di uno stile testuale differenziato, risultano impossibili senza violare i vincoli del sistema.

2.2.4 Vincoli computazionali e di interoperabilità

Un ulteriore ostacolo ha riguardato l'impossibilità di integrare l'algoritmo con Python, linguaggio esterno utilizzato nella fase preliminare per la riduzione dei dati tramite l'algoritmo di Douglas-Peucker (*Douglas-Peucker algorithm*)[5].

La scelta di adottare questa specifica tecnica è maturata a seguito di un confronto tecnico con il team di ingegneria. Attingendo al background accademico dei colleghi, l'algoritmo è stato identificato come la soluzione ottimale per risolvere il "collo di bottiglia" causato dall'eccessiva mole di dati, permettendo così di alleggerire il carico sul database e velocizzare le operazioni di calcolo del tool.

In un contesto privo di vincoli, tale procedura avrebbe potuto essere incorporata direttamente all'interno del tool, permettendo di automatizzare la selezione dei punti significativi e ridurre in modo nativo il numero di campioni da caricare. A causa delle restrizioni del software legacy, si è reso necessario implementare un processo articolato e suddiviso in più fasi:

1. estrazione dei dati grezzi da ciascun datasheet e caricamento in un primo file Excel;
2. esecuzione esterna dello script Python per ridurre i dati da oltre 1000 punti a circa 50;

3. esportazione dei risultati in un secondo file Excel ridotto;

4. conversione manuale del file in JSON;

5. inserimento finale dei dati ridotti nel codice JavaScript del Battery Sizing Tool.

È fondamentale precisare che l'adozione di questo processo ibrido non è stata immediata, ma ha seguito un rigoroso percorso di validazione ("proof of concept"). Prima dell'integrazione ufficiale, è stata condotta un'approfondita fase di test a campione sui risultati generati dall'algoritmo. Tale procedura di controllo qualità ha permesso di certificare che la drastica riduzione dei punti non comportasse la perdita di informazioni significative, scongiurando il rischio di errori di approssimazione che avrebbero potuto compromettere l'affidabilità tecnica del dimensionamento.

Questo workflow aggiuntivo, ripetuto per ogni modello di batteria, ha comportato un importante dispendio di tempo e ha reso il processo di aggiornamento dei dati non automatizzabile. In assenza di vincoli, tutte queste operazioni sarebbero state integrate direttamente nell'algoritmo, con evidenti benefici in termini di efficienza e manutenzione.

2.2.5 Vincoli sulla navigazione e sull'usabilità

Il sistema utilizzato non permette l'inserimento di link esterni né la creazione di nuove schermate indipendenti. Di conseguenza, non è possibile ampliare l'esperienza di navigazione attraverso pagine aggiuntive che offrano sezioni dedicate all'analisi, alla consultazione del database o alla visualizzazione dettagliata dei risultati.

L'impossibilità di organizzare i contenuti in più pagine web comporta un'interfaccia inevitabilmente densa, con limitazioni nell'esposizione delle informazioni tecniche. Un esempio evidente riguarda l'impossibilità di creare una pagina dedicata alla visualizzazione dei dati utilizzati nei calcoli o di predisporre confronti strutturati tra differenti configurazioni di batterie. La mancanza di navigazione modulare riduce inoltre il potenziale del tool come strumento formativo o decisionale per gli utenti finali.

Capitolo 3

Architettura del Tool e Soluzioni implementate

3.1 Architettura e funzionamento del Battery Sizing Tool

Il Battery Sizing Tool è stato progettato come un sistema autonomo e completamente integrato nel software aziendale utilizzato per la gestione delle offerte commerciali. L'intera logica applicativa è implementata in un unico file `JavaScript`, che funge sia da motore di calcolo sia da contenitore del database interno necessario per le elaborazioni. Tale file include un insieme di metodi dedicati alla gestione degli input, all'esecuzione dei calcoli e alla generazione delle soluzioni finali; inoltre contiene strutture dati organizzate in array di oggetti che rappresentano le caratteristiche tecniche delle diverse batterie disponibili.

La raccolta dei dati avviene attraverso una serie di campi selezionabili dall'utente direttamente nel front-end del gestionale di vendita. L'interfaccia grafica riproduce tutte le scelte commerciali e tecniche richieste dai processi aziendali standard, consentendo al personale di definire il profilo di progetto richiesto dal cliente mediante una configurazione guidata. L'integrazione totale con il sistema di vendita costituisce uno dei principali risultati evidenziati nella sezione dei requisiti: il tool, infatti, permette di unificare e sostituire diversi strumenti preesistenti, riducendo il numero di passaggi necessari per ottenere una configurazione valida e accelerando significativamente la preparazione delle offerte.

3.1.1 Struttura della schermata utente

L'interfaccia del Battery Sizing Tool è suddivisa in tre sezioni principali, corrispondenti alle fasi logiche del processo di configurazione.

Project Parameters

La parte superiore della schermata (Figura 3.1) è denominata *Project Parameters* ed è dedicata all'inserimento dei parametri iniziali del progetto. In questa sezione l'utente specifica:

- il prodotto UPS selezionato e la relativa potenza nominale;
- il numero di UPS in parallelo e la presenza di ridondanza;
- il tipo di batteria desiderata (VRLA o Litio);
- il carico totale richiesto;
- l'autonomia necessaria e il margine di sicurezza da applicare;
- altri parametri operativi come il Power Factor o eventuali vincoli dimensionali.

Tutti i campi sono collegati a specifici metodi del motore `JavaScript`, che raccolgono i valori selezionati e li trasformano in variabili utilizzate nella fase di calcolo.

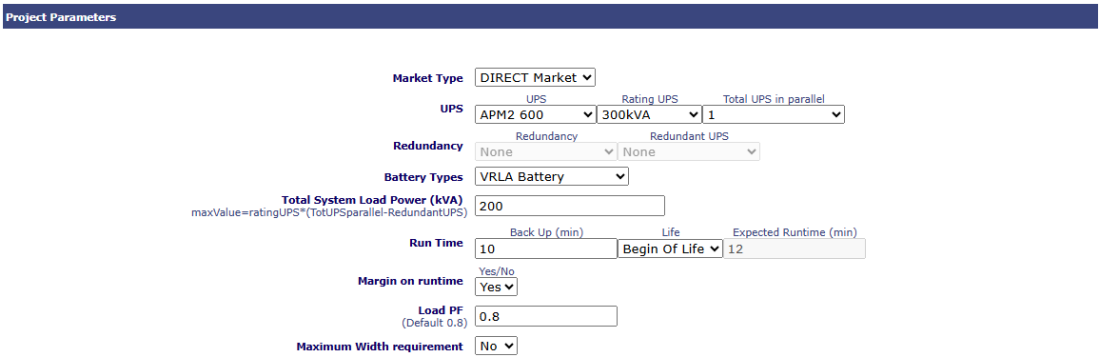


Figura 3.1: Interfaccia della sezione *Project Parameters*.

Results

La sezione centrale della schermata (Figura 3.2) è denominata *Results* ed è il cuore del tool. Essa è composta da due aree distinte:

- **Area di integrazione** – mostra i campi necessari per popolare automaticamente il gestionale di vendita una volta scelta la batteria. Questa parte è fondamentale per la generazione della BOM finale.
- **Area delle soluzioni compatibili** – viene popolata dopo l'esecuzione dell'algoritmo e presenta tutte le configurazioni di batteria che soddisfano i parametri inseriti.

L'utente può ordinare le soluzioni tramite un radio button secondo due criteri:

- prezzo SPL stimato;
- larghezza totale della soluzione.

La selezione di uno dei criteri richiama l'algoritmo di calcolo, che rielabora i dati e aggiorna dinamicamente la lista delle soluzioni proposte.

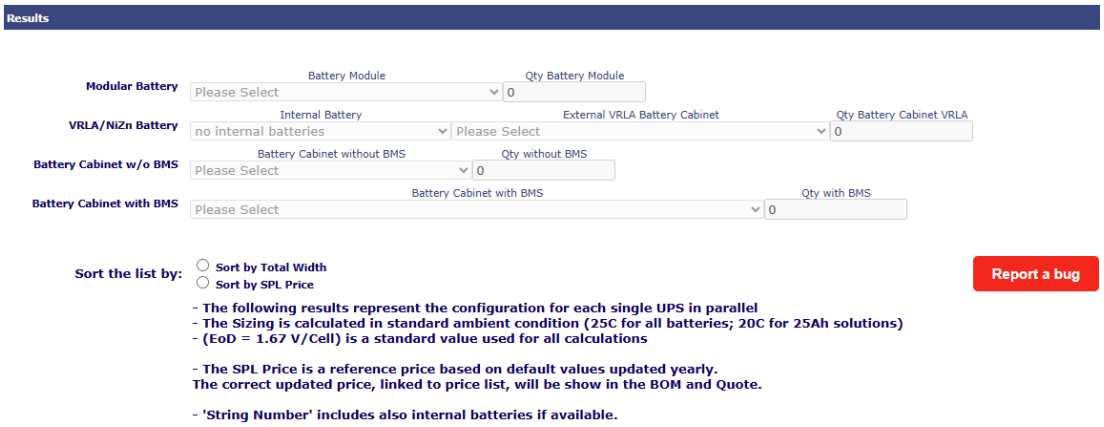


Figura 3.2: Sezione *Results* prima della selezione di una soluzione.

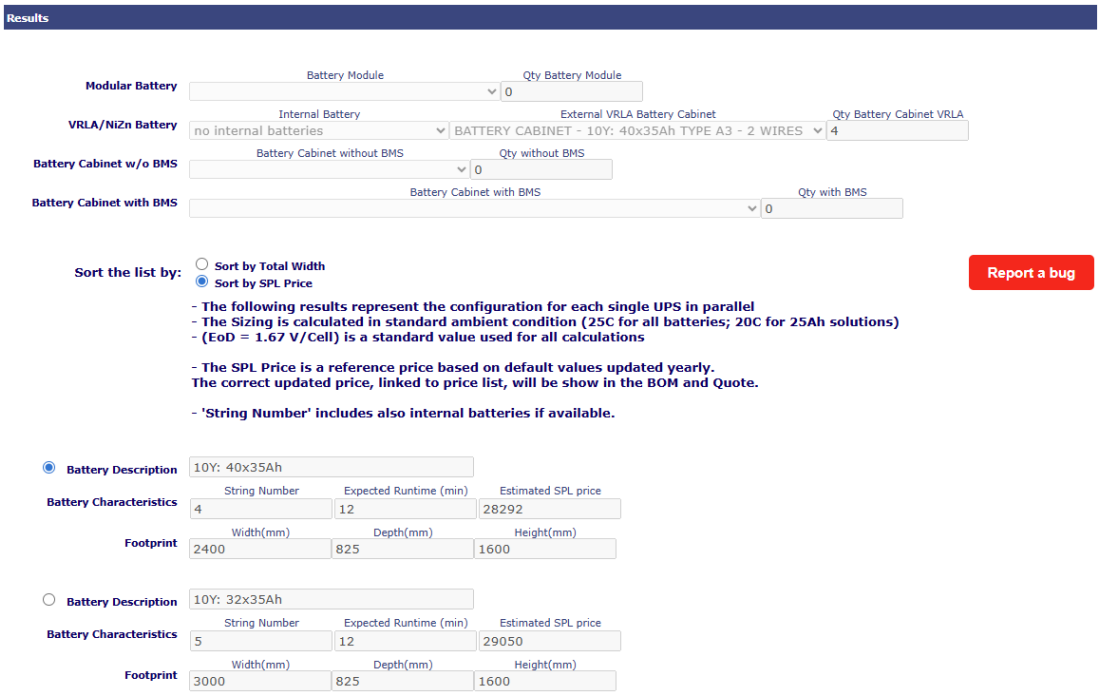


Figura 3.3: Esempio di soluzioni di batteria compatibili con i parametri selezionati.

Una volta popolata, la sezione presenta per ogni soluzione una descrizione del modello di batteria, il numero di stringhe previste, l'autonomia stimata, gli ingombri fisici (espressi in L × P × H) e un prezzo SPL indicativo.

L'utente può selezionare la configurazione preferita tramite il pulsante dedicato posto accanto a ciascuna soluzione.

Generazione della BOM

Dopo la selezione, il tool compila automaticamente i campi dell'area di integrazione, consentendo al software di vendita di generare la *Bill of Materials* (BOM) completa del progetto (Figura 3.4). La BOM contiene tutti i codici articolo necessari, comprese le batterie selezionate, e rappresenta il documento finale da allegare alla quotazione destinata al cliente.

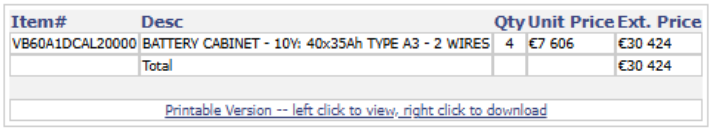


Figura 3.4: Esempio di BOM generata dal gestionale dopo la selezione della soluzione.

In questo modo il Battery Sizing Tool garantisce la continuità operativa tra algoritmo tecnico e sistema commerciale, eliminando passaggi manuali, riducendo gli errori e uniformando il processo di configurazione delle soluzioni batterie.

3.2 Soluzioni implementate per la gestione dei vincoli

Lo sviluppo di un applicativo software non può prescindere da un'analisi approfondita delle scelte implementative. Come insegnato dagli studi accademici in ambito algoritmico e ingegneristico, la definizione delle strategie di progettazione costituisce il fulcro della buona riuscita di qualsiasi progetto informatico. Prima ancora di procedere con la scrittura del codice è necessario comprendere i punti di forza del contesto applicativo, individuare i colli di bottiglia e valutare in che modo affrontare i potenziali problemi di implementazione.

Nel caso del *Battery Sizing Tool*, una parte rilevante del lavoro preliminare ha riguardato la raccolta delle informazioni dai vari *stakeholder* coinvolti: responsabili commerciali, tecnici specializzati, esperti del software di vendita e referenti del reparto che gestisce i dati batteria. Il confronto con tali figure ha permesso di chiarire quali fossero le funzionalità richieste, le criticità note, i limiti strutturali dell'ambiente di sviluppo e le possibili incongruenze nella disponibilità dei dati.

L'obiettivo di questo capitolo è quindi quello di descrivere in modo puntuale i principali vincoli individuati nel Capitolo 2 e illustrare le soluzioni tecniche adottate per aggirarli o superarli, mantenendo invariato il risultato atteso dal punto di vista funzionale e qualitativo.

3.2.1 Gestione dei dati in assenza di un database esterno

Il primo e più rilevante vincolo affrontato durante lo sviluppo del *Battery Sizing Tool* riguarda l'impossibilità di integrare un database esterno interrogabile dal software di vendita. Tale sistema, infatti, non consente connessioni dirette verso fonti dati esterne né l'uso di servizi intermedi per la gestione delle informazioni. Questo limite ha reso necessario studiare una soluzione che permettesse di accedere in modo rapido e strutturato a un insieme eterogeneo di dati tecnici senza poter ricorrere a strumenti tradizionali come database relazionali, endpoint API o file esterni consultabili dinamicamente.

Per comprendere se esistessero precedenti o approcci consolidati, è stata avviata un'interlocuzione con diversi colleghi in altre sedi europee. Tuttavia, nessuno degli interlocutori aveva mai affrontato un problema di natura analoga, né risultavano disponibili implementazioni che potessero essere riutilizzate o adattate. Ciò ha comportato la necessità di definire una soluzione originale e pienamente compatibile con i vincoli dell'ambiente.

Dopo un confronto con il team tecnico, la scelta più efficiente è stata quella di incorporare direttamente nel codice dell'algoritmo dei veri e propri "micro-database", implementati come array di oggetti JavaScript. Questa soluzione permette di avere all'interno del runtime tutti i dati necessari, senza necessità di interrogazioni esterne.

Il primo insieme di dati strutturati implementato è stato quello riguardante i parametri descrittivi delle batterie: nome visualizzato in output, dimensioni fisiche del cabinet, prezzo indicativo e altri attributi necessari per il calcolo o per la visualizzazione dei risultati. Tale struttura funge da archivio interno in cui vengono aggiunte, quando necessario, nuove batterie da integrare nel sistema.

Poiché l'integrazione diretta di file Excel non era possibile, si è reso necessario convertire i datasheet forniti dall'azienda in un formato nativamente compatibile con JavaScript. La soluzione adottata è stata la conversione dei file Excel, suddivisi in più fogli e colonne, in due array di oggetti JSON distinti: `valoriBOL` e `valoriEOL`, che rappresentano rispettivamente i valori *Begin of Life* e *End of Life* delle batterie. La distinzione è fondamentale poiché i valori tecnici variano sensibilmente in base allo stato di vita dell'accumulatore.

È stato poi necessario integrare un terzo array di oggetti, dedicato all'efficienza dei gruppi di continuità (UPS) in funzione del carico totale e del carico effettivamente richiesto. Tale struttura, derivata da tabelle complesse e ricche di valori simili tra loro, è stata semplificata mediante operazioni preliminari di raffinamento dei dati, così da consentire interrogazioni più rapide e selezioni prive di ambiguità.

In sintesi, l'algoritmo incorpora un unico grande "database" interno composto da diversi array di oggetti JSON:

- **valoriBOL**: valori forniti dal produttore relativi alla fase iniziale di vita della batteria;
- **valoriEOL**: valori relativi alla fase di fine vita;

- **attributiCabinet**: parametri geometrici e descrittivi dell'armadio batteria;
- **AcDcEfficiency**: efficienza dei vari modelli UPS in funzione della percentuale di carico.

Questi insiemi di dati vengono filtrati dinamicamente sulla base dei parametri impostati dall'utente nella sezione *Project Parameters*. Il tipo di UPS e il relativo rating, combinati con il valore di *Total System Load Power*, consentono di determinare la percentuale di carico e di selezionare l'efficienza corretta dal database interno.

Il tipo di batteria rappresenta il primo livello di filtro: il software di vendita limita già la scelta ai modelli compatibili con l'UPS selezionato. Una volta avviato il calcolo, l'algoritmo processa sequenzialmente ciascun modello di batteria filtrando i valori in base allo stato di vita (BOL o EOL), al nome del modello e all'autonomia richiesta.

Se il valore di autonomia è presente nel database, l'algoritmo ne utilizza direttamente il valore associato. In caso contrario, viene eseguita un'interpolazione lineare tra i due valori adiacenti così da ottenere un dato più preciso e coerente con i requisiti tecnici.

Una volta completato il filtraggio e ottenuti i valori necessari, il sistema esegue i calcoli per la prima batteria, genera il relativo output e procede iterativamente con tutte le batterie compatibili.

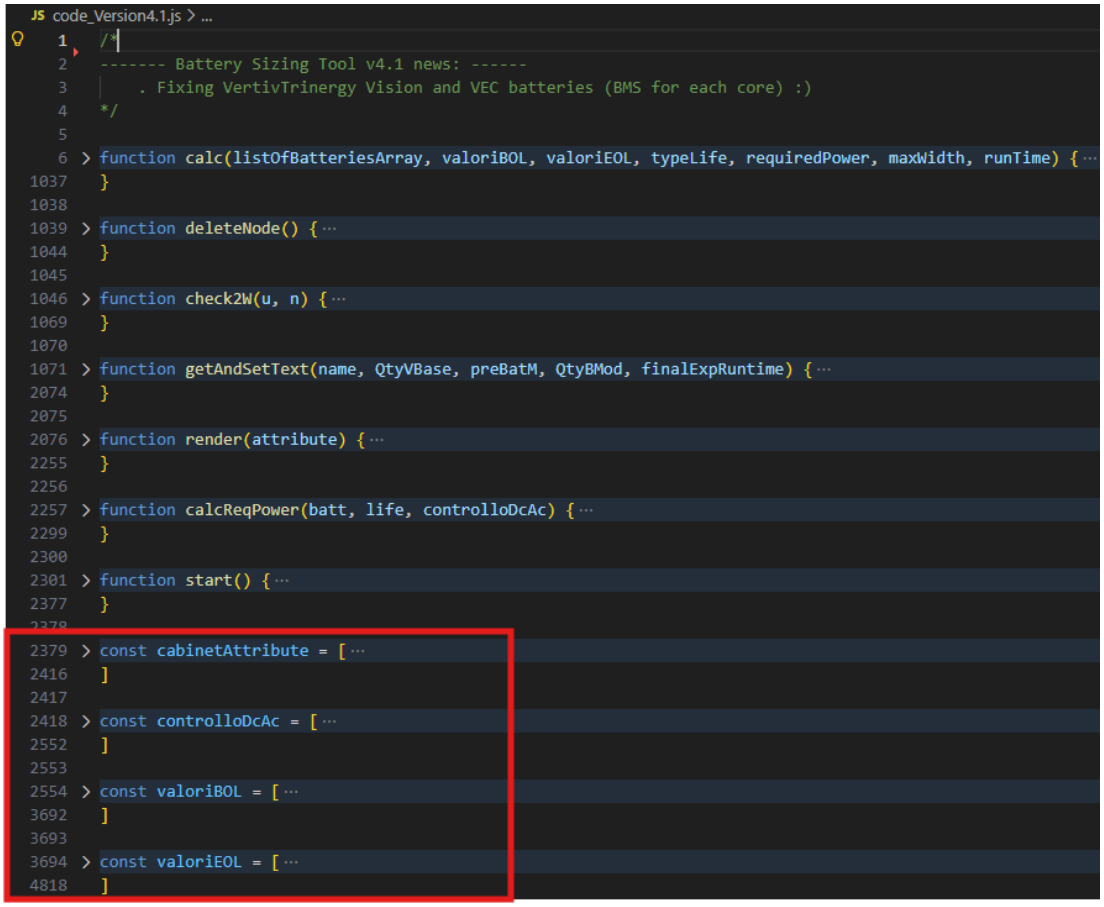


Figura 3.5: Estratto del file JavaScript: in evidenza le strutture dati che compongono il database interno del *Battery Sizing Tool*.

Come mostrato in Figura 3.5, la parte finale del file JavaScript contiene le strutture dati che costituiscono il database interno del *Battery Sizing Tool*. Questa sezione del codice comprende gli array `cabinetAttribute`, `controlloDcAc`, `valoriBOL` e `valoriEOL`, impiegati dall'algoritmo per filtrare e combinare i dati necessari ai calcoli di dimensionamento. A causa della loro estensione, tali strutture non vengono riportate integralmente, ma è stata inclusa un'anteprima per evidenziare la loro collocazione all'interno del programma.

3.2.2 Limitazioni dovute al software legacy e implementazione senza API o librerie esterne

Una delle difficoltà principali incontrate durante lo sviluppo del *Battery Sizing Tool* è stata l'impossibilità di utilizzare API o librerie esterne, sia per l'elaborazione dei dati sia per potenziali funzioni aggiuntive come la raccolta di statistiche o la visualizzazione avanzata dei risultati. Questa limitazione è dovuta alla natura

legacy del software di vendita che ospita il tool, il quale consente esclusivamente l'utilizzo di funzioni JavaScript di base, senza possibilità di integrazione di componenti esterni.

In un contesto ideale sarebbe stato possibile arricchire il sistema con strumenti già pronti per:

- raccogliere informazioni sull'utilizzo del tool (utenti, richieste, prodotti più configurati);
- aggiungere grafici dinamici aggiornati in tempo reale in base ai parametri selezionati;
- filtrare e interrogare rapidamente il database tramite metodi forniti da librerie come `lodash` o `underscore`;
- eseguire interpolazioni e operazioni matematiche ottimizzate tramite apposite API.

Tuttavia, nessuna di queste funzionalità era implementabile nel sistema disponibile. È stato pertanto necessario sviluppare manualmente tutte le operazioni avanzate, ricreando da zero molte funzioni comunemente fornite da librerie esterne.

Filtraggio dei dati senza funzioni avanzate

Un esempio significativo riguarda l'interpolazione lineare dei valori di autonomia. Poiché non era possibile affidarsi a funzioni matematiche preesistenti, è stato necessario:

1. filtrare manualmente i valori del database in base al nome della batteria;
2. scorrere sequenzialmente tutte le righe dell'array fino a individuare il valore immediatamente precedente e quello successivo al valore richiesto;
3. calcolare l'interpolazione tramite una formula implementata manualmente;
4. restituire il valore interpolato come input per i calcoli successivi.

Questa procedura, normalmente automatizzata e ottimizzata da librerie esterne, ha richiesto un'implementazione completamente artigianale, con impatto diretto sia sulla complessità del codice sia sulle prestazioni generali.

Assenza di librerie grafiche

Una ulteriore conseguenza dell'ambiente *legacy* è stata l'impossibilità di integrare librerie per la visualizzazione grafica dei risultati. L'aggiunta di grafici esplicativi avrebbe permesso agli ingegneri di comprendere più facilmente l'andamento delle prestazioni della batteria selezionata e di valutarne la piena compatibilità con le richieste del cliente. Questa componente, pur essendo altamente desiderabile, non è ancora stata implementata nel tool poiché richiede una soluzione che superi i vincoli imposti dal sistema attuale.

Mancata integrazione dell'algoritmo di Douglas–Peucker

Una limitazione collegata riguarda anche l'impossibilità di integrare l'algoritmo di Douglas–Peucker direttamente nel file JavaScript del tool. Questo algoritmo sarebbe stato essenziale per ridurre in modo efficiente il dataset contenente i valori di autonomia. A causa del vincolo tecnico, è stato necessario sviluppare un processo alternativo esterno, scritto in Python, come verrà descritto nel dettaglio nella sottosezione 3.2.4.

Popolamento dinamico del DOM senza funzioni avanzate

Anche l'aggiornamento della pagina in risposta alle modifiche dei parametri utente ha richiesto soluzioni manuali. Per garantire un refresh completo e coerente dei risultati, è stato necessario:

- raggruppare tutti gli elementi dinamici all'interno di un unico nodo del DOM;
- eliminare e ricreare tale nodo ad ogni modifica dei parametri;
- implementare una funzione dedicata per il popolamento dei campi tramite operazioni di `get` e `set`, con controlli sul corretto inserimento dei valori.

Questa procedura garantisce un aggiornamento affidabile dell'interfaccia ma richiede un lavoro manuale significativo, che in altri contesti sarebbe gestito automaticamente da framework o librerie specializzate.

In sintesi, a causa delle restrizioni del software di vendita, l'intero sistema è stato sviluppato utilizzando esclusivamente JavaScript nativo, senza l'ausilio di strumenti esterni. Questo ha richiesto un notevole lavoro di progettazione, la riscrittura manuale di funzioni comunemente disponibili in librerie esterne e la

ricerca di soluzioni efficaci per mantenere alta la qualità dell'esperienza utente nonostante le limitazioni tecniche.

3.2.3 Adattamento a un DOM non modificabile nel software di vendita

Un ulteriore vincolo significativo ha riguardato l'impossibilità di modificare liberamente il *Document Object Model* (DOM) della piattaforma di vendita all'interno della quale il *Battery Sizing Tool* è integrato. Il software, essendo basato su un'architettura legacy, consente esclusivamente l'utilizzo di strutture HTML già previste dal sistema, impedendo l'introduzione di nuove sezioni, elementi grafici o componenti interattivi. Questo aspetto ha influenzato in modo sostanziale sia la progettazione dell'interfaccia sia la gestione dei contenuti generati dal tool.

In particolare, non è stato possibile introdurre tabelle dinamiche, grafici interattivi o sezioni personalizzate utili a migliorare la leggibilità dei risultati. Ogni elemento visivo doveva obbligatoriamente conformarsi alla struttura esistente, utilizzando unicamente le tabelle e i campi già presenti nel gestionale. Tale limitazione ha reso impraticabile, ad esempio, la creazione di pagine dedicate per visualizzare:

- l'insieme dei dati tecnici utilizzati per i calcoli,
- listini prezzi aggiuntivi o opzioni complementari,
- informazioni avanzate destinate agli Application Engineer.

L'assenza di un meccanismo di navigazione interna ha ulteriormente complicato l'esperienza utente, rendendo impossibile strutturare il tool come un vero e proprio mini–sito web articolato su più schermate. Qualsiasi informazione doveva essere mostrata all'interno di un'unica pagina, aumentando la complessità del layout e dei meccanismi di aggiornamento dell'interfaccia.

Dal punto di vista implementativo, per rispettare le restrizioni imposte dal software di vendita, è stato necessario replicare all'interno del tool la struttura delle tabelle predefinite. Le sezioni grafiche sono state generate tramite un metodo dedicato (`render()`), sviluppato manualmente in JavaScript, che ricrea per ogni soluzione di batteria una serie di elementi con la stessa formattazione, gli stessi stili e le stesse dimensioni utilizzate dal gestionale. Non disponendo della

possibilità di utilizzare librerie esterne per il rendering, l'intero processo è stato implementato tramite manipolazione diretta del DOM.

Questa scelta è stata obbligatoria anche per quanto riguarda la gestione dei campi necessari alla creazione della *Bill of Materials* (BOM). Poiché non esiste un collegamento diretto tra la selezione della soluzione preferita e la BOM finale, è stato necessario introdurre campi nascosti (*hidden fields*) o sezioni dedicate all'interno delle tabelle esistenti, nei quali salvare temporaneamente i valori selezionati. Questa soluzione, pur funzionando correttamente, comporta un maggiore utilizzo di memoria e contribuisce a incrementare la complessità del codice.

Conseguenze pratiche delle limitazioni del DOM. Le restrizioni imposte dal DOM non modificabile hanno avuto un impatto significativo sia sulla progettazione sia sull'evoluzione futura del tool. In particolare, esse hanno determinato:

- un aumento considerevole del numero di linee di codice necessarie per replicare manualmente strutture altrimenti gestibili da librerie esterne;
- una maggiore complessità nella manutenzione del tool, dovuta alla presenza di numerose funzioni dedicate al popolamento e alla gestione delle tabelle;
- la necessità di effettuare test approfonditi per evitare errori nel riempimento delle celle o nella sincronizzazione dei valori tra interfaccia e BOM;
- una ridotta flessibilità nell'introdurre modifiche future all'interfaccia grafica;
- un peggioramento delle prestazioni nelle situazioni in cui il numero di soluzioni da visualizzare è elevato, a causa dei ripetuti aggiornamenti del DOM.

In sintesi, le restrizioni relative alla struttura del DOM hanno rappresentato uno dei vincoli più significativi dell'intero progetto, richiedendo soluzioni creative e un attento lavoro di integrazione per garantire un risultato coerente con le necessità del sistema.

3.2.4 Riduzione dei dati tramite algoritmo esterno e vincoli di integrazione

Uno dei principali vincoli affrontati nello sviluppo del *Battery Sizing Tool* riguarda l'impossibilità di utilizzare un database esterno e, allo stesso tempo, la

necessità di limitare il numero di dati caricati nel codice JavaScript. I datasheet forniti dai produttori includono tipicamente centinaia, talvolta migliaia di valori sperimentali relativi ai tempi di scarica delle batterie. Inserire integralmente tali dataset all'interno del tool avrebbe comportato un consistente appesantimento in termini di memoria, tempi di caricamento e difficoltà manutentive.

Poiché il software di vendita non consente l'esecuzione di codice Python né l'integrazione diretta di algoritmi complessi esterni, è stato necessario ricorrere a un processo di pre-elaborazione svolto completamente al di fuori del tool. Tale processo è stato implementato mediante un algoritmo Python eseguito tramite l'ambiente *Anaconda Navigator*, utilizzando *Spyder* come ambiente di sviluppo. L'obiettivo principale era ridurre ciascun dataset da un numero iniziale variabile tra 500 e 1500 punti a circa 50 valori significativi.

Per ottenere tale riduzione è stato impiegato l'algoritmo di Douglas-Peucker [5], una tecnica matematica ampiamente utilizzata nell'elaborazione di curve e segnali per la semplificazione di polilinee. L'algoritmo funziona individuando il punto più distante dal segmento che unisce gli estremi della curva: se tale distanza è inferiore a una soglia prestabilita (generalmente indicata con il parametro ϵ), l'intera curva viene approssimata dal segmento stesso. In caso contrario, l'algoritmo divide la curva in corrispondenza di quel punto e si richiama ricorsivamente sulle due metà. Applicato al Battery Sizing Tool, questo metodo ha permesso di scartare i punti geometricamente ridondanti, preservando l'andamento reale della curva di scarica della batteria.

Il workflow completo definito per la riduzione dei dati è articolato come segue:

1. acquisizione dei dati originali dal datasheet del fornitore e salvataggio in un file Excel;

2. esecuzione dell'algoritmo Python, che elabora i dati ed estrae i punti più significativi mediante il metodo di Douglas–Peucker;

3. generazione di un nuovo foglio Excel contenente esclusivamente i valori ridotti;

4. esportazione manuale del foglio in formato JSON;

5. integrazione dei valori JSON nel database interno del *Battery Sizing Tool*.

Figura 3.6: Esempio di esecuzione dell'algoritmo Python per la riduzione dei punti tramite metodo di Douglas–Peucker.

La Figura 3.6 mostra un esempio del codice Python utilizzato e del grafico risultante, nel quale è possibile osservare la sovrapposizione tra i dati originali e i punti semplificati.

Questa soluzione presenta numerosi vantaggi. La riduzione del dataset permette di limitare drasticamente il numero di valori caricati nel sistema, migliorando le performance del tool sia durante il filtraggio sia in fase di calcolo. Inoltre, la modularità del processo consente una manutenzione più agevole: eventuali aggiornamenti o correzioni dei dati richiedono modifiche esclusivamente al codice Python, senza intervenire sulla logica JavaScript.

Sono tuttavia presenti anche alcuni svantaggi. Il principale riguarda la necessità di eseguire manualmente numerosi passaggi: conversione dei dati, esportazione, caricamento dei file JSON e verifica dei risultati. Questo flusso operativo introduce un rischio di errore umano e richiede tempi aggiuntivi, soprattutto in caso di aggiornamento dei datasheet o introduzione di nuovi modelli di batteria.

Confronto numerico tra dataset originali e dataset ridotti Per rendere più evidente l'impatto del processo di riduzione dei dati, è utile riportare un confronto quantitativo tra il numero di punti iniziali e quello finale dopo l'applicazione dell'algoritmo di Douglas–Peucker. In molti casi, i datasheet forniti dai produttori presentano tra i 500 e i 1500 valori sperimentali; grazie al processo di semplificazione, tali dataset vengono ridotti mediamente a circa 40–60 punti

significativi. Ciò equivale, in termini percentuali, ad una riduzione dell'ordine dell'80%–95% del numero di punti complessivi, senza compromettere la qualità della curva di scarica rappresentata.

Impatto sulle prestazioni L'effetto della riduzione è evidente anche dal punto di vista computazionale. Diminuendo drasticamente la quantità di dati da processare:

- si riducono i tempi di filtraggio e interpolazione nel *Battery Sizing Tool*;
- diminuisce il tempo di caricamento della pagina e del motore JavaScript;
- si riduce il rischio di sforare i limiti di memoria del software di vendita;
- si migliora la reattività complessiva dell'interfaccia utente durante il calcolo delle soluzioni.

In particolare, la riduzione dei punti rende più veloce l'estrazione delle autonomie richieste e la successiva esecuzione dell'interpolazione lineare, che da operazioni complesse su centinaia di righe si riducono a operazioni su poche decine di valori.

Considerazioni finali e prospettive future Sebbene il workflow attuale rappresenti un compromesso efficace, sono possibili ulteriori miglioramenti futuri. Un'evoluzione naturale sarebbe l'automatizzazione completa del processo di conversione dei dati, eliminando gli step manuali necessari per passare da Excel a JSON. Un ulteriore passo avanti potrebbe consistere nell'esposizione di un microservizio esterno, dedicato esclusivamente all'elaborazione dei datasheet tramite Python, che permetterebbe di aggiornare automaticamente i valori ridotti senza intervento umano. Tale prospettiva resta subordinata a possibili evoluzioni del software di vendita, il quale attualmente non consente l'integrazione diretta di componenti esterni.

3.2.5 Gestione della navigazione e concentrazione delle funzionalità in un'unica schermata

Un ulteriore vincolo rilevante affrontato nello sviluppo del *Battery Sizing Tool* riguarda le limitazioni imposte dal software di vendita in termini di navigazione e creazione di pagine aggiuntive. L'ambiente di sviluppo consente esclusivamente l'integrazione di file JavaScript, limitando fortemente la possibilità di introdurre

nuove schermate, sezioni HTML dedicate o strutture di navigazione avanzate basate su HTML e CSS.

In particolare, il DOM dell'applicativo risulta rigidamente predefinito e non modificabile: qualsiasi contenuto aggiuntivo deve essere necessariamente inserito all'interno delle strutture già esistenti, senza la possibilità di creare nuove pagine o componenti indipendenti. Questo vincolo ha reso complessa l'implementazione di soluzioni orientate alla separazione logica delle funzionalità, tipica delle applicazioni web moderne.

Alla luce di tali limitazioni, è stata adottata una strategia di concentrazione delle principali funzionalità del *Battery Sizing Tool* all'interno di un'unica schermata operativa, seguendo il paradigma delle *Single Page Applications* (SPA) [7]. All'interno di questa vista vengono presentate sia le soluzioni di batterie compatibili con i parametri selezionati, sia le informazioni necessarie per la selezione finale della configurazione più idonea. Questa scelta progettuale, sebbene comporti la generazione di pagine potenzialmente molto lunghe in presenza di numerose soluzioni compatibili, consente di evitare una navigazione frammentata e riduce il rischio di perdita di contesto da parte dell'utente.

Per mantenere un adeguato livello di leggibilità e usabilità, è stata preservata integralmente la struttura delle tabelle già presenti nel software di vendita, evitando l'introduzione di layout personalizzati che avrebbero potuto generare problemi di impaginazione, incompatibilità grafiche o incoerenze stilistiche. Tale approccio ha permesso di contenere la complessità visiva della schermata e di garantire una coerenza grafica con il resto dell'applicativo.

A causa dell'impossibilità di creare pagine dedicate di approfondimento, alcune informazioni di supporto sono state rese disponibili attraverso soluzioni alternative. In particolare, una descrizione sintetica dei principali passaggi di calcolo e delle logiche adottate dal tool è stata inserita in forma testuale all'interno della console di sviluppo del browser, accessibile tramite gli strumenti di ispezione della pagina (tasto `F12`). Sebbene questa soluzione non sia ottimale dal punto di vista dell'usabilità, essa rappresenta l'unica modalità compatibile con i vincoli del sistema per fornire informazioni tecniche di dettaglio senza alterare la struttura dell'interfaccia.

Non è stato inoltre possibile integrare direttamente all'interno del software un manuale utente o un manuale tecnico per sviluppatori, a causa dell'impossibilità di aggiungere nuove sezioni descrittive o pagine informative. Tale esigenza è stata quindi soddisfatta attraverso la creazione di un percorso formativo dedicato sotto

forma di webinar, pubblicato nel portale aziendale dei corsi di formazione. In questo contesto, il funzionamento del *Battery Sizing Tool* viene illustrato in modo approfondito dal punto di vista dell'utente finale, con esempi pratici e sessioni di domande e risposte.

Questo approccio consente di mantenere una documentazione sempre accessibile e facilmente aggiornabile. In occasione di modifiche sostanziali o aggiornamenti rilevanti del tool, vengono inoltre organizzate sessioni formative dedicate per informare gli utenti sulle nuove funzionalità e sulle variazioni introdotte. Sebbene tale soluzione non sostituisca una documentazione direttamente integrata nel software, essa rappresenta un compromesso efficace e coerente con i vincoli tecnologici imposti dall'ambiente legacy.

Considerazioni conclusive sul Capitolo 3

Il presente capitolo ha illustrato in modo dettagliato l'architettura del *Battery Sizing Tool* e le principali soluzioni implementate per superare i numerosi vincoli imposti dall'ambiente di sviluppo e dal software di vendita in cui il tool è integrato. Attraverso l'analisi delle scelte progettuali e implementative, è stato possibile evidenziare come ciascun vincolo abbia influito in maniera significativa sia sulla struttura del codice sia sull'esperienza utente finale.

In particolare, l'assenza di un database esterno, l'impossibilità di utilizzare librerie e API aggiuntive, la rigidità del DOM, i limiti nella navigazione e l'impossibilità di integrare direttamente algoritmi esterni hanno reso necessario adottare soluzioni non convenzionali. Tali soluzioni, pur risultando più complesse rispetto a quelle normalmente adottate in contesti meno restrittivi, hanno consentito di raggiungere comunque gli obiettivi funzionali prefissati, garantendo correttezza dei risultati, integrazione con il sistema di vendita e continuità operativa con i processi aziendali esistenti.

Il capitolo ha inoltre messo in evidenza come molte delle scelte effettuate siano il risultato di compromessi tecnici: da un lato l'esigenza di mantenere prestazioni accettabili e una buona usabilità, dall'altro la necessità di rispettare vincoli strutturali non aggirabili. In questo contesto, il *Battery Sizing Tool* rappresenta un esempio concreto di come un approccio progettuale attento e consapevole possa trasformare limiti tecnologici in soluzioni operative efficaci, seppur non ottimali dal punto di vista teorico.

Queste considerazioni costituiscono il punto di partenza per il capitolo successivo, nel quale verrà analizzato come le stesse esigenze avrebbero potuto essere affrontate in un contesto privo di tali vincoli. Il confronto tra l'approccio vincolato descritto in questo capitolo e un'ipotetica soluzione libera da restrizioni permetterà di evidenziare in modo ancora più chiaro l'impatto dei limiti tecnologici sulle scelte progettuali, sulle prestazioni del sistema e sulla sua evolvibilità futura.

Capitolo 4

Confronto con soluzioni prive di vincoli

Nei Capitoli precedenti sono stati analizzati, rispettivamente, i vincoli tecnologici imposti dal software di vendita e le soluzioni progettuali adottate per realizzare il *Battery Sizing Tool* all'interno di tali limitazioni. Il presente capitolo introduce un'analisi comparativa, finalizzata a valutare come le medesime esigenze funzionali e progettuali avrebbero potuto essere soddisfatte in un contesto privo di restrizioni tecnologiche e strutturali.

L'obiettivo di questo confronto non è quello di individuare carenze nell'implementazione realizzata, bensì di evidenziare le differenze tra un approccio fortemente vincolato, necessariamente più complesso e indiretto, e un'architettura teoricamente ottimale, in grado di garantire maggiore flessibilità, semplicità manutentiva e migliori prestazioni complessive.

Attraverso questa analisi sarà possibile mettere in luce il valore delle soluzioni adottate nel Capitolo 3, mostrando come esse rappresentino un compromesso progettuale consapevole, dettato dal contesto applicativo reale, e non una limitazione concettuale o tecnica del progetto.

4.1 Obiettivo del confronto e contesto di riferimento

I Capitoli 2 e 3 hanno descritto in modo dettagliato i vincoli tecnici, architetturali e operativi che hanno caratterizzato lo sviluppo del *Battery Sizing Tool*, nonché le soluzioni adottate per superare tali limitazioni all'interno del software

di vendita. Il presente capitolo ha l'obiettivo di analizzare lo stesso problema progettuale da una prospettiva alternativa, ipotizzando un contesto di sviluppo privo dei vincoli descritti in precedenza.

Lo scopo del confronto non è quello di valutare la bontà assoluta delle soluzioni implementate, bensì di evidenziare come le scelte progettuali siano state fortemente influenzate dall'ambiente di esecuzione e dalle restrizioni imposte dal sistema ospitante. In particolare, verrà messo in evidenza come, in un contesto teoricamente ideale, molte delle complessità affrontate nel Capitolo 3 potrebbero essere risolte in maniera più diretta, modulare ed efficiente.

Il confronto verrà condotto assumendo un ambiente di sviluppo standard per applicazioni web o software stand-alone, caratterizzato da:

- pieno controllo del DOM e della struttura dell'interfaccia utente;
- possibilità di utilizzare database esterni per la gestione dei dati;
- integrazione diretta di linguaggi e librerie eterogenee (ad esempio Python e JavaScript);
- supporto alla navigazione multi-pagina e a interfacce grafiche avanzate;
- assenza di vincoli stringenti sul caricamento e sull'elaborazione dei dati.

All'interno di tale scenario, il *Battery Sizing Tool* potrebbe essere progettato seguendo un'architettura più flessibile e scalabile, con una netta separazione tra logica di business, gestione dei dati e presentazione. Questo capitolo intende quindi fornire una visione comparativa tra:

1. l'approccio realmente adottato, vincolato dal contesto del software di vendita;
2. un approccio teoricamente ottimale, sviluppato in assenza di tali restrizioni.

L'analisi permetterà di evidenziare i benefici potenziali in termini di prestazioni, manutenibilità, usabilità e semplicità implementativa, chiarendo al contempo come le soluzioni adottate nel progetto reale rappresentino un compromesso necessario e consapevole, piuttosto che una limitazione progettuale.

4.2 Gestione dei dati in un contesto privo di vincoli tecnologici

Nel presente paragrafo viene analizzato come le principali scelte progettuali adottate per il *Battery Sizing Tool* sarebbero potute evolvere in un contesto privo dei vincoli tecnologici descritti nei capitoli precedenti. In particolare, l'attenzione è rivolta alla gestione dei dati, evidenziando le differenze tra l'approccio vincolato realmente adottato e una soluzione teoricamente ottimale.

4.2.1 Utilizzo di un database esterno per la gestione dei dati

In assenza del vincolo legato all'impossibilità di utilizzare un database esterno, la soluzione architetturalmente più appropriata sarebbe stata l'adozione di un database relazionale dedicato alla gestione dei dati del *Battery Sizing Tool*. All'interno di tale database sarebbero stati memorizzati in modo strutturato tutti i valori necessari al corretto funzionamento del sistema, inclusi i punti completi dei datasheet delle batterie, i parametri di efficienza delle macchine e gli attributi utilizzati per l'indicizzazione e il confronto tra le diverse soluzioni disponibili.

L'impiego di un database relazionale avrebbe consentito di accedere ai dati in maniera diretta ed efficiente tramite query mirate, eliminando la necessità di scorrere sequenzialmente grandi array di oggetti all'interno del codice JavaScript. Questo approccio avrebbe permesso di recuperare esclusivamente le informazioni rilevanti per ogni fase di calcolo, riducendo sensibilmente il carico computazionale e migliorando i tempi di esecuzione complessivi dell'algoritmo.

Dal punto di vista implementativo, la presenza di un database avrebbe inoltre semplificato il codice sorgente, rendendo superflua la gestione manuale di strutture dati complesse in memoria. L'algoritmo di selezione delle batterie avrebbe potuto operare direttamente su insiemi di dati già filtrati e indicizzati, risultando più snello, leggibile e meno oneroso in termini di manutenzione.

Un ulteriore vantaggio significativo sarebbe stato rappresentato da una maggiore affidabilità dei dati. Non sarebbe stato necessario applicare algoritmi di riduzione preventiva dei dataset, come nel caso della soluzione vincolata, poiché il database avrebbe potuto gestire senza difficoltà grandi volumi di informazioni. Questo avrebbe garantito l'utilizzo dell'intero insieme di dati forniti dai produtto-

ri, riducendo il rischio di approssimazioni e aumentando la precisione dei risultati finali.

Infine, l'aggiornamento e l'estensione del sistema sarebbero risultati notevolmente più semplici. L'introduzione di nuovi modelli di batterie o la modifica di parametri esistenti avrebbe richiesto unicamente l'aggiornamento delle tabelle del database, senza la necessità di intervenire direttamente sul codice del tool. Tale approccio avrebbe migliorato la scalabilità del sistema e facilitato la gestione delle evoluzioni future del *Battery Sizing Tool*.

4.3 Vincoli software e stack tecnologico

Un secondo ambito di confronto riguarda lo stack tecnologico utilizzato. Come evidenziato nell'analisi dei vincoli, l'utilizzo forzato di "JavaScript puro" (Vanilla JavaScript) all'interno di un ambiente legacy ha comportato notevoli limitazioni espressive e funzionali.

4.3.1 Adozione di Framework moderni e librerie grafiche

In uno scenario privo di vincoli, la scelta ottimale sarebbe ricaduta sull'integrazione di framework reattivi moderni (come React[1], Vue.js[2] o Angular[3]). L'adozione di un'architettura a componenti avrebbe garantito una maggiore coerenza strutturale, facilitando la gestione dello stato dell'applicazione e la manutenibilità del codice nel tempo.

In particolare, un ambiente di sviluppo moderno avrebbe permesso di:

- **Modellare il design grafico** con maggiore libertà, superando la rigidità del template aziendale e offrendo un'esperienza visiva più curata e in linea con gli standard web attuali;
- **Integrare librerie di visualizzazione dati** (come Chart.js[4] o D3.js[5]) per generare grafici interattivi. Al contrario della soluzione attuale, puramente tabellare, il tool avrebbe potuto mostrare curve di scarica dinamiche e istogrammi comparativi, fornendo un supporto visivo immediato alle decisioni tecniche;

[1] https://react.dev
[2] https://vuejs.org
[3] https://angular.io
[4] https://www.chartjs.org
[5] https://d3js.org

- **Gestire la documentazione integrata**, creando sezioni dedicate ("Help" o "Wiki") accessibili direttamente dall'applicazione, contenenti il manuale utente e le specifiche di sviluppo, migliorando così l'autonomia degli utenti e la trasferibilità del progetto.

4.4 Evoluzione dell'interfaccia e della navigazione

Un limite critico del *Battery Sizing Tool* attuale risiede nell'architettura "Single Page" obbligata [7], che costringe a condensare input, output e controlli in un'unica schermata a scorrimento verticale.

4.4.1 Navigazione strutturata e Dashboard personalizzate

In un contesto architetturale libero, l'interfaccia sarebbe stata riprogettata secondo una logica modulare e multi-pagina, migliorando drasticamente l'usabilità (User Experience). La soluzione ottimale prevedeva:

- **Suddivisione in sotto-pagine funzionali:** invece di un unico flusso continuo, le informazioni accessorie (spiegazione del tool, consultazione del database grezzo, manualistica) sarebbero state allocate in pagine dedicate, mantenendo la schermata di calcolo pulita ed essenziale.
- **Dashboard personalizzate per tipologia di utente:** implementazione di un sistema di profili utente, capace di mostrare viste differenti per Ingegneri (dettagli tecnici approfonditi) e Commerciali (focus su prezzi e codici).
- **Gestione delle preferenze (User Preferences):** possibilità per l'utente di salvare configurazioni "preferite" o template di progetto ricorrenti, rendendo le operazioni ripetitive accessibili con un singolo click.
- **Filtri avanzati:** implementazione di meccanismi di filtraggio automatizzati e "a faccette" (es. selezione multipla per marca di batteria), gestiti dinamicamente dal framework grafico senza ricaricamenti pesanti della pagina.

In sintesi, l'approccio non vincolato avrebbe trasformato il tool da semplice "calcolatore" a vera e propria applicazione gestionale, centrata sulle esigenze specifiche del singolo utente.

4.5 Integrazione computazionale e algoritmi

L'ultimo, ma fondamentale, punto di confronto riguarda la gestione della logica di calcolo e l'integrazione con il linguaggio Python. Nel progetto attuale, l'uso di Python è rimasto confinato a un'esecuzione esterna e manuale, fungendo da pre-processore scollegato dal tool principale.

4.5.1 Backend integrato e automazione dei processi

In un'architettura web standard (Client-Server), la soluzione ideale avrebbe previsto l'implementazione di un backend dedicato (ad esempio sviluppato in Python con framework come Django o FastAPI) capace di dialogare nativamente con il frontend.

Questa architettura avrebbe permesso di:

1. **Centralizzare la logica algoritmica:** gestire tutto il flusso di calcolo in un unico ambiente interoperabile, eliminando la necessità di passaggi manuali, conversioni di file o "vie traverse" per importare i dati.

2. **Esecuzione real-time dell'algoritmo di Douglas-Peucker:** invece di pre-calcolare staticamente i dati ridotti, il server avrebbe potuto attingere al database completo (es. 1000 punti per curva) ed eseguire la riduzione on-demand solo sui dati richiesti. Ciò avrebbe garantito la massima precisione senza appesantire la memoria del browser cliente.

3. **Integrità e sicurezza dei dati:** evitando la manipolazione manuale di file Excel e JSON, si sarebbe azzerato il rischio di errore umano (copia-incolla errati, perdita di cifre significative), garantendo che il risultato finale fosse sempre matematicamente coerente con la fonte dati originale.

La transizione da un workflow frammentato (Excel → Python locale → JSON → JavaScript) a una pipeline integrata (Database → Backend Python → API → Frontend) rappresenta il salto di qualità più significativo che l'assenza di vincoli avrebbe reso possibile, portando il livello di automazione ed efficienza del sistema al suo massimo potenziale.

Capitolo 5

Conclusioni

Il percorso descritto in questo elaborato ha attraversato l'intero ciclo di vita del *Battery Sizing Tool*, dalla definizione delle esigenze aziendali fino all'analisi critica delle scelte implementative.

In sede conclusiva, è opportuno ribadire che l'oggetto primario di questa tesi non è stato la realizzazione di un prodotto software, bensì l'indagine ingegneristica sulle barriere tecnologiche che ne hanno condizionato lo sviluppo. Il tool realizzato funge quindi da caso di studio per dimostrare come, in contesti aziendali consolidati ("legacy"), la capacità di adattamento e l'ingegnosità nell'aggirare i vincoli contino tanto quanto la pura competenza di programmazione.

Ripercorrendo le tappe dello sviluppo, appare evidente come la sfida principale non sia stata la complessità matematica dell'algoritmo in sé, quanto piuttosto la necessità di calare tale logica all'interno di un ecosistema tecnologico fortemente vincolato. L'infrastruttura legacy del software di vendita, con la sua chiusura verso database esterni, l'assenza di un backend dedicato e la rigidità dell'interfaccia, ha rappresentato un banco di prova significativo per l'ingegneria del software.

Sintesi dei risultati raggiunti

Nonostante le restrizioni, il risultato finale soddisfa pienamente i requisiti operativi. Il tool è oggi uno strumento funzionante che ha permesso di:

- **Centralizzare il know-how:** le regole di compatibilità e i dati tecnici, prima dispersi in fogli Excel e nella "memoria storica" degli ingegneri, sono ora codificati in un unico motore software univoco.

- **Democratizzare il processo tecnico:** rendendo il dimensionamento accessibile anche alla forza vendita tramite un'interfaccia guidata, si è ridotto il carico di lavoro ripetitivo sugli Application Engineer, permettendo loro di concentrarsi su progetti a maggior valore aggiunto.
- **Garantire la correttezza del dato:** l'eliminazione dei passaggi manuali di trascrizione e l'integrazione diretta con la distinta base (BOM) hanno abbattuto drasticamente il rischio di errore umano in fase di offerta.

Il valore del compromesso ingegneristico

L'aspetto forse più interessante emerso da questo lavoro di tesi è la rivalutazione del concetto di "vincolo". Come evidenziato nel confronto con le soluzioni ideali (Capitolo 4), un approccio tecnologicamente libero avrebbe certamente portato a un'architettura più moderna, modulare ed elegante. Tuttavia, l'ingegneria nel mondo reale non è quasi mai l'applicazione di soluzioni da manuale in un ambiente asettico, bensì la capacità di trovare la soluzione più efficace date le risorse disponibili.

Le strategie adottate, dal "micro-database" JSON interno, all'algoritmo di riduzione dati in Python, fino alla manipolazione diretta del DOM, rappresentano risposte creative a problemi concreti. Sebbene non costituiscano lo "stato dell'arte" dello sviluppo web moderno, esse costituiscono lo "stato dell'arte" di ciò che era possibile realizzare in quel preciso contesto aziendale. Il *Battery Sizing Tool* dimostra che è possibile innovare anche all'interno di sistemi legacy, a patto di accettare la complessità implementativa come prezzo per l'integrazione operativa.

Sviluppi futuri

Guardando avanti, il tool non deve essere considerato un punto di arrivo immutabile. L'analisi svolta ha gettato le basi per future evoluzioni, qualora i vincoli della piattaforma ospitante dovessero allentarsi. L'eventuale apertura verso API esterne o l'adozione di un nuovo sistema CPQ più moderno potrebbero sbloccare le potenzialità descritte nel quarto capitolo: aggiornamenti dati in tempo reale, interfacce reattive e calcolo distribuito.

In conclusione, questo progetto lascia in eredità all'azienda non solo uno strumento software immediatamente utile, ma anche una metodologia di approccio ai

problemi complessi, dimostrando come la competenza tecnica possa trasformare limiti strutturali in soluzioni di business efficienti e scalabili.

Bibliografia

[1] IEEE recommended practice for sizing lead-acid batteries for stationary applications, 2020. Disponibile su IEEE Xplore.

[2] ECMAscript 2023 language specification, 2023. Standard ufficiale per il linguaggio JavaScript.

[3] DealHub. Legacy vs. next-gen CPQ: Upgrade or fall behind, 2025. Accesso: 2026-02-02.

[4] MDN Web Docs. Introduction to the DOM, 2024. Accesso: 2026-02-02.

[5] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[6] IBM. What is legacy application modernization?, 2024. Accesso: 2026-02-02.

[7] Michael Mikowski and Josh Powell. *Single Page Web Applications: JavaScript End-to-End*. Manning Publications, 2013.

[8] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2015. Capitolo 9: Software Evolution e Legacy Systems.

[9] Vertiv Group Corp. Vertiv - critical infrastructure technologies and continuity services, 2024. Sito ufficiale aziendale. Accesso: 2026-02-02.

Ringraziamenti

Desidero innanzitutto ringraziare il mio Relatore, Prof. Angelo Di Iorio, per la disponibilità e il supporto dimostrati durante la stesura di questo elaborato.

Un ringraziamento speciale va alla mia Correlatrice, Dott.ssa Letizia Ghirardello, e a tutto il team di Vertiv. Grazie per avermi accolto in un ambiente di lavoro stimolante e per avermi dato l'opportunità di confrontarmi con sfide reali, permettendomi di crescere professionalmente e umanamente.

A papà e mamma, il mio sole e la mia luna.
Grazie per avermi educato e cresciuto nel miglior modo possibile. Grazie per avermi aiutato a superare tutti gli ostacoli incontrati lungo il mio percorso. Grazie per non avermi mai fatto mancare nulla e per avermi sempre sostenuto. Grazie per avermi sempre ascoltato: qualunque fosse il problema o la difficoltà, sapevo di poter contare su due pilastri pronti a guidarmi verso la scelta migliore. Mi sento estremamente fortunato ad avere dei genitori come voi, che mi hanno insegnato ad apprezzare ogni aspetto di questa vita, soprattutto i piccoli gesti. Voglio anche chiedervi scusa per i miei errori e per le volte in cui mi sono comportato male nei vostri confronti, perché non lo meritate. Gran parte del merito di questo traguardo è vostro; poche righe di ringraziamento alla fine di una tesi non basteranno mai a ripagare tutto l'amore e i sacrifici che avete fatto per me, ma spero di avervi reso orgogliosi dell'uomo che avete cresciuto e che sono diventato. VI VOGLIO BENE

Ai miei nonni: Nonna Dina, Nonno Tonno, Nonna Gina,
Mi avete aiutato a diventare la versione migliore di me stesso, mi avete amato incondizionatamente e vi siete sempre presi cura di me in ogni minimo dettaglio. Mi avete guidato nelle scelte più importanti con la vostra saggezza, a volte usando

poche parole, altre raccontandomi le preziose esperienze vissute in prima persona. Un semplice ringraziamento non potrà mai ripagare tutto ciò che avete fatto per me in questi anni, ma spero siate orgogliosi della persona che sono diventato, perché è anche merito vostro.

Un pensiero speciale lo riservo a te, **Nonno Bruno**. Mi hai visto nascere e vivere i primi anni di vita, poi Dio ti ha portato via da me perché lassù c'era bisogno di un'anima pura come la tua. Non ho avuto il tempo di conoscerti a fondo, ma so che da quel momento sei sempre stato al mio fianco, come il mio angelo custode. Poiché un figlio prende spesso ispirazione dal proprio padre, mi piace pensare che il mio papà ti somigli molto: per questo ti ringrazio, per aver cresciuto insieme a nonna un uomo e un papà perfetto come il mio.

A Jenny, colei che mi ha insegnato cosa vuol dire amare incondizionatamente qualcuno senza il bisogno di usare le parole, ma semplicemente attraverso i gesti. Quando dicono che il cane è il migliore amico dell'uomo, io ci credo fermamente. Anzi, per me sei stata molto più di una migliore amica: sei sempre stata come una sorellina. Ti sarò per sempre grato per tutto ciò che mi hai insegnato; mi hai donato un amore immenso e in cambio non hai mai chiesto nulla, se non un po' di coccole e la pappa. Ogni volta che tornavo a casa ti trovavo lì, pronta a saltarmi addosso scodinzolando, felice di dimostrarmi tutto il tuo affetto anche se ero uscito solo per pochi minuti. Purtroppo oggi non potrai essere qui a festeggiare con me, ma ti porterò per sempre nel mio cuore. Sei la cosa più bella che potesse capitarmi durante l'infanzia e, crescendo insieme, sei diventata una presenza sempre più preziosa, accompagnandomi in silenzio in tutte le nostre avventure. Lassù so di avere un angelo a quattro zampe che mi proteggerà per sempre. Un giorno ci ritroveremo e torneremo a correre insieme, proprio come quando eravamo piccolini.
Grazie di tutto Gigì, ti amo... dal tuo Dado

A Beppe, Samanta e Anita, la mia seconda famiglia. Mi avete fatto capire che la famiglia non è solo quella di sangue; mi avete accolto in casa vostra fin dal primo giorno come se fossi vostro figlio. Grazie per tutto quello che avete fatto per me in questi anni. Spero di essermi sempre comportato in maniera corretta

e di aver ripagato il vostro affetto nel miglior modo possibile. Nonostante non ci vedessimo tutti i giorni, siete stati sempre presenti nel raggiungimento di questo mio obiettivo, quindi questo traguardo è dedicato anche a voi.

A Ricky, il mio fratello acquisito. Sono passati tanti anni da quando giocavamo alla Nintendo in camera tua, e ora siamo qui insieme a festeggiare un mio traguardo. O meglio, un nostro traguardo, perché anche tu fai parte di questo percorso. Ci siamo divertiti, ci siamo sostenuti a vicenda e siamo cresciuti insieme fino a oggi: per questo un ringraziamento speciale va anche a te.

A Jesus. È difficile spiegare a chi non lo ha vissuto come un piccolo campo da calcio e un pallone possano trasformare un compagno di squadra in un compagno di vita. Beh, da quel giorno a Ozzano sono passati tanti anni e ne abbiamo condivise tante insieme (calcio, scuola, uscite e vacanze), ma siamo sempre rimasti uniti, e questo è l'importante. Nonostante ci si veda meno, ogni volta non cambia nulla, perché in fondo siamo sempre quei ragazzini che correvano spensierati dietro a un pallone.

A Tom Bors. Anche in questo caso il campo da calcio ha fatto da protagonista, trasformando due giocatori in due veri amici: il 9 e l'11. Ci siamo capiti e trovati fin da subito e, nonostante tutto, siamo ancora qui. Siamo partiti dallo spogliatoio di Poggio e, dopo tante (ma tante) panchine insieme, siamo rimasti profondamente legati. Io sarò sempre qui per te, anche perché in futuro Thiago avrà bisogno di uno zio con i piedi buoni che gli spieghi come calciare in porta.

Alla Fiore, la mia prima maestra di scuola e di vita. Grazie per avermi sopportato e aiutato quando ero piccolo, e per avermi guidato nelle ore scolastiche in maniera impeccabile.
Un enorme grazie va anche a **mamma Tina**, che di sicuro, da lassù, sarà felice di vedere che il suo piccolo Thomas ha raggiunto questo traguardo.

A tutti i miei amici. Grazie per essere sempre stati al mio fianco nei momenti di difficoltà e per avermi tirato su il morale anche quando era a terra. Chi

conosco fin da piccolo e chi ho incontrato più tardi, ci siamo sempre sostenuti a vicenda e accompagnati nei nostri rispettivi percorsi. Penso che questo sia il vero significato dell'amicizia. Grazie.

Durante l'ingresso in campo, l'ultima posizione viene lasciata, di norma, al giocatore più importante, colui che chiude la fila, come a voler dire: "Ci sono, ora si può iniziare".
Vedi, **Sophia**, ti ho lasciata per ultima proprio per questo motivo: tu che sei l'amore della mia vita meriti la parte finale, per concludere in bellezza.
Sono passati cinque anni da quando ci siamo conosciuti, cinque anni ricchi di avventure e traguardi condivisi. Insieme abbiamo attraversato l'ultima fase del Covid, la mia maturità (in cui sei stata letteralmente la mia salvezza, aiutandomi a studiare quando mi ero ridotto all'ultimo minuto) e poi la tua, tra ansie, scleri e nervoso che abbiamo affrontato mano nella mano. Abbiamo condiviso l'emozione della mia immatricolazione all'università e, poco dopo, abbiamo festeggiato anche la tua.
Sono stati cinque anni di vita insieme in cui ci siamo amati profondamente. In realtà, fin dai primissimi giorni avevo già capito che il nostro sarebbe stato un amore allo stato puro, degno del più bel film romantico. È stato naturale e bellissimo entrare subito a far parte l'uno della famiglia dell'altra, sentendoci a casa fin dal primo istante. E guardandoci oggi, sono immensamente orgoglioso di noi e di come sta procedendo la nostra vita insieme.
Voglio ringraziarti per tutti i meravigliosi viaggi che abbiamo fatto finora e per tutti quelli che ancora riempiranno i nostri anni futuri. Ma soprattutto, voglio dirti grazie per il sostegno incondizionato che mi hai dato. Il tuo amore, la tua felicità e soprattutto il tuo sorriso mi hanno sempre sostenuto, illuminando anche le mie giornate più pesanti. Mi hai dato la forza di resistere sui libri, mi hai spronato a dare sempre il meglio di me, a non mollare nei momenti di sconforto e a portare a termine tutte le mie attività fino a raggiungere i miei obiettivi, compreso questo che celebriamo insieme oggi, perché gran parte del merito è anche tuo.
So che non bastano queste poche righe per spiegare quanto tu sia essenziale per me, ma spero di dimostrartelo ogni giorno, anche nelle giornate più negative, con i piccoli gesti e con tutto l'affetto e l'amore che ti trasmetto.
A volte penso che ci siamo trovati come attraverso un Ponte di Einstein-Rosen,

un tunnel spazio-temporale capace di unire due universi distanti in un batter d'occhio. Credo davvero che sia stato il destino a far incrociare le nostre strade in quell'istante perfetto, e spero con tutto il cuore che ci tenga uniti per sempre, affinché la nostra vita continui a essere, riprendendo la nostra frase preferita, proprio *"Come nelle favole"*.

Infine, un ringraziamento va anche a chi non ha creduto in me o ha scelto di allontanarsi: mi avete dato la determinazione e la spinta necessaria per arrivare fin qui e dimostrare il mio vero valore. I vostri dubbi sono stati benzina per il mio traguardo.

Con affetto,

Thomas