Error_418

GitHub/Error-418-SWE error418swe@gmail.com

Specifica Tecnica

Informazioni

Versione	1.8.0
Uso	Esterno
Stato	Approvato
Responsabile	Zaccone Rosario
Redattori	Nardo Silvio Todesco Mattia
Verificatore	Banzato Alessio
Destinatari	Gruppo Error_418 Vardanega Tullio Cardin Riccardo

Registro delle modifiche

Ver.	Data	PR	Titolo	Redattore	Verificatore
1.8.0	04-04-2024	428	DOC-682 Riorganizzare paragrafo tecnologie	Carraro Riccardo	Banzato Alessio
1.7.0	02-04-2024	406	DOC-654 Estendere sezione Tecnologie	Carraro Riccardo	Gardin Giovanni
1.6.0	27-03-2024	367	DOC-651 Redigere sezione Classi	Nardo Silvio	Gardin Giovanni
1.5.0	27-03-2024	367	DOC-650 Redigere sezione Design pattern utilizzati	Nardo Silvio	Gardin Giovanni
1.4.0	27-03-2024	367	DOC-649 Redigere sezione Struttura	Nardo Silvio	Gardin Giovanni
1.7.0	27-03-2024	367	DOC-652 Redigere sezione Componenti	Nardo Silvio	Gardin Giovanni
1.3.0	21-03-2024	385	DOC-595 Redigere sezione Requisiti soddisfatti	Todesco Mattia	Banzato Alessio
1.2.0	18-03-2024	376	DOC-606 Aggiungere sezione Requisiti	Todesco Mattia	Banzato Alessio
1.1.1	16-03-2024	370	DOC-598 Modifiche a sezione Database	Todesco Mattia	Banzato Alessio
1.1.0	11-03-2024	360	DOC-563 Redigere sezione Tecnologie	Todesco Mattia	Banzato Alessio

Indice dei contenuti

1 Introduzione 1
1.1 Scopo del documento 1
1.2 Approccio alla redazione 1
1.3 Scopo del prodotto 1
1.4 Glossario 1
1.5 Riferimenti 1
1.5.1 Riferimenti a documentazione interna 2
1.5.2 Riferimenti normativi 2
1.5.3 Riferimenti informativi 2
1.5.4 Riferimenti a documentazione tecnica 2
2 Requisiti 4
2.1 Requisiti di sistema minimi 4
2.2 Requisiti hardware 4
2.3 Browser 4
3 Tecnologie 5
3.1 Introduzione 5
3.2 Tecnologie implementative 5
3.2.1 Next.js 5
3.2.2 React 5
3.2.3 Node.js 5
3.2.4 Tailwind CSS 6
3.2.5 Shadcn/ui 6
3.2.6 Three.js 6
3.2.7 @react-three/fiber 6
3.2.8 @react-three/drei 7
3.3 Tecnologie per la validazione dei dati 7
3.3.1 Zod 7
3.4 Tecnologie per la persistenza dei dati 7
3.4.1 PostgreSQL 7
3.5 Tecnologie per il testing 7
3.5.1 Jest 7
3.6 Tecnologie per il deployment 8
3.6.1 Docker 8
3.6.2 Docker Compose 8
4 Architettura di sistema 9
4.1 Architettura di implementazione 9
4.1.1 Vantaggi 9
4.1.2 Svantaggi 9
4.2 Persistence layer 10
4.2.1 Next.js Server Actions 10

4.2.2 Server Actions implementate 11
4.2.3 Repository Pattern 13
4.2.4 Data Mapper Pattern 14
4.3 Business layer 15
4.3.1 Bin 15
4.3.2 Zone 17
4.3.3 Product 20
4.3.4 Order 21
4.3.5 SVG 21
4.3.6 Floor 22
4.3.7 FloorStrategy *Strategy Pattern* 23
4.3.8 Search Engine *Factory e Strategy Pattern* 26
4.3.9 Provider Pattern 29
4.4 Presentation layer 32
4.4.1 UI 32
4.4.2 Three.js 40
5 Architettura di deployment 41
5.1 Analisi dell'architettura monolitica 41
5.2 Deployment con Docker 42
5.2.1 Ambiente Docker 42
5.3 Deployment e installazione 43
5.3.1 Download di WMS3 43
5.3.2 Avvio di WMS3 43
5.3.3 Terminare l'esecuzione 44
6 Database 44
6.1 Entità 44
6.2 Relazioni 45
6.2.1 Interrogazione del database 45
7 Requisiti soddisfatti 46
7.1 Requisiti funzionali soddisfatti 46
7.2 Requisiti di qualità soddisfatti 54

Indice delle immagini

Figura 1: Layered architecture. 9
Figura 2: Next.js caching and revalidating 10
Figura 3: Diagramma delle classi del layer di persistenza 15
Figura 4: Diagramma delle classi del layer di business 25
Figura 5: Diagramma delle classi Floor del layer di business 26
Figura 6: Diagramma delle classi Search del layer di business 29
Figura 7: Ambiente Docker 43
Figura 8: Schema ER del Database. 44

Indice delle tabelle

Tabella 1: Requisiti di sistema minimi **4**
Tabella 2: Requisiti hardware **4**
Tabella 3: Browser supportati **4**
Tabella 4: Modalità di download WMS3 ..... **43**
Tabella 5: Requisiti funzionali **46**
Tabella 6: Requisiti di qualità **54**

1 Introduzione

1.1 Scopo del documento

Il presente documento ha come obiettivo la descrizione dettagliata delle scelte progettuali effettuate, al fine di garantire una comprensione chiara e completa del software "WMS3: Warehouse Management 3D", proposto da Sanmarco Informatica S.p.A.

Gli aspetti fondamentali riguardano l'architettura implementativa, analizzando tecnologie e design pattern adottati, e l'architettura di deployment del prodotto.

Mediante il documento si intende fornire le linee guida per lo sviluppo del software, garantendo la coerenza con i requisiti individuati nel documento di Analisi dei Requisiti$_G$ e il loro soddisfacimento.

1.2 Approccio alla redazione

Il presente documento viene redatto in modo incrementale assicurando la coerenza delle informazioni al suo interno con gli sviluppi in corso e le esigenze evolutive del progetto.

1.3 Scopo del prodotto

Il fine ultimo è lo sviluppo di un software, "WMS3: Warehouse Management 3D", che rivoluziona la gestione di un magazzino$_G$, trascendendo la rappresentazione bidimensionale tradizionale a favore di un ambiente tridimensionale più informativo e intuitivo.

"WMS3" si distingue per le sue funzionalità avanzate, tra cui la creazione personalizzata di un magazzino$_G$ e delle sue componenti, arricchita da una visualizzazione tridimensionale che offre una comprensione spaziale ottimale grazie alla possibilità di manipolare la vista.

Il software consente inoltre l'accesso e la visualizzazione delle informazioni relative alla merce$_G$ e alla disposizione degli scaffali, sfruttando un database$_G$ SQL$_G$ per il caricamento di tali dati.

Un altro aspetto fondamentale di "WMS3" è la facilità con cui è possibile emettere richieste di spostamento della merce$_G$ all'interno del magazzino$_G$, rendendo la gestione logistica un processo semplice e intuitivo. Il software integra funzionalità di filtraggio e ricerca delle merci, presentando i risultati in modo grafico per una interpretazione immediata.

Per concludere, "WMS3" supporta la personalizzazione dell'ambiente attraverso l'importazione di planimetrie in formato SVG$_G$, permettendo una configurazione avanzata del layout del magazzino$_G$.

1.4 Glossario

Al fine di agevolare la comprensione del presente documento, viene fornito un glossario che espliciti il significato dei termini di dominio specifici del progetto. I termini di glossario sono evidenziati nel testo mediante l'aggiunta di una "*G*" a pedice degli stessi:

Termine di glossario$_G$

Le definizioni sono disponibili nel documento *Glossario v1.3.0*.

1.5 Riferimenti

1.5.1 Riferimenti a documentazione$_G$ interna

- Documento *Glossario v1.3.0*:
  https://github$_G$.com/Error-418-SWE$_G$/Documenti/blob/main/3%20-%20PB/Glossario_v1.3.0.pdf *(ultimo accesso 25/02/2024)*
- Documento *Analisi dei Requisiti$_G$ v2.0.1*:
  https://github$_G$.com/Error-418-SWE$_G$/Documenti/blob/main/3%20-%20PB/Documentazione$_G$%20esterna/Analisi%20dei%20Requisiti_v2.0.1.pdf *(ultimo accesso 25/02/2024)*

1.5.2 Riferimenti normativi

- Regolamento del progetto didattico$_G$:
  https://www.math.unipd.it/~tullio/IS$_G$-1/2023/Dispense/PD2.pdf *(ultimo accesso 20/03/2024)*
- Capitolato$_G$ "Warehouse Management 3D" (C5) di *Sanmarco Informatica S.p.A.*:
  https://www.math.unipd.it/~tullio/IS$_G$-1/2023/Progetto/C5.pdf *(ultimo accesso 13/02/2024)*

1.5.3 Riferimenti informativi

- Verbali interni;
- Verbali esterni;
- Analisi dei requisiti:
  https://www.math.unipd.it/~tullio/IS$_G$-1/2023/Dispense/T5.pdf *(ultimo accesso 20/03/2024)*
- Analisi e descrizione delle funzionalità, Use Case$_G$ e relativi diagrammi (UML$_G$):
  https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf *(ultimo accesso 20/03/2024)*

1.5.4 Riferimenti a documentazione$_G$ tecnica

- Docker$_G$:
  https://docs.docker$_G$.com/engine/ *(ultimo accesso 28/03/2024)*
- Docker Compose$_G$:
  https://docs.docker$_G$.com/compose/ *(ultimo accesso 28/03/2024)*
- Jest$_G$:
  https://jestjs.io/docs/getting-started *(ultimo accesso 28/03/2024)*
- Next.js$_G$:
  https://nextjs.org/docs *(ultimo accesso 28/03/2024)*
- PostgreSQL$_G$:
  https://www.postgresql$_G$.org/docs/16/index.html *(ultimo accesso 28/03/2024)*
- React$_G$:
  https://react$_G$.dev/reference/react$_G$ *(ultimo accesso 28/03/2024)*
- @react$_G$-three/drei:
  https://github$_G$.com/pmndrs/drei?tab=readme-ov-file#index *(ultimo accesso 28/03/2024)*
- @react$_G$-three/fiber:
  https://docs.pmnd.rs/react$_G$-three-fiber/ *(ultimo accesso 28/03/2024)*

- shadcn/ui:
  https://ui.shadcn.com/docs *(ultimo accesso 28/03/2024)*
- Tailwind CSS:
  https://tailwindcss.com/docs/ *(ultimo accesso 28/03/2024)*
- Three.js$_G$:
  https://threejs.org/docs/ *(ultimo accesso 28/03/2024)*
- Zod:
  https://zod.dev/ *(ultimo accesso 28/03/2024)*

2 Requisiti

Di seguito sono elencati i requisiti minimi necessari per l'esecuzione dell'applicazione, comprese le caratteristiche necessarie per configurare l'ambiente di sviluppo del progetto.

2.1 Requisiti di sistema minimi

Componente	Versione$_G$	Riferimenti
Docker$_G$	$\geq$ 24.0.7	https://docs.docker$_G$.com/
Docker Compose$_G$	$\geq$ 2.23.3	https://docs.docker$_G$.com/compose/

Tabella 1: Requisiti di sistema minimi

2.2 Requisiti hardware

Componente	Requisito$_G$ minimo
Processore	Processore a 64 bit con SLAT (Second Level Address Translation)
Memoria RAM	4GB DDR4
Spazio su disco	$\geq$ 20 GB

Tabella 2: Requisiti hardware

2.3 Browser$_G$

Browser$_G$	Versione$_G$
Google Chrome	$\geq$ 89
Microsoft Edge	$\geq$ 89
Mozilla Firefox	$\geq$ 16.4
Apple Safari	$\geq$ 108
Opera Browser$_G$	$\geq$ 76
Google Chrome per Android	$\geq$ 89
Apple Safari per iOS	$\geq$ 17.1
Samsung Internet	$\geq$ 23

Tabella 3: Browser$_G$ supportati

3 Tecnologie

3.1 Introduzione

In questa sezione, viene presentata una panoramica completa degli strumenti e delle tecnologie utilizzati per lo sviluppo e l'implementazione del software "WMS3". Questo include una descrizione dettagliata delle tecnologie, del linguaggio di programmazione adottato, delle librerie e dei framework necessari.

L'obiettivo principale è assicurare che il software sia sviluppato utilizzando le tecnologie adeguate in termini di efficacia ed efficienza e che soddisfi i requisiti individuati nel documento *Analisi dei Requisiti*$_G$ *v2.0.1*.

3.2 Tecnologie implementative

3.2.1 Next.js$_G$

- Framework di sviluppo web front-end basato su React$_G$ e utilizzato per la creazione di applicazioni web. Offre funzionalità avanzate realizzazione di API$_G$, gestione del routing e Server Action.
- **Versione$_G$:** 14.1.0.
- **Link:** https://nextjs.org/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$, JSX$_G$, JSON$_G$.
- **Contesto di utilizzo:**
  - Creazione della struttura dell'applicazione;
  - Implementazione delle API$_G$;
  - Gestione del routing.

3.2.2 React$_G$

- Libreria JavaScript$_G$ utilizzata per la creazione di interfacce utente$_G$ dinamiche, reattive e stateful. Si basa sul concetto di "components", ovvero blocchi di codice autonomi che gestiscono la propria logica e rendering.
- **Versione$_G$:** 18.0.0.
- **Link:** https://reactjs.org/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$, JSX$_G$, JSON$_G$.
- **Contesto di utilizzo:**
  - Creazione di interfacce utente$_G$ dinamiche;
  - Implementazione dei componenti web.

3.2.3 Node.js$_G$

- Runtime system orientato agli eventi per l'esecuzione di codice JavaScript$_G$ estendibile tramite moduli. Viene utilizzato per eseguire il codice JavaScript$_G$ lato server.
- **Versione$_G$:** 20.11.0.
- **Link:** https://nodejs.org/ *(ultimo accesso 04/04/2024)*

- **Linguaggio:** TypeScript$_G$, JSON$_G$.
- **Contesto di utilizzo:**
  - Codifica lato server.

3.2.4 Tailwind CSS

- Framework CSS utilizzato per lo sviluppo di interfacce utente$_G$ web. Offre una serie di classi predefinite per la definizione dello stile degli elementi.
- **Versione$_G$:** 3.4.1.
- **Link:** https://tailwindcss.com/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** CSS.
- **Contesto di utilizzo:**
  - Definizione stile e layout componenti web.

3.2.5 Shadcn/ui

- Raccolta di componenti React$_G$ personalizzati per la creazione di interfacce utente$_G$. Offre una serie di componenti pronti all'uso per la realizzazione di interfacce grafiche.
- **Versione$_G$:** 0.8.0.
- **Link:** https://ui.shadcn.com/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$, JSX$_G$.
- **Contesto di utilizzo:**
  - Creazione di interfacce utente$_G$ grafiche.

Librerie e framework ambiente 3D

3.2.6 Three.js$_G$

- Libreria JavaScript$_G$ utilizzata per creare e visualizzare grafica computerizzata 3D animata in un browser$_G$ Web utilizzando WebGL$_G$. Offre funzionalità avanzate per la creazione di ambienti 3D interattivi.
- **Versione$_G$:** 0.161.2.
- **Link:** https://threejs.org/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$.
- **Contesto di utilizzo:**
  - Creazione ambiente 3D.

3.2.7 @react$_G$-three/fiber

- Libreria open-source che facilita l'integrazione di Three.js$_G$ all'interno di applicazioni React$_G$. Offre funzionalità avanzate per la creazione di grafica 3D animata.
- **Versione$_G$:** 8.15.16.
- **Link:** https://docs.pmnd.rs/react$_G$-three-fiber/getting-started/introduction *(ultimo accesso 04/04/2024)*

- **Linguaggio:** TypeScript$_G$, JSX$_G$.
- **Contesto di utilizzo:**
  - Creazione ambiente 3D.

3.2.8 @react$_G$-three/drei

- Libreria che fornisce componenti e utilità per semplificare lo sviluppo di applicazioni in 3D utilizzando React$_G$ e Three.js$_G$. Offre funzionalità avanzate per la creazione di ambienti 3D interattivi.
- **Versione$_G$:** 9.97.6.
- **Link:** https://www.npmjs.com/package/@react$_G$-three/drei *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$, JSX$_G$.
- **Contesto di utilizzo:**
  - Creazione ambiente 3D.

3.3 Tecnologie per la validazione dei dati

3.3.1 Zod

- Zod è una libreria di validazione dei dati per TypeScript$_G$. Viene utilizzata per definire schemi di validazione dei dati e garantire che i dati ricevuti siano conformi a tali schemi.
- **Versione$_G$:** 3.22.4.
- **Link:** https://zod.dev/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$.
- **Contesto di utilizzo:**
  - Validazione dei dati inseriti dall'utente$_G$.

3.4 Tecnologie per la persistenza dei dati

3.4.1 PostgreSQL$_G$

- PostgreSQL$_G$ è un sistema di gestione di database$_G$ relazionali. Viene utilizzato per la memorizzazione e la gestione dei dati relativi al software "WMS3".
- **Versione$_G$:** 16.2.
- **Link:** https://www.postgresql$_G$.org/docs/16/index.html *(ultimo accesso 04/04/2024)*
- **Linguaggio:** SQL$_G$.
- **Contesto di utilizzo:**
  - Memorizzazione e gestione dei dati relativi ai bin$_G$, ai prodotti e alle zone del magazzino$_G$.

3.5 Tecnologie per il testing$_G$

3.5.1 Jest$_G$

- Jest$_G$ è un framework di testing$_G$ per JavaScript$_G$ e TypeScript$_G$. Viene utilizzato principalmente per lo unit e l'integration testing$_G$, offrendo funzionalità avanzate come la parallelizzazione dei test e il mocking delle dipendenze.
- **Versione$_G$:** 29.7.0.
- **Link:** https://jestjs.io/docs/getting-started *(ultimo accesso 04/04/2024)*
- **Linguaggio:** TypeScript$_G$, JavaScript$_G$.
- **Contesto di utilizzo:**
  - Implementazione della suite di unit testing$_G$;
  - Implementazione della suite di integration testing$_G$.

3.6 Tecnologie per il deployment

3.6.1 Docker$_G$

- Docker$_G$ è un software utilizzato per il processo di deployment di applicazioni software. Permette di eseguire processi informatici in ambienti isolati chiamati container, garantendo la portabilità e la scalabilità delle applicazioni.
- **Versione$_G$:** 24.0.7.
- **Link:** https://docs.docker$_G$.com/engine/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** YAML.
- **Contesto di utilizzo:**
  - Deployment del software "WMS3" mediante container Docker$_G$;
  - Isolamento dell'ambiente di sviluppo.
- **Immagini Docker$_G$ utilizzate:**
  - **PostgreSQL$_G$**: container per il database$_G$ relazionale;
    - Immagine: `postgres`$_G$`:16.2`.
  - **Web**: container per l'applicazione web;
    - Immagine: `node`$_G$`:20-alpine`.

3.6.2 Docker Compose$_G$

- Docker Compose$_G$ è uno strumento per la definizione e l'esecuzione di applicazioni multi-container. Viene utilizzato per gestire l'orchestrazione dei container Docker$_G$ e semplificare il processo di deployment.
- **Versione$_G$:** 2.23.3.
- **Link:** https://docs.docker$_G$.com/compose/ *(ultimo accesso 04/04/2024)*
- **Linguaggio:** YAML.
- **Contesto di utilizzo:**
  - Gestione dell'orchestrazione dei container Docker$_G$ utilizzati.

4 Architettura di sistema

4.1 Architettura di implementazione

Il software WMS3 al fine di perseguire manutenibilità, flessibilità e scalabilità, adotta ed implementa un'architettura "layered", nota anche come "Multi-tier architecture".

I layer definiti sono "closed", ovvero una richiesta si sposta esclusivamente da un livello superiore a quello immediatamente adiacente.

Tale architettura permette di individuare e suddividere la logica del software in 3 principali aspetti, definiti tier (separation of concerns), quali:

- **Persistence layer**: gestisce l'accesso al database$_G$ e fornisce gli strumenti dedicati alla lettura dei dati al suo interno. I dati letti vengono processati al fine di poter creare gli elementi del Business layer;
- **Business layer**: si occupa di elaborare i dati ricevuti dal layer di persistenza e applicare le regole di business definite. È responsabile di implementare la logica dell'applicazione in modo indipendente dalle tecnologie di persistenza e di presentazione utilizzate;
- **Presentation layer**: permette di trasformare i dati elaborati dal Business layer e le informazioni in una forma comprensibile e accessibile agli utenti finali. Questo include la creazione di interfacce utente$_G$ grafiche e visualizzazioni 3D degli elementi di interesse.

Ciascun layer possiede il suo sistema di classi e componenti e prevede metodi per comunicare con i layer adiacenti.

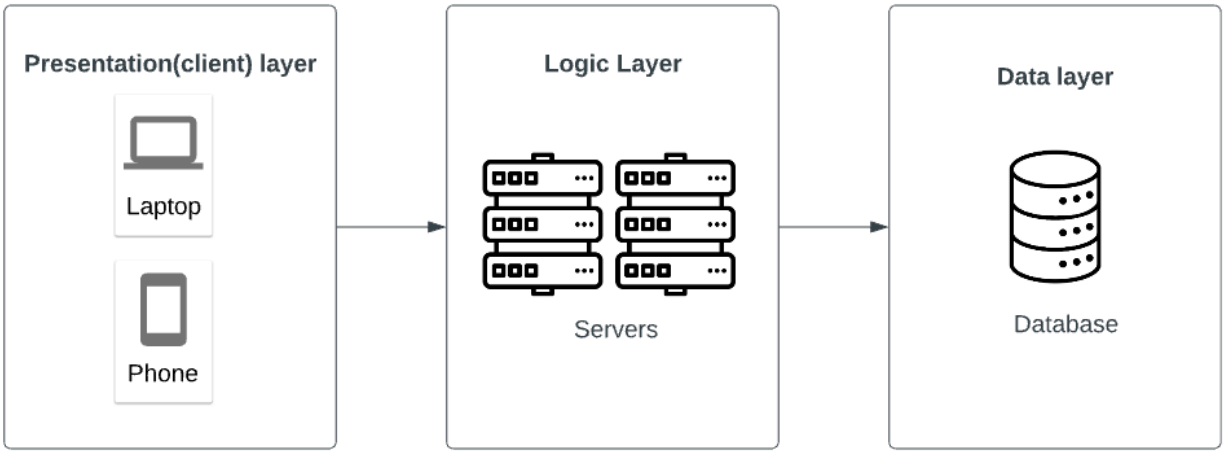

Figura 1: Layered architecture.

4.1.1 Vantaggi

- Ogni livello dell'archiettetura crea un livello di astrazione che permette di perseguire la *separation of concerns* e di rendere il software più manutenibile e scalabile;
- Semplicità di implementazione in termini di costi e tempo;
- Semplicità di test e debug.

4.1.2 Svantaggi

- Cambiamenti consistenti possono richiedere modifiche in layer diversi.

4.2 Persistence layer

Il layer di persistenza è responsabile della gestione dell'accesso al database$_G$ e della lettura dei dati al suo interno. I dati letti vengono processati e trasformati in oggetti del Business layer.

4.2.1 Next.js$_G$ Server Actions

L'accesso al database$_G$ avviene mediante Server Actions, feature$_G$ offerta da Next.js$_G$ che permette la definizione di codice eseguibile solamente lato server, senza eseguire operazioni di fetch tipiche delle chiamate API$_G$. Questo permette di non esporre sulla rete endpoint sensibili e di mantenere la sicurezza dei dati.

Quando viene richiamata una Server Action, Next.js$_G$ può restituire sia l'interfaccia utente$_G$ aggiornata sia i nuovi dati in un'unica risposta, integrandosi perfettamente con l'architettura *caching and revalidating* di Next.js$_G$. Tale architettura implementa una strategia di caching ottimizzata per massimizzare le prestazioni e minimizzare i costi. Questo comporta la generazione statica delle route e la memorizzazione nella cache delle richieste di dati. Il diagramma sottostante illustra il comportamento predefinito della cache: una route viene generata staticamente al momento della compilazione o quando viene visitata per la prima volta.

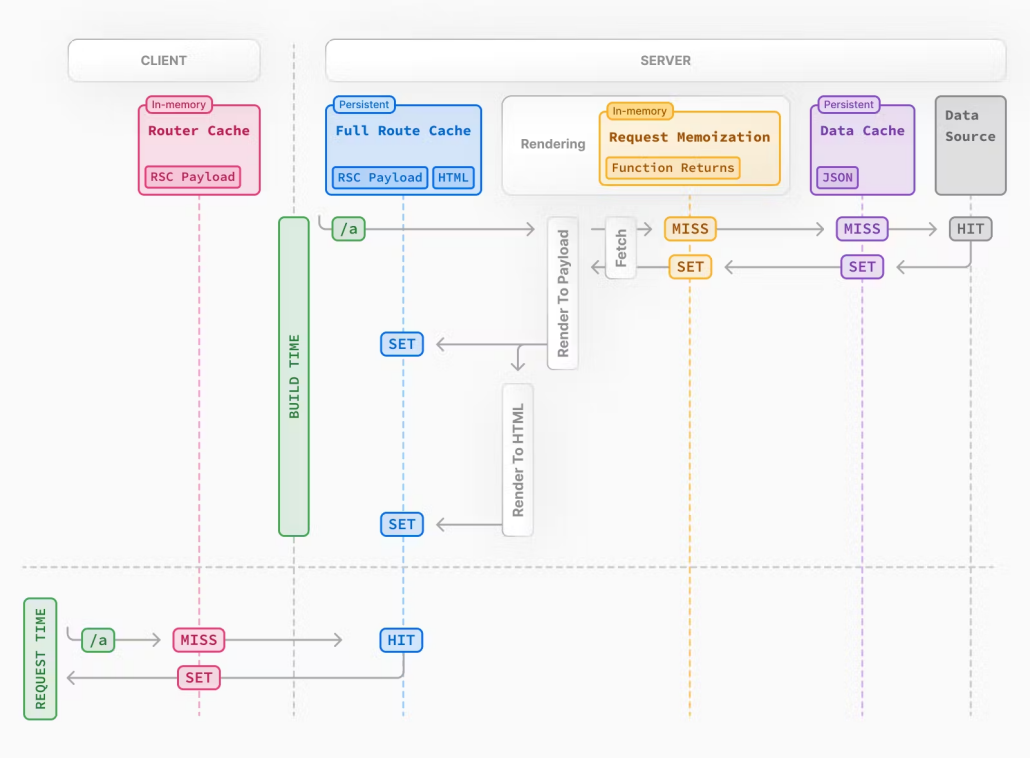

Figura 2: Next.js$_G$ caching and revalidating

Le Server Actions sono del tutto analoghe all'utilizzo di una funzione all'interno del codice, ma con la differenza che la loro esecuzione avviene lato server, mediante una chiamata POST interamente gestita da Next.js$_G$.

Il risultato della chiamata viene restituito in formato JSON$_G$.

4.2.2 Server Actions implementate

Le Server Actions implementate trovano applicazione in operazioni di lettura dal database$_G$, pertanto la natura asincrona delle operazioni implica l'utilizzo di Promise per la gestione dei risultati.

- **getAllBins**:
  - **obiettivo:** ottenere le informazioni di tutti i bin$_G$ presenti nel database$_G$;
  - **parametri:** nessuno;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente lista dei bin$_G$ presenti nel database$_G$;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getBinById**:
  - **obiettivo:** dato un codice identificativo univoco, ritorna le informazioni relative al bin$_G$ corrispondente lette dal database$_G$;
  - **parametri:** `id:string` del bin$_G$ interessato;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente le informazioni del bin$_G$ corrispondente;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getAllCategories**:
  - **obiettivo:** ottenere le informazioni di tutte le categorie presenti nel database$_G$;
  - **parametri:** nessuno;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente lista delle categorie presenti nel database$_G$;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getAllProducts**:
  - **obiettivo:** ottenere le informazioni di tutti i prodotti presenti nel database$_G$;
  - **parametri:** nessuno;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente lista dei prodotti presenti nel database$_G$;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getProductById**:
  - **obiettivo:** dato un codice identificativo univoco, ritorna le informazioni relative al prodotto corrispondente lette dal database$_G$;
  - **parametri:** `id:string` del prodotto interessato;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente le informazioni del prodotto corrispondente;
    - **Esito negativo:** JSON$_G$ vuoto.

- **SVGSanitize**:
  - **obiettivo:** dato un file SVG$_G$, ne effettua la sanificazione da elementi non necessari;
  - **parametri:** `svg`$_G$`:string` del file SVG$_G$ da sanificare;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente il file SVG$_G$ pulito;
    - **Esito negativo:** JSON$_G$ vuoto.

- **readSavedSVG**:
  - **obiettivo:** leggere il contenuto di un file SVG$_G$ salvato;
  - **parametri:** nessuno;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente il file SVG$_G$ salvato;
    - **Esito negativo:** JSON$_G$ vuoto.

- **saveSVG**:
  - **obiettivo:** salvare un file SVG$_G$ sul server;
  - **parametri:** `svg`$_G$`:string` del file SVG$_G$ da salvare;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente il file SVG$_G$ salvato;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getAllZones**:
  - **obiettivo:** ottenere le informazioni di tutte le zone presenti nel database$_G$;
  - **parametri:** nessuno;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente lista delle zone presenti nel database$_G$;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getBinsByZoneId**:
  - **obiettivo:** dato un codice identificativo univoco, ritorna le informazioni relative ai bin$_G$ presenti nella zona corrispondente lette dal database$_G$;
  - **parametri:** `id:string` della zona interessata;
  - **risultati:**
    - **Esito positivo:** JSON$_G$ contenente lista dei bin$_G$ presenti nella zona corrispondente;
    - **Esito negativo:** JSON$_G$ vuoto.

- **getZoneById**:
  - **obiettivo:** dato un codice identificativo univoco, ritorna le informazioni relative alla zona corrispondente lette dal database$_G$;

- **parametri:** `id:string` della zona interessata;
- **risultati:**
  - **Esito positivo:** JSON$_G$ contenente le informazioni della zona corrispondente;
  - **Esito negativo:** JSON$_G$ vuoto.

4.2.3 Repository$_G$ Pattern

Il Repository$_G$ Pattern permette di separare la logica di business dalla logica di accesso ai dati, garantendo una maggiore flessibilità e manutenibilità del codice. L'obiettivo è creare un livello di astrazione tra la logica di accesso ai dati e la logica di business, consentendo di modificare l'implementazione del database$_G$ senza influenzare il codice di business.

Vantaggi

- Dependency inversion principle: i moduli di alto livello non dipendono dai moduli di basso livello, rendendo la logica di business indipendente dalla logica di accesso ai dati;
- La separazione tra la logica di business e la logica di accesso ai dati semplifica la manutenzione e il testing$_G$ del codice.

Interfaccia implementata

- **DataRepositoryInterface**: interfaccia che definisce i metodi per il retrieve dei dati dal database$_G$.
  - **Metodi:**
    - **getAll**:
      - **obiettivo:** ottenere tutte le informazioni relative agli oggetti;
      - **parametri:** nessuno;
      - **risultati:**
        - **Esito positivo:** Promise contenente la lista degli oggetti;
        - **Esito negativo:** Promise contenente lista vuota.
    - **getById**:
      - **obiettivo:** ottenere le informazioni relative all'oggetto corrispondente al codice identificativo univoco;
      - **parametri:** `id:string` dell'oggetto interessato;
      - **risultati:**
        - **Esito positivo:** Promise contenente le informazioni dell'oggetto corrispondente;
        - **Esito negativo:** Promise contenente lsita vuota.

Classi implementate

L'interfaccia `DataRepositoryInterface` è implementata dalle seguenti classi:

- **productRepository**: è responsabile dell'ottenimento dei dati relativi agli oggetti `Product`.
  - **Metodi**:
    - **getAll**: ottiene tutte le informazioni relative ai prodotti;

    - **getById**: ottiene le informazioni relative al prodotto corrispondente al codice identificativo univoco.
- **zoneRepository**: è responsabile dell'ottenimento dei dati relativi agli oggetti `Zone`.
  - **Metodi**:
    - **getAll**: ottiene tutte le informazioni relative alle zone;
    - **getById**: ottiene le informazioni relative alla zona corrispondente al codice identificativo univoco.

4.2.4 Data Mapper Pattern

Il Data Mapper Pattern, assieme al Repository$_G$ Pattern, permette di separare la logica di business dalla logica di accesso ai dati. Il Data Mapper Pattern si occupa di mappare i dati letti dal database$_G$ in oggetti del Business layer, garantendo una maggiore flessibilità e manutenibilità del codice.

Permette la mappatura dei dati letti dal database$_G$ in oggetti del Business layer, stabilendo un contratto che i dati letti devono rispettare per essere trasformati in oggetti.

Vantaggi

- Separazione della logica di business dalla logica di accesso ai dati;
- Permette di circoscrivere la complessità relativa alla logica di creazione degli oggetti di business;
- Maggiore flessibilità e manutenibilità del codice;
- Facilità di testing$_G$ e debugging.

Interfaccia implementata

- **DataMapperInterface**: interfaccia che definisce i metodi per la creazione di oggetti a partire dai dati letti dal database$_G$.
  - **Metodi:**
    - **toDomain**: metodo astratto che definisce la logica di mappatura dei dati JSON$_G$ recuperati dal database$_G$ in oggetti di business.
      - **parametri**: `data:JSON`$_G$ contenente i dati recuperati dal database$_G$;
      - **risultati**: oggetto di tipo `T` corrispondente all'oggetto di business.

Classi implementate

- **binMapper**: è responsabile della creazione di oggetti `Bin`$_G$.
  - **Metodi:**
    - **toDomain**: mappatura dei dati JSON$_G$ in oggetti `Bin`$_G$.
- **productMapper**: è responsabile della creazione di oggetti `Product`.
  - **Metodi:**
    - **toDomain**: mappatura dei dati JSON$_G$ in oggetti `Product`.
- **zoneMapper**: è responsabile della creazione di oggetti `Zone`
  - **Metodi:**
    - **toDomain**: mappatura dei dati JSON$_G$ in oggetti `Zone`.

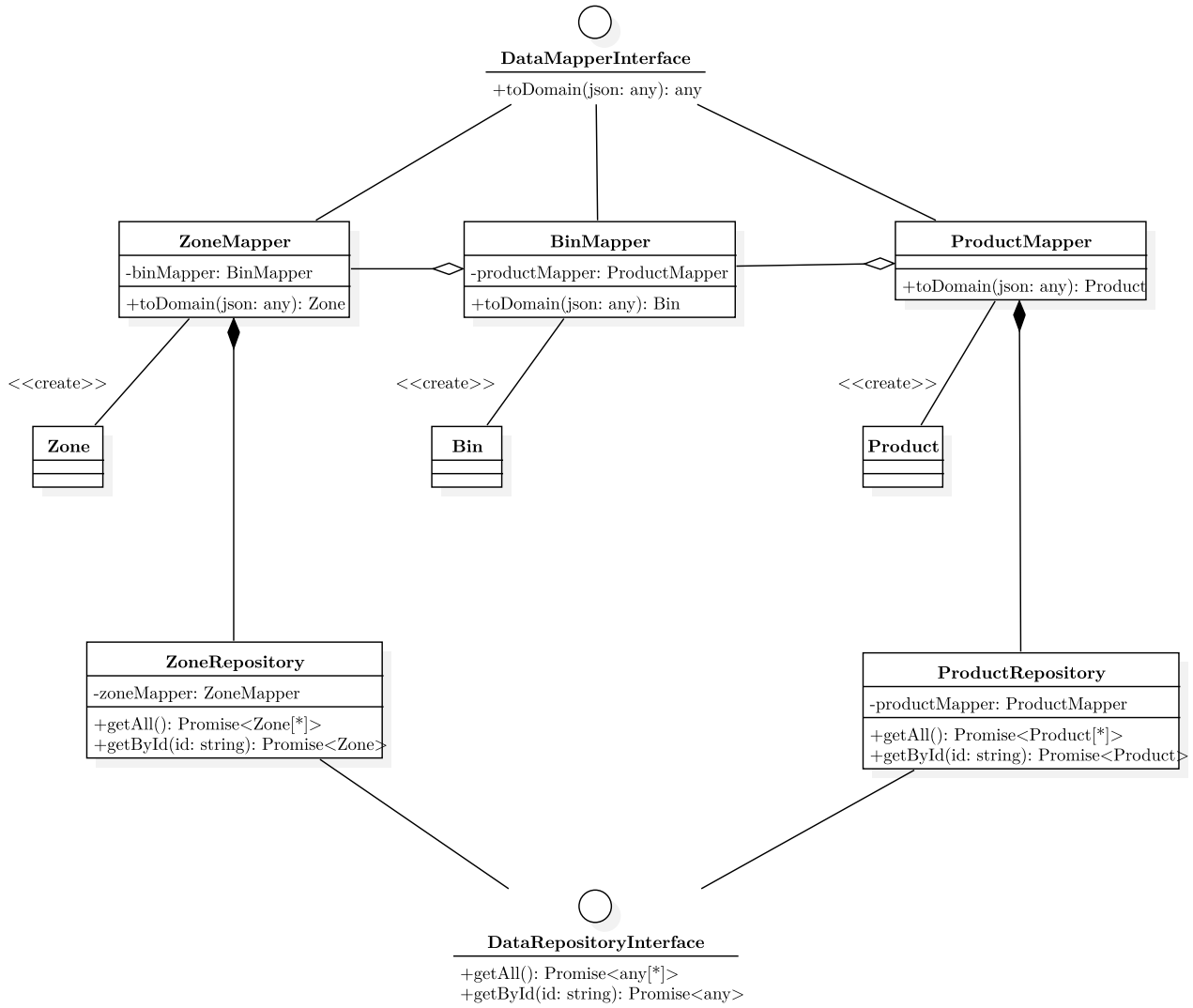

Figura 3: Diagramma delle classi del layer di persistenza

Nel diagramma delle classi del layer di persistenza fornito, le classi `Zone`, `Bin`$_G$ e `Product` sono rappresentate senza gli attributi e i metodi per garantire una maggiore chiarezza grafica. Tali informazioni sono rappresentate dettagliatamente nel diagramma delle classi del layer di business.

4.3 Business layer

Il layer di business è responsabile dell'elaborazione dei dati ricevuti dal layer di persistenza e dell'applicazione delle regole di business definite. È responsabile di implementare la logica dell'applicazione in modo indipendente dalle tecnologie di persistenza e di presentazione utilizzate.

4.3.1 Bin$_G$

Rappresenta un elemento bin$_G$, ovvero uno spazio definito in grado di contenere un prodotto.

Attributi:

- **id**:
  - **descrizione**: stringa di massimo 10 caratteri che rappresenta il codice identificativo univoco del bin_G. La struttura dell'id è la seguente:

    `idZona_letteraColonna_numeroLivello`

    La lettera corrispondente alla colonna fa riferimento ad una mappatura per cui "A" equivale alla colonna zero, e viene incrementata seguendo i caratteri dell'alfabeto inglese con l'aumentare del numero della colonna.

    Dopo la lettera "Z" viene utilizzato "AA" proseguendo con la logica descritta;
  - **tipo**: `string`;
  - **visibilità**: `private`.

- **level**:
  - **descrizione**: intero che rappresenta il numero del livello di appartenenza;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **column**:
  - **descrizione**: intero che rappresenta il numero della colonna di appartenenza;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **height**:
  - **descrizione**: numero in virgola mobile che rappresenta l'altezza del bin_G;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **length**:
  - **descrizione**: numero in virgola mobile che rappresenta la profondità del bin_G;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **width**:
  - **descrizione**: numero in virgola mobile che rappresenta la larghezza del bin_G;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **product**:
  - **descrizione**: oggetto `Product` che rappresenta il prodotto contenuto nel bin_G;
  - **tipo**: `Product`;
  - **visibilità**: `private`.

- **state**:
  - **descrizione**: enumerazione `BinState` che identifica lo stato di un bin_G contestualmente alla richiesta di spostamento dei prodotti.
    - **Valori dell'enumerazione**:
      - **Idle**: valore di default, dichiara che il bin_G non è coinvolto in richieste di spostamento di prodotti;
      - **ProductIncoming**: dichiara che il bin_G è coinvolto in una richiesta di spostamento di un prodotto, il quale deve essere immesso al suo interno;
      - **ProductOutgoing**: dichiara che il bin_G è coinvolto in una richiesta di spostamento del prodotto al suo interno, il quale deve essere prelevato.
  - **tipo**: `BinState`;
  - **visibilità**: `private`.

Metodi:
- **Getters**:
  - Getters presenti per ogni attributo della classe.
- **Setters**:
  - **setId**:
    - **obiettivo**: permette di modificare l'attributo `id`;
    - **parametri**: `id:string` che rappresenta il nuovo codice identificativo univoco;
    - **tipo di ritorno**: `void`.
  - **setProduct**:
    - **obiettivo**: permette di assegnare un prodotto al bin_G;
    - **parametri**: `product:Product` che rappresenta il prodotto da assegnare;
    - **tipo di ritorno**: `void`.
  - **setState**:
    - **obiettivo**: permette di modificare lo stato del bin_G;
    - **parametri**: `state:BinState` che rappresenta il nuovo stato;
    - **tipo di ritorno**: `void`.
- **clearProduct**:
  - **obiettivo**: permette di assegnare il valore `null` all'attributo `product`;
  - **parametri**: nessuno;
  - **tipo di ritorno**: `void`.

4.3.2 Zone

Rappresenta una zona di contenimento dei bin_G, interpretabile come $scaffale_G$ (se a più livelli) o area singola di stoccaggio (se a un solo livello).

Attributi:

- **id**:
  - **descrizione**: stringa di massimo 10 caratteri che rappresenta il codice identificativo univoco della zona;
  - **tipo**: `string`;
  - **visibilità**: `private`.

- **xcoordinate**:
  - **descrizione**: numero in virgola mobile che rappresenta la coordinata X di posizione nel piano;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **ycoordinate**:
  - **descrizione**: numero in virgola mobile che rappresenta la coordinata Y di posizione nel piano;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **height**:
  - **descrizione**: numero in virgola mobile che rappresenta l'altezza della zona;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **length**:
  - **descrizione**: numero in virgola mobile che rappresenta la profondità della zona;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **width**:
  - **descrizione**: numero in virgola mobile che rappresenta la larghezza della zona;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **bins**:
  - **descrizione**: lista di oggetti `Bin`$_G$ che rappresentano i bin$_G$ presenti nella zona;
  - **tipo**: `Bin`$_G$`[]`;
  - **visibilità**: `private`.

- **orientation**:
  - **descrizione**: enumerazione `ZoneOrientation` che identifica l'orientamento della zona rispetto all'asse X.
  - **tipo**: `boolean`:

  - **True**: orientamento orizzontale, parallelo all'asse X;
  - **False**: orientamento verticale, perpendicolare all'asse X.
- **visibilità**: `private`.

Metodi:

- **Getters**:
  - Getters presenti per ogni attributo della classe.
- **Setters**:
  - **setXCoordinate**:
    - **obiettivo**: permette di modificare la coordinata X di posizione nel piano;
    - **parametri**: `x:number` che rappresenta la nuova coordinata X;
    - **tipo di ritorno**: void.
  - **setYCoordinate**:
    - **obiettivo**: permette di modificare la coordinata Y di posizione nel piano;
    - **parametri**: `y:number` che rappresenta la nuova coordinata Y;
    - **tipo di ritorno**: void.
- **getBin**:
  - **obiettivo**: ottenere il bin_G corrispondente ad un codice identificativo univoco;
  - **parametri**: `id:string` del bin_G interessato;
  - **tipo di ritorno**: $\texttt{Bin}_G$ corrispondente al codice identificativo univoco fornito o `null`.
- **getLevels**:
  - **obiettivo**: ottenere una lista contenente le liste di bin_G che rappresentano i livelli della zona;
  - **parametri**: nessuno;
  - **tipo di ritorno**: lista di liste di bin_G $\texttt{Bin}_G\texttt{[[]]}$.
- **getColumns**:
  - **obiettivo**: ottenere una lista contenente le liste di bin_G che rappresentano le colonne della zona;
  - **parametri**: nessuno;
  - **tipo di ritorno**: lista di liste di bin_G $\texttt{Bin}_G\texttt{[[]]}$.
- **getMaxUsedLevel**:
  - **obiettivo**: ottenere il numero dell'ultimo livello della zona con almeno un bin_G contenente un prodotto;
  - **parametri**: nessuno;
  - **tipo di ritorno**: intero che rappresenta il numero dell'ultimo livello `number`.
- **getMaxUsedColumn**:
  - **obiettivo**: ottenere il numero dell'ultima colonna della zona con almeno un bin_G contenente un prodotto;
  - **parametri**: nessuno;
  - **tipo di ritorno**: intero che rappresenta il numero dell'ultima colonna `number`.

4.3.3 Product

Rappresenta il prodotto da gestire in magazzino$_G$.

Attributi:

- **id**:
  - **descrizione**: stringa di massimo 10 caratteri che rappresenta il codice identificativo univoco del prodotto;
  - **tipo**: `string`;
  - **visibilità**: `private`.

- **name**:
  - **descrizione**: stringa che rappresenta il nome del prodotto;
  - **tipo**: `string`;
  - **visibilità**: `private`.

- **weight**:
  - **descrizione**: numero in virgola mobile che rappresenta il peso del prodotto;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **height**:
  - **descrizione**: numero in virgola mobile che rappresenta l'altezza del prodotto;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **length**:
  - **descrizione**: numero in virgola mobile che rappresenta la profondità del prodotto;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **width**:
  - **descrizione**: numero in virgola mobile che rappresenta la larghezza del prodotto;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **categories**:
  - **descrizione**: lista di stringhe che rappresentano le categorie di appartenenza del prodotto;
  - **tipo**: `string[]`;
  - **visibilità**: `private`.

Metodi:

- **Getters**:
  - Getters presenti per ogni attributo della classe.

4.3.4 Order

Rappresenta un ordine di movimentazione di un prodotto tra un bin$_G$ di partenza e uno di destinazione.

Attributi:

- **id**:
  - **descrizione**: stringa di massimo 10 caratteri che rappresenta il codice identificativo univoco dell'ordine;
  - **tipo**: `string`;
  - **visibilità**: `private`.

- **startPoint**:
  - **descrizione**: oggetto `Bin`$_G$ che rappresenta il bin$_G$ di partenza dell'ordine;
  - **tipo**: `Bin`$_G$;
  - **visibilità**: `private`.

- **endPoint**:
  - **descrizione**: oggetto `Bin`$_G$ che rappresenta il bin$_G$ di destinazione dell'ordine;
  - **tipo**: `Bin`$_G$;
  - **visibilità**: `private`.

- **product**:
  - **descrizione**: oggetto `Product` che rappresenta il prodotto coinvolto nell'ordine;
  - **tipo**: `Product`;
  - **visibilità**: `private`.

Metodi:

- **Getters**:
  - Getters presenti per ogni attributo della classe.

4.3.5 SVG$_G$

Rappresenta un file SVG$_G$ utilizzato per la configurazione dell'ambiente di lavoro mediante file SVG$_G$.

Attributi:

- **length**:
  - **descrizione**: numero in virgola mobile che rappresenta la profondità dell'immagine rappresentata dal file;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **width**:
  - **descrizione**: numero in virgola mobile che rappresenta la larghezza dell'immagine rappresentata dal file;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **svg**$_G$:
  - **descrizione**: stringa che rappresenta il contenuto del file SVG$_G$;
  - **tipo**: `string`;
  - **visibilità**: `private`.

Metodi:
- **Getters**:
  - Getters presenti per ogni attributo della classe.

4.3.6 Floor

Rappresenta il piano dell'ambiente 3D.

Attributi:
- **length**:
  - **descrizione**: numero in virgola mobile che rappresenta la profondità del piano;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **width**:
  - **descrizione**: numero in virgola mobile che rappresenta la larghezza del piano;
  - **tipo**: `number`;
  - **visibilità**: `private`.

- **SVG**$_G$:
  - **descrizione**: oggetto `SVG`$_G$ che rappresenta il file SVG$_G$ del piano;
  - **tipo**: `SVG`$_G$;
  - **visibilità**: `private`.

Metodi:

- **Getters**:
  - Getters presenti per ogni attributo della classe.
- **Setters**:
  - Setters presenti per gli attributi `length`, `width` e $\texttt{SVG}_G$.
- **clone**:
  - **obiettivo**: permette di creare una copia dell'oggetto invocante;
  - **parametri**: nessuno;
  - **tipo di ritorno**: `Floor`.

4.3.7 FloorStrategy *Strategy Pattern*

Potendo generare l'oggetto `Floor` con modalità diverse a seconda della presenza del file SVG_G, la sua creazione è gestita tramite il design pattern Strategy e le relative classi che implementano l'interfaccia `FloorStrategy`.

La scelta di tale pattern è dettata dalla necessità di separare l'algoritmo di creazione dell'oggetto `Floor` dalla sua implementazione, permettendo di variare il comportamento dell'oggetto in base al contesto. L'input dell'algoritmo di creazione è il medesimo, ma il comportamento varia a seconda della strategia adottata.

Interfaccia implementata

- **FloorStrategy**: interfaccia che definisce il metodo per la creazione di un oggetto `Floor`.
  - **Metodi**:
    - **createFloor**: metodo astratto che definisce la logica di creazione dell'oggetto `Floor`.
      - **parametri**: `URLSearchParams` che rappresenta i parametri per la creazione dell'oggetto `Floor`;
      - **tipo di ritorno**: Promise contente l'oggetto `Floor` creato.

Classi implementate

Le classi che implementano l'interfaccia `FloorStrategy` sono:

- **StandardFloorStrategy**:
  - **obiettivo**: rappresenta la creazione di un elemento `Floor` senza file SVG_G;
  - **metodo**: `createFloor`.
- **CustomFloorStrategy**:
  - **obiettivo**: rappresenta la creazione di un elemento `Floor` con file SVG_G;
  - **metodo**: `createFloor`.

La decisione in merito alla strategia da adottare è sancita dalla classe `FloorStrategyContext`. Questa classe, in base alla presenza o meno del file SVG_G, sceglie la strategia da adottare per la creazione dell'oggetto `Floor`.

FloorStrategyContext

Attributi

- **strategy**: oggetto che implementa l'interfaccia `FloorStrategy` che rappresenta la strategia da adottare per la creazione dell'oggetto `Floor`.

Metodi

- **Setters**:
  - **setStrategy**: metodo che permette di impostare la strategia da adottare per la creazione dell'oggetto `Floor`.
- **createFloor**:
  - **obiettivo**: metodo che permette di creare l'oggetto `Floor` in base alla strategia adottata;
  - **parametri**: `URLSearchParams` che rappresenta i parametri per la creazione dell'oggetto `Floor`;
  - **tipo di ritorno**: Promise contenente l'oggetto `Floor` creato.

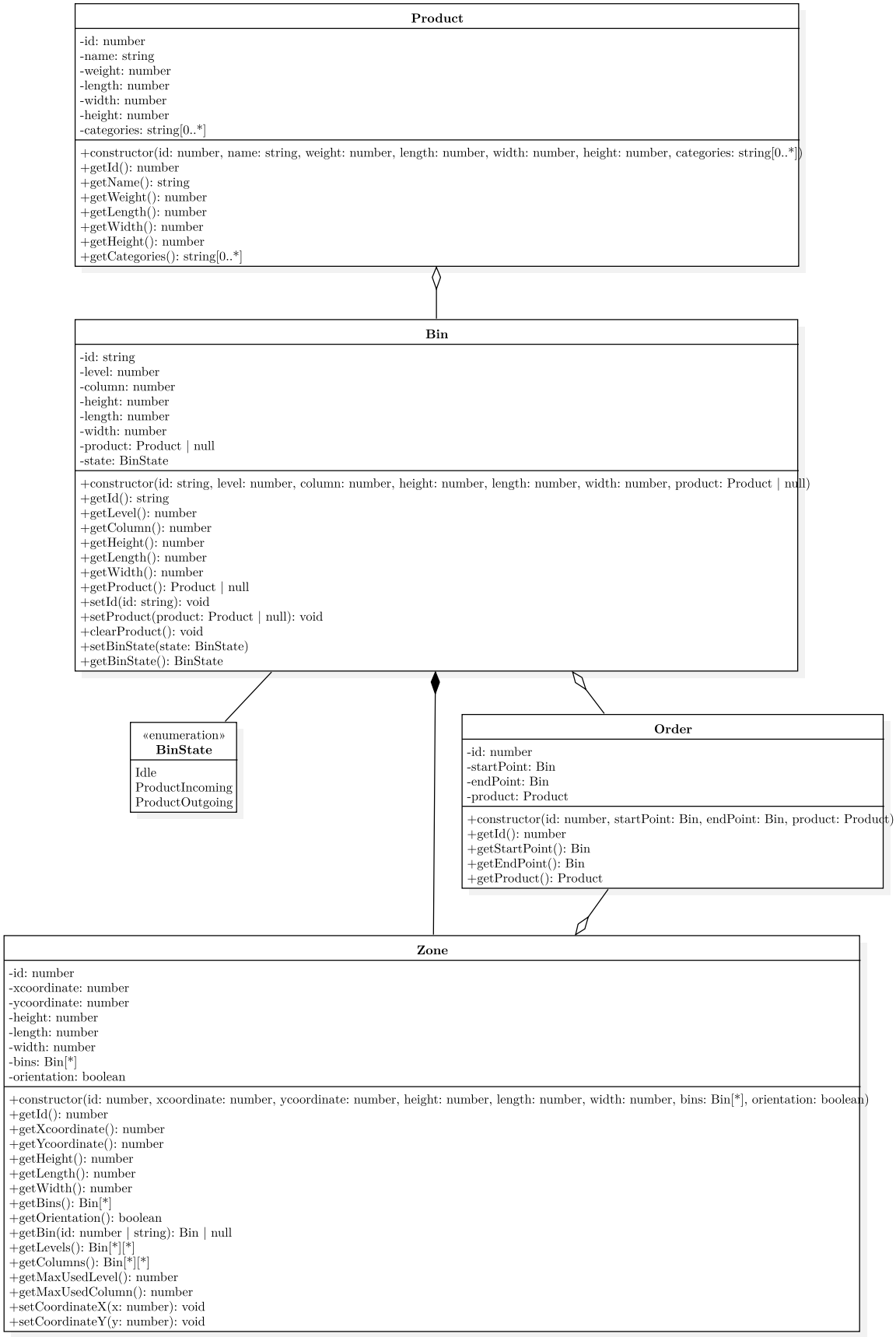

Figura 4: Diagramma delle classi del layer di business

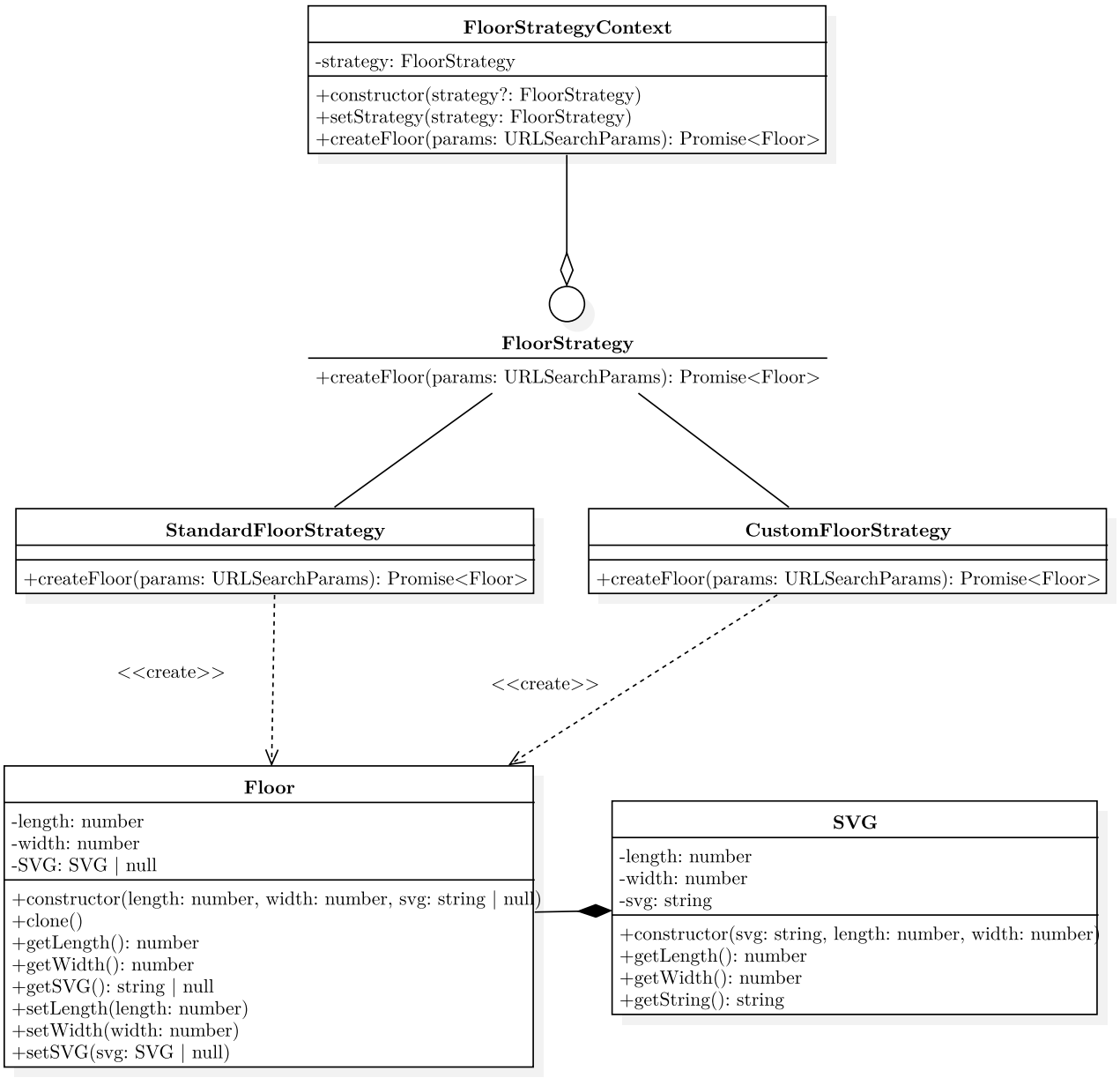

Figura 5: Diagramma delle classi Floor del layer di business

4.3.8 Search Engine *Factory e Strategy Pattern*

Il Factory Pattern, utilizzato insieme allo Strategy Pattern, permette di gestire la possibilità di cercare specifici prodotti e specifiche zone. Disponendo infatti di famiglie diverse di algoritmi per la ricerca (Strategy Pattern per Zone e Product), il Factory Pattern permette di adottare il corretto algoritmo in funzione del tipo di oggetto fornito.

Lo Strategy Pattern permette di definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Questo pattern permette di variare l'algoritmo indipendentemente dal contesto in cui viene utilizzato, rendendo facilmente estendibile il sistema di ricerca.

Interfaccia implementata

- **SearchStrategy<T extends Zone | Product>**: interfaccia che definisce il metodo per la ricerca di un oggetto. L'interfaccia è parametrizzata in modo da poter essere utilizzata per la ricerca di oggetti di tipo `Zone` o `Product`.

Metodi

- **search**: metodo astratto che definisce la logica di ricerca dell'oggetto.
  - **parametri**:
    - **list : T**[]: lista di oggetti in cui cercare;
    - **query : string**: stringa da cercare;
    - **type : string**: tipo di ricerca da effettuare;
  - **tipo di ritorno**: Promise contenente la lista di oggetti che rispondono al parametro di ricerca.

Classi implementate

- **ProductSearchStrategy**:

  Permette la ricerca di prodotti in base al loro codice identificativo, nome o categoria;

  Attributi

  - Nessuno.

  Metodi

  - **search**:
    - **obiettivo**: ricerca di prodotti in base al loro codice identificativo, nome o categoria;
    - **parametri**:
      - **list : Product**[]: lista di prodotti in cui cercare;
      - **query : string**: stringa da cercare;
      - **type : string**: tipo di ricerca da effettuare;
        - **Valori ammessi**: "id", "name", "category".
    - **tipo di ritorno**: lista di prodotti che rispondono al parametro di ricerca `Product[]`.

  - **searchById**:
    - **obiettivo**: ricerca di prodotti in base al loro codice identificativo;
    - **parametri**:
      - **list : Product**[]: lista di prodotti in cui cercare;
      - **query : string**: stringa da cercare.
    - **tipo di ritorno**: lista di prodotti che rispondono al parametro di ricerca `Product[]`.

  - **searchByName**:
    - **obiettivo**: ricerca di prodotti in base al loro nome;
    - **parametri**:

  - **list : Product**[]: lista di prodotti in cui cercare;
  - **query : string**: stringa da cercare.
- **tipo di ritorno**: lista di prodotti che rispondono al parametro di ricerca `Product[]`.

– **searchByCategory**:
- **obiettivo**: ricerca di prodotti in base alla categoria;
- **parametri**:
  - **list : Product**[]: lista di prodotti in cui cercare;
  - **query : string**: stringa da cercare.
- **tipo di ritorno**: lista di prodotti che rispondono al parametro di ricerca `Product[]`.

- **ZoneSearchStrategy**:

  Permette la ricerca di zone in base al loro codice identificativo.

  Attributi

  – Nessuno.

  Metodi

  – **search**:
  - **obiettivo**: ricerca di zone in base al loro codice identificativo;
  - **parametri**:
    - **list : Zone**[]: lista di zone in cui cercare;
    - **query : string**: stringa da cercare.
    - **type : string**: tipo di ricerca da effettuare;
      - **Valori ammessi**: “id”.
  - **tipo di ritorno**: lista di zone che rispondono al parametro di ricerca `Zone[]`.

  – **searchById**:
  - **obiettivo**: ricerca di zone in base al loro codice identificativo;
  - **parametri**:
    - **list : Zone**[]: lista di zone in cui cercare;
    - **query : string**: stringa da cercare;
  - **tipo di ritorno**: lista di zone che rispondono al parametro di ricerca `Zone[]`.

- **SearchStrategyFactory**:

Permette la creazione di oggetti `SearchStrategy` in base al tipo di oggetto da cercare.

Attributi

- Nessuno.

Metodi

- **static createSearchStrategy<T extends Zone | Product>**:
  - **obiettivo**: creazione di oggetti `SearchStrategy` in base al tipo di oggetto da cercare;
  - **parametri**:
    - **type : string**: tipo di oggetto da cercare;
      - **Valori ammessi**: “zone”, “product”.
  - **tipo di ritorno**: oggetto `SearchStrategy` corrispondente al tipo di oggetto da cercare.

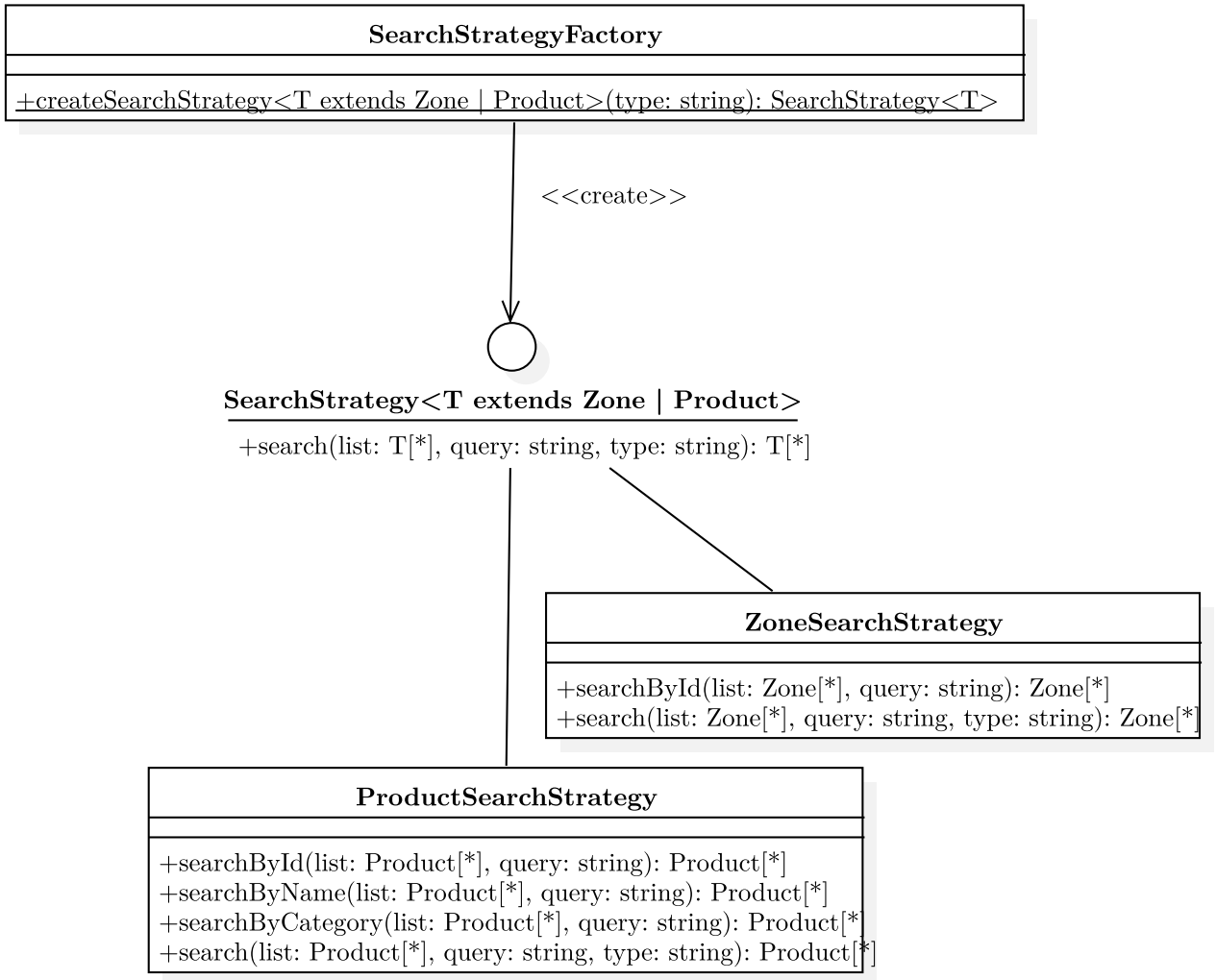

Figura 6: Diagramma delle classi Search del layer di business

4.3.9 Provider Pattern

Il Provider Pattern permette di gestire lo stato dell’applicazione in modo centralizzato, permettendo di mantenere lo stato dell’applicazione in un unico punto, semplificando la gestione e la condivisione dei dati tra i componenti. Questo pattern inoltre previene il fenomeno di prop drilling, ovvero la necessità di passare attraverso più livelli di componenti valori che non sono utilizzati direttamente dal componente intermedio.

Inoltre, creando un sistema di controllo dello stato centralizzato e condiviso, si evita la duplicazione dei dati e si facilita la manutenzione del codice.

Questo pattern è derivativo dall'utilizzo di React$_G$ come framework, in quanto la sua stessa natura funzionale è incentrata sulla gestione dello stato dell'applicazione e dei componenti. Le Context API $_G$ pertanto prevedono la possibilità di dichiarare un contesto contenente dati e funzioni che possono essere condivisi tra i componenti figli.

Questo contesto, mediante i componenti Provider implementati, permette di condividere lo stato dell'applicazione tra i componenti figli, evitando la necessità di passare manualmente i dati attraverso i componenti intermedi.

Componenti implementati

Provider del modello

- **zonesProvider**:
  - **obiettivo**: fornisce un provider per gestire dati relativi agli oggetti `Zone`;
  - **stato gestito**:
    - **zones**: lista di oggetti `Zone` che rappresentano le zone dell'ambiente di lavoro;
      - **tipo**: `Zone[]`.
    - **setZones**: funzione che permette di modificare la lista di oggetti `Zone`;
      - **tipo**: `(zones: Zone[]) => void`.
    - **zonesLoaded**: booleano che rappresenta lo stato di caricamento delle zone;
      - **tipo**: `boolean`.
    - **setZonesLoaded**: funzione che permette di modificare lo stato di caricamento delle zone.
      - **tipo**: `(loaded: boolean) => void`.
  - **tipo**: `JSX`$_G$`.Element`.

- **productsProvider**:
  - **obiettivo**: fornisce un provider per gestire dati relativi agli oggetti `Product`;
  - **stato gestito**:
    - **products**: lista di oggetti `Product` che rappresentano i prodotti dell'ambiente di lavoro;
      - **tipo**: `Product[]`.
    - **setProducts**: funzione che permette di modificare la lista di oggetti `Product`;
      - **tipo**: `(products: Product[]) => void`.
    - **productsLoaded**: booleano che rappresenta lo stato di caricamento dei prodotti;
      - **tipo**: `boolean`.
    - **setProductsLoaded**: funzione che permette di modificare lo stato di caricamento dei prodotti.
      - **tipo**: `(loaded: boolean) => void`.
  - **tipo**: `JSX`$_G$`.Element`.

- **ordersProvider**:
  - **obiettivo**: fornisce un provider per gestire dati relativi agli oggetti `Order`;

  - **stato gestito**:
    - **orders**: lista di oggetti `Order` che rappresentano gli ordini dell'ambiente di lavoro;
      - **tipo**: `Order[]`.
    - **setOrders**: funzione che permette di modificare la lista di oggetti `Order`.
      - **tipo**: `(orders: Order[]) => void`.
  - **tipo**: $\texttt{JSX}_G\texttt{.Element}$.

- **floorProvider**:
  - **obiettivo**: fornisce un provider per gestire dati relativi all'oggetto `Floor`;
  - **stato gestito**:
    - **floor**: oggetto `Floor` che rappresenta il piano dell'ambiente di lavoro;
      - **tipo**: `Floor`.
    - **setFloor**: funzione che permette di modificare l'oggetto `Floor`;
      - **tipo**: `(floor: Floor) => void`.
    - **floorRefresher**: valore intero utilizzato per forzare il refresh del piano;
      - **tipo**: `number`.
    - **setFloorRefresher**: funzione che permette di modificare il valore di `floorRefresher`.
      - **tipo**: `(refresher: number) => void`.
  - **tipo**: $\texttt{JSX}_G\texttt{.Element}$.

Provider per elementi UI

- **ElementDetailsProvider**
  - **obiettivo**: fornisce un provider per gestire dati relativi ai componenti da visualizzare in un pannello dedicato;
  - **stato gestito**:
    - **elementDetails**: oggetto che rappresenta il componente da visualizzare;
      - **tipo**: $\texttt{JSX}_G\texttt{.Element}$.
    - **setElementDetails**: funzione che permette di modificare l'oggetto `element`;
      - **tipo**: `(element: JSX`$_G$`.Element) => void`.
    - **showElementDetails**: booleano che rappresenta la visibilità del pannello;
      - **tipo**: `boolean`;
    - **setShowElementDetails**: funzione che permette di modificare la visibilità del pannello.
      - **tipo**: `(show: boolean) => void`.
  - **tipo**: $\texttt{JSX}_G\texttt{.Element}$.

- **formContextProvider**:
  - **obiettivo**: fornisce un provider per gestire dati relativi allo stato di processing di un form. Utilizzato per il lo stato di processing del file SVG_G caricato durante il processo di configurazione dell'ambiente;
  - **stato gestito**:
    - **processing**: booleano che rappresenta lo stato di processing del form;

  - **tipo**: `boolean`.
- **setProcessing**: funzione che permette di modificare lo stato di processing del form.
  - **tipo**: `(processing: boolean) => void`.
- **tipo**: `JSX`$_G$`.Element`.

Provider ambiente 3D

- **warehouseProvider**
  - **obiettivo**: fornisce un provider per gestire dati relativi alle variabili di contesto dell'ambiente grafico;
  - **stato gestito**:
    - **selectedBin**: oggetto `Bin`$_G$ che rappresenta il bin$_G$ selezionato;
      - **tipo**: `Bin`$_G$.
    - **setSelectedBin**: funzione che permette di modificare l'oggetto `selectedBin`;
      - **tipo**: `(bin`$_G$`: Bin`$_G$`) => void`.
    - **gridCellSize**: numero in virgola mobile che rappresenta la dimensione di una cella della griglia;
      - **tipo**: `number`.
    - **setGridCellSize**: funzione che permette di modificare la dimensione di una cella della griglia;
      - **tipo**: `(size: number) => void`.
    - **moveCameraToPosition**: funzione che permette di spostare la camera in una posizione specifica.
      - **tipo**: `(position: Vector3) => void`.
    - **cameraRef**: riferimento alla camera dell'ambiente 3D.
      - **tipo**: `RefObject\<CameraControls\>`.

4.4 Presentation layer

4.4.1 UI

L'interfaccia utente$_G$ è realizzata mediante elementi importati da shadcn/u e componenti personalizzati.

I componenti realizzati sono i seguenti:

- **Form di configurazione dell'ambiente**:
  - **creationForm**:
    - **descrizione**: componente dinamico dedicato alla creazione dell'ambiente di lavoro. Rappresenta la struttura portante del form di configurazione dell'ambiente di lavoro, contenendo al suo interno i componenti `manualCreationFrame` e `svgCreationFrame`.
    - **interazione con l'utente**$_G$:
      - **RadioGroup**: permette di selezionare il metodo di creazione dell'ambiente di lavoro;
        - **nome:** "choice";

- **opzioni**:
  - **manuale**: “Creazione manuale”;
  - **custom**: “Creazione mediante file SVG$_G$“.

– **manualCreationFrame**: componente dedicato alla creazione manuale dell’ambiente di lavoro;

– **svgCreationFrame**: componente dedicato alla definizione dei parametri di creazione dell’ambiente di lavoro mediante file SVG$_G$;

– **CheckBox**: permette di selezionare se importare o meno i dati relativi ai Prodotti dal database$_G$.
  - **nome**: “loadProdotti”;

- **validazione**:
  – **zodScheme**: schema dinamico dedicato alla validazione dei dati di creazione dell’ambiente di lavoro. La validazione è effettuata in base al metodo di creazione selezionato determinato dal valore di “choice”.
    - **manualCreationSchema**: schema dedicato alla validazione dei dati di creazione dell’ambiente di lavoro mediante configurazione manuale. In particolare:
      – **choice**: deve corrispondere a “manuale”;
      – **loadProdotti**: `boolean`;
      – **larghezza**: `number` e maggiore di 0;
      – **profondità**: `number` e maggiore di 0.
    - **svgCreationSchema**: schema dedicato alla validazione dei dati di creazione dell’ambiente di lavoro mediante file SVG$_G$. In particolare:
      – **choice**: deve corrispondere a “custom”;
      – **loadProdotti**: `boolean`;
      – **loadScaffali**: `boolean`;
      – **latoMaggiore**: `number` e maggiore di 0;
      – **svg$_G$**: stringa non vuota.

– **manualCreationFrame**:
  - **descrizione**: componente dedicato alla creazione manuale dell’ambiente di lavoro. Contiene i campi relativi alla definizione delle dimensioni del piano.
  - **interazione con l’utente$_G$**:
    – **Input**: permette di inserire la larghezza del piano;
      - **nome**: “larghezza”.
    – **Input**: permette di inserire la profondità del piano.
      - **nome**: “profondità”.
  - **validazione**:
    – **manualCreationSchema**.

– **svgCreationFrame**:

- **descrizione**: componente dedicato alla definizione dei parametri di creazione dell'ambiente di lavoro mediante file SVG$_G$;
- **interazione con l'utente**$_G$:
  - **dropFileArea**: permette di caricare un file SVG$_G$;
  - **Input**: permette di inserire il lato maggiore del piano;
    - **nome**: "latoMaggiore".
  - **CheckBox**: permette di selezionare se importare o meno i dati relativi ai Scaffali dal database$_G$.
    - **nome**: "loadScaffali".
- **validazione**:
  - **svgCreationSchema**.

- **dropFileArea**:
  - **descrizione**: componente dedicato al caricamento di un file SVG$_G$;
  - **interazione con l'utente**$_G$:
    - **FileInput**: permette di selezionare un file da caricare;
      - **nome**: "file";
      - **formato file ammesso**: "image/svg$_G$+xml";
      - **dimensione massima file:** 10MB.
  - **validazione**:
    - **svgCreationSchema**.

- **Componenti relativi ai prodotti**:
  - **productItem**:
    - **descrizione**: componente dedicato alla visualizzazione di un prodotto;
    - **parametri**:
      - **product**: oggetto `Product` che rappresenta il prodotto da visualizzare.
    - **informazioni visualizzate**:
      - **ID**: codice identificativo univoco del prodotto;
      - **Nome**: nome del prodotto;
      - **Categorie**: categorie di appartenenza del prodotto.
    - **interazione con l'utente**$_G$:
      - **Button**: permette di visualizzare i dettagli del prodotto;
        - **azione**: mostra i dettagli del prodotto mediante il componente `productItemDetails`.
  - **productsPanel**:
    - **descrizione**: componente dedicato alla visualizzazione di tutti i prodotti presenti nel magazzino$_G$. Ogni prodotto è rappresentato da un `ProductItem`;
    - **interazione con l'utente**$_G$:
      - **SearchBar**: permette di cercare un prodotto all'interno della lista;
        - **azione**: filtra i prodotti presenti in base alla stringa inserita;
        - **tipologia di ricerca**:

        - **ID**: ricerca per codice identificativo univoco;
        - **Nome**: ricerca per nome.
    - **Combobox**: permette di selezionare la categoria di appartenenza dei prodotti da visualizzare;
      - **opzioni**: lista di categorie di appartenenza dei prodotti presenti nel magazzino$_G$.
    - **Tabs**: permette di visuallizza o i prodotti collocati o i prodotti non collocati.
      - **opzioni**:
        - **Collocati**: visualizza solo i prodotti collocati;
        - **Non collocati**: visualizza solo i prodotti non collocati.

  - **productItemDetails**:
    - **descrizione**: componente dedicato alla visualizzazione delle informazioni dettagliate di un prodotto;
    - **parametri**:
      - **product**: oggetto `Product` che rappresenta il prodotto di cui visualizzare i dettagli.
    - **informazioni visualizzate**:
      - **ID**: codice identificativo univoco del prodotto;
      - **Nome**: nome del prodotto;
      - **Peso**: peso del prodotto;
      - **Larghezza**: larghezza del prodotto;
      - **Lunghezza**: lunghezza del prodotto;
      - **Altezza**: altezza del prodotto;
      - **Categorie**: categorie di appartenenza del prodotto.

- **Componenti relativi ai bin$_G$**:
  - **binItemDetails**:
    - **descrizione**: componente dedicato alla visualizzazione delle informazioni dettagliate di un bin$_G$;
    - **parametri**:
      - **bin$_G$**: oggetto `Bin`$_G$ che rappresenta il bin$_G$ di cui visualizzare i dettagli.
    - **informazioni visualizzate**:
      - **ID**: codice identificativo univoco del bin$_G$;
      - **Larghezza**: larghezza del bin$_G$;
      - **Lunghezza**: lunghezza del prodotto;
      - **Altezza**: altezza del prodotto;
      - **ProductItemDetails**: componente dedicato alla visualizzazione dei dati di un prodotto. Visibile solo se all'interno del bin$_G$ è presente un prodotto.
    - **interazione con l'utente$_G$**:
      - **ProductComboBox**: nel caso il bin$_G$ fosse vuoto, è possibile collocare al suo interno un prodotto selezionandolo dalla lista dei prodotti non collocati.

  - **ProductComboBox**:
    - **descrizione**: componente dedicato alla selezione di un prodotto da collocare in un bin$_G$;
    - **parametri**:
      - **bin$_G$**: oggetto `Bin`$_G$ che rappresenta il bin$_G$ in cui collocare il prodotto.
    - **interazione con l'utente$_G$**:
      - **Combobox**: permette di selezionare un prodotto dalla lista dei prodotti non collocati.
        - **opzioni**: lista di oggetti `Product` non collocati.

- **Componenti relativi agli ordini**:
  - **orderItem**:
    - **descrizione**: componente dedicato alla visualizzazione di un ordine;
    - **parametri**:
      - **order**: oggetto `Order` che rappresenta l'ordine da visualizzare.
    - **informazioni visualizzate**:
      - **ID**: codice identificativo univoco dell'ordine;
      - **Prodotto**: nome del prodotto coinvolto nell'ordine;
      - **Bin$_G$ di partenza**: codice identificativo univoco del bin$_G$ di partenza;
      - **Bin$_G$ di destinazione**: codice identificativo univoco del bin$_G$ di destinazione.
  - **ordersPanel**:
    - **descrizione**: componente dedicato alla visualizzazione di tutti gli ordini presenti nel magazzino$_G$. Ogni ordine è rappresentato da un `OrderItem`.

- **Componenti relativi alle impostazioni**:
  - **settingsPanel**:
    - **descrizione**: componente dedicato alla visualizzazione delle impostazioni dell'applicazione e della versione$_G$ dell'applicativo;
    - **interazione con l'utente$_G$**:
      - floorDimensionsItem: componente dedicato alla visualizzazione e modifica delle dimensioni del piano;
      - restoreItem: componente dedicato al ripristino o alla reimpostazione dell'ambiente di lavoro.
    - **validazione**:
      - **zodDimensionScheme**: schema dedicato alla validazione dei dati dimensionali per la modifica del piano. In particolare:
        - **larghezza**: number e maggiore di 0;
        - **profondità**: number e maggiore di 0.

  - **floorDimensionsItem**:
    - **descrizione**: componente dedicato alla visualizzazione e modifica delle dimensioni del piano;
    - **interazione con l'utente$_G$**:

  - **Input**: permette di inserire la larghezza del piano;
    - **nome**: “larghezza”.
  - **Input**: permette di inserire la profondità del piano.
    - **nome**: “profondità”.
- **validazione**:
  - **zodDimensionScheme**.

- **restoreItem**:
  - **descrizione**: componente dedicato al ripristino o alla reimpostazione dell’ambiente di lavoro;
  - **interazione con l’utente**$_G$:
    - **Button**: permette di ripristinare l’ambiente di lavoro;
      - **azione**: ripristina l’ambiente di lavoro allo stato iniziale.
    - **Button**: permette di reimpostare l’ambiente di lavoro.
      - **azione**: riporta al `creationForm` avviando nuovamente la procedura di configurazione dell’ambiente.

- **Zone**:
  - **zoneCreationFrame**:
    - **descrizione**: componente dedicato alla creazione e/o modifica di una zona;
    - **parametri**:
      - **zone**: oggetto `Zone` che rappresenta la zona da modificare, opzionale.
    - **interazione con l’utente**$_G$:
      - **Input**: permette di insereire l’ID della zona (disabilitato in caso di modifica);
        - **nome**: “ID”.
      - **Input**: permette di inserire la lunghezza della zona;
        - **nome**: “lunghezza”.
      - **Input**: permette di inserire la larghezza della zona;
        - **nome**: “larghezza”.
      - **Input**: permette di inserire l’altezza della zona;
        - **nome**: “altezza”.
      - **Combobox**: permette di selezionare l’orientamento della zona;
        - **opzioni**: “Verticale”, “Orizzontale”.
      - **RadioGroup**: permette di indicare la modalità di definizione delle colonne:
        - **opzioni**:
          - **manuale**: Abilita il campo “nColumns” per l’inserimento manuale del numero di colonne di larghezza uguale in cui suddividere la zona;
          - **custom**: Abilita il campo “customColumns” per l’inserimento della stringa rappresentate le dimensioni delle colonne in cui suddividere la zona.
            - **formato**: “dim1 dim2 dim3 … dimn”.
      - **Button**: permette di incrementare il numero di livelli della zona;

    - **azione**: aggiunge un livello alla zona. Ogni livello aggiunto è rappresentato dal componente `levelItem`.
  - **Button**: permette di salvare la zona.
    - **azione**: salva la zona.
- **validazione**:
  - **zoneZodSchemes**: schema dedicato alla validazione dei dati di creazione e modifica di una zona. In particolare:
    - **ID**: stringa non vuota;
    - **lunghezza**: `number` e maggiore di 0;
    - **larghezza**: `number` e maggiore di 0;
    - **altezza**: `number` e maggiore di 0;
    - **orientamento**: stringa non vuota;
    - **nColumns**: `number` e maggiore di 0;
    - **customColumns**: stringa non vuota.

- **levelItem**:
  - **descrizione**: componente dedicato alla visualizzazione di un ripiano$_G$ durante il processo di modifica o creazione;
  - **informazioni visualizzate**:
    - **Livello del piano**: livello identificativo del ripiano$_G$: indica la posizione del ripiano$_G$ all'interno dello scaffale$_G$;
    - **Altezza**: altezza del ripiano$_G$.
  - **interazione con l'utente**$_G$:
    - **Input**: permette di inserire l'altezza del ripiano$_G$;
      - **nome**: "altezza".
    - **Button**: permette di eliminare il ripiano$_G$.
      - **azione**: elimina il ripiano$_G$.

- **bin**$_G$**_columns**:
  - **descrizione**: definisce le colonne del componente `data-table` utilizzato all'interno di `zoneItemDetails` per la visualizzazione dei bin$_G$ presenti all'interno della zona interssata. Le colonne definite sono:
    - **Id**: id del bin$_G$;
    - **Prodotto**: nome del prodotto presente all'interno del bin$_G$ (se presente);
    - **Button**: permette di visualizzare i dettagli del bin$_G$.
      - **azione**: mostra i dettagli del bin$_G$ mediante il componente `binItemDetails`.

- **zoneItem**:
  - **descrizione**: componente dedicato alla visualizzazione di una zona;
  - **parametri**:
    - **zone**: oggetto `Zone` che rappresenta la zona da visualizzare.

- **informazioni visualizzate**:
  - **ID**: codice identificativo univoco della zona.
- **interazione con l'utente**$_G$:
  - **Button**: permette di visualizzare i dettagli della zona;
    - **azione**: mostra i dettagli della zona mediante il componente `zoneItemDetails`.
  - **Button**: permette di cancellare la zona.
    - **azione**: cancella la zona.

– **zoneItemDetails**:
- **descrizione**: componente dedicato alla visualizzazione delle informazioni dettagliate di una zona;
- **parametri**:
  - **zone**: oggetto `Zone` che rappresenta la zona di cui visualizzare i dettagli.
- **informazioni visualizzate**:
  - **ID**: codice identificativo univoco della zona;
  - **Lunghezza**: lunghezza della zona;
  - **Larghezza**: larghezza della zona;
  - **Altezza**: altezza della zona;
  - **Orientamento**: orientamento della zona;
  - **Data-Table**: visualizzazione dei bin$_G$ presenti all'interno della zona mediante il componente `bin_columns`.
- **interazione con l'utente**$_G$:
  - **Button**: permette di modificare la zona;
    - **azione**: modifica la zona mediante il componente `zoneCreationFrame`.
  - **Button**: permette di cancellare la zona.
    - **azione**: cancella la zona.

– **zonePanel**:
- **descrizione**: componente dedicato alla visualizzazione di tutte le zone presenti nel magazzino$_G$. Ogni zona è rappresentata da un `ZoneItem`;
- **interazione con l'utente**$_G$:
  - **SearchBar**: permette di cercare una zona all'interno della lista;
    - **azione**: filtra le zone presenti in base alla stringa inserita;
    - **tipologia di ricerca**:
      - **ID**: ricerca per codice identificativo univoco.
  - **Button**: permette di aggiungere una nuova zona.
    - **azione**: aggiunge una nuova zona mediante il componente `zoneCreationFrame`.

- **Panel**:
  - **descrizione**: componente dedicato alla visualizzazione di un pannello laterale. Utilizzato dai componenti `zonePanel`, `productsPanel`, `ordersPanel` e `settingsPanel`.

4.4.2 Three.js$_G$

Gli oggetti di modello vengono passati come parametri ai componenti Three.js$_G$ per la loro creazione, rendendo dunque indipendente l'oggetto di business indipendente dalla sua rappresentazione grafica.

Mediante l'utilizzo dei framework @react$_G$-three/fiber e @react$_G$-three/drei, è possibile creare elementi 3D all'interno di un'applicazione React$_G$ trattando gli elementi come componenti.

- **Floor**:
  - **descrizione**: componente dedicato alla visualizzazione del piano dell'ambiente di lavoro;
  - **parametri**:
    - Floor: oggetto `Floor` che rappresenta il piano dell'ambiente di lavoro.

- **Bin3D**:
  - **descrizione**: componente dedicato alla visualizzazione di un bin$_G$ all'interno dell'ambiente di lavoro;
  - **parametri**:
    - Bin$_G$: oggetto `Bin`$_G$ che rappresenta il bin$_G$ da visualizzare.
  - **visualizzazione**:
    - **box**: rappresenta il bin$_G$ all'interno dell'ambiente di lavoro. Il colore del box varia a seconda dei seguenti fattori:
      - **blu**: bin$_G$ occupato contenente un prodotto;
      - **nero**: bin$_G$ occupato non contenente un prodotto;
      - **rosso**: bin$_G$ selezionato;
      - **giallo**: bin$_G$ di destinazione di un ordine;
      - **verde**: bin$_G$ di partenza di un ordine.
  - **interazione con l'utente$_G$**:
    - **onDoubleClick**: permette di selezionare il bin$_G$ visualizzando i dettagli mediante i componenti `binItemDetails` e `productItemDetails` (se contenente un prodotto);
    - **onDrag**: permette di spostare il bin$_G$ all'interno dell'ambiente di lavoro, in modo da generare un nuovo ordine di spostamento. L'evento onDrag è disponibile solo per i bin3D che rappresentano un bin$_G$ il cui *state* ottenibile dal metodo `getBinState` risulta **Idle** e con un prodotto al suo interno.

- **Zone3D**:
  - **descrizione**: componente dedicato alla visualizzazione di una zona all'interno dell'ambiente di lavoro. I bin$_G$ al suo interno sono generati e visualizzati mediante il componente `Bin3D`;
  - **parametri**:
    - Zone: oggetto `Zone` che rappresenta la zona da visualizzare.
  - **interazione con l'utente$_G$**:
    - **onDoubleClick**: permette di selezionare la zona visualizzando i dettagli mediante il componente `zoneItemDetails`;
    - **onDrag**: permette di spostare la zona all'interno dell'ambiente di lavoro.

- **Warehouse**:
  - **descrizione**: componente dedicato alla visualizzazione dell'ambiente di lavoro e del Canvas di rendering;
  - **interazione con l'utente**$_G$:
    - **Zone3D**: permette di visualizzare le zone presenti all'interno dell'ambiente di lavoro;
    - **Floor**: permette di visualizzare il piano dell'ambiente di lavoro;
    - **Grid**: permette di visualizzare la griglia di riferimento per il collocamento degli scaffali nell'ambiente di lavoro;
    - **GridModeSelector**: permette di selezionare la dimensione della griglia di riferimento;
    - **CameraControls ed ExtendedCameraControls**: permette di controllare la camera all'interno dell'ambiente di lavoro;
    - **KeyboardControls**: permette di controllare la camera mediante tastiera.

- **ExtendedCameraControls**:
  - **descrizione**: componente dedicato al controllo avanzato della camera all'interno dell'ambiente di lavoro;
  - **interazione con l'utente**$_G$:
    - **onKeyDown**: permette di reimpostare la posizione della camera mediante il tasto "R";
    - **useFrame**: intercettati i tasti "W", "A", "S", "D" e "Shift" permette di spostare la camera all'interno dell'ambiente di lavoro.

- **GridModeSelector**:
  - **descrizione**: componente dedicato alla selezione della dimensione della griglia di riferimento;
  - **interazione con l'utente**$_G$:
    - **ToggleGruop**: permette di selezionare la dimensione della griglia di riferimento;
      - **opzioni**:
        - **0**: griglia non visibile;
        - **0.1**: griglia con celle di 0.1 unità, rappesentanti 10cm;
        - **0.5**: griglia con celle di 0.5 unità, rappesentanti 50cm;
        - **1**: griglia con celle di 1 unità, rappesentanti 1m.

5 Architettura di deployment

Nel contesto del progetto didattico$_G$, l'adozione di un'architettura monolitica è stata determinata valutando fattori strategici in termini di risorse, tempi ed esperienza del gruppo.

5.1 Analisi dell'architettura monolitica

L'architettura monolitica rappresenta un modello di progettazione architetturale in cui tutte le funzionalità e i componenti del prodotto software risiedono in un unico sistema autocontenuto e indipendente.

Nel contesto del progetto didattico$_G$, l'architettura monolitica è stata scelta tenendo conto dei seguenti vantaggi e considerazioni:

- **Gestione di una singola unità**: l'intero set di funzionalità risiede in unico sistema. Questo approccio semplifica lo sviluppo e la manutenzione del codice, rimuovendo la complessità associata alla comunicazione tra servizi e alla gestione dei dati distribuiti in un'architettura a microservizi;
- **Sviluppo e testing**$_G$: data l'esperienza limitata del gruppo, l'adozione di un'architettura monolitica permette di concentrarsi sullo sviluppo e il testing$_G$ del prodotto in modo più tempestivo e diretto, e di focalizzarsi maggiormente sulla realizzazione delle funzionalità chiave del progetto rispetto alla gestione delle complessità di un'architettura distribuita;
- **Aggiornamenti e manutenzione**: la gestione di un unico sistema semplifica la manutenzione e i numerosi aggiornamenti del codice durante lo sviluppo dettati dalla limitata esperienza del gruppo. Tuttavia, apportare modifiche consistenti può richiedere aggiornamenti in parti diverse del sistema.
- **Deployment**: l'architettura monolitica semplifica il processo di deployment, dovendo gestire un unico sistema.

5.2 Deployment con Docker$_G$

Il processo di deployment del software è gestito mediante l'utilizzo di Docker$_G$ e Docker Compose$_G$. La scelta di utilizzare Docker$_G$ è stata determinata dai seguenti fattori:

- **Isolamento**: Docker$_G$ permette di eseguire processi informatici in ambienti isolati chiamati container. Questo garantisce che il software funzioni in modo coerente e affidabile, creando una base comune di sviluppo tra i Programmatori e garantendo che il prodotto software sia isolato dall'ambiente di esecuzione;
- **Portabilità**: Docker$_G$ semplifica il processo di deployment del prodotto in diversi ambienti, garantendo che il software funzioni in modo coerente e affidabile su qualsiasi piattaforma;
- **Semplicità di installazione e avvio**: impostata la configurazione dei `Dockerfile` e di Docker Compose$_G$, l'avvio e la gestione dei container è standardizzata.

In conclusione, l'architettura monolitica si allinea perfettamente con le esigenze e le limitazioni del progetto, rendendola una scelta ragionata e valida. Questa scelta permette di concentrarsi sulle priorità chiave: sviluppare un prodotto funzionante e di alta qualità in tempi ragionevoli, pur mantenendo la flessibilità di apportare modifiche in base alle esigenze emerse.

5.2.1 Ambiente Docker$_G$

I container Docker$_G$ sono organizzati e gestiti mediante Docker Compose$_G$.

Sono presenti due container:

- **app**: container contenente l'applicazione web;
  - **Immagine**: `node`$_G$`:20-alpine`;
  - **Porta**: `3000`;

  - **Dipendenza**: `postgres`$_G$;
  - **Rete**: `webnet`.
- **postgres**$_G$: container contenente il database$_G$ PostgreSQL$_G$;
  - **Immagine**: `postgres`$_G$`:16.2`;
  - **Porta**: `5432`;
  - **Dipendenza**: nessuna;
  - **Rete**: `webnet`.

I container sono all'interno della stessa rete `webnet` che utilizza il driver di rete `bridge`.

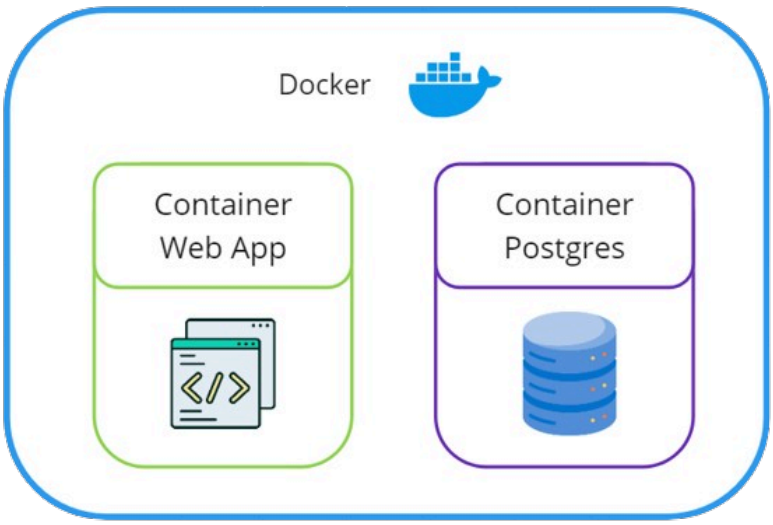

Figura 7: Ambiente Docker$_G$

5.3 Deployment e installazione

5.3.1 Download di WMS3

Modalità	Source
Link diretto (consigliato)	https://github$_G$.com/Error-418-SWE$_G$/WMS3/releases
Shell	`git`$_G$ `clone git`$_G$`@github`$_G$`.com:Error-418-SWE`$_G$`/WMS3.git`$_G$
Shell	`git`$_G$ `clone https://github`$_G$`.com/Error-418-SWE`$_G$`/WMS3.git`$_G$

Tabella 4: Modalità di download WMS3

5.3.2 Avvio di WMS3

Per avviare la web app è necessario collocarsi all'interno della cartella scaricata al passaggio *Download di WMS3* (Sezione 5.3.1) ed eseguire il comando

```
docker compose_G up
```

Completato l'avvio dei container, la web app sarà disponibile all'indirizzo

http://localhost:3000/

5.3.3 Terminare l'esecuzione

Per terminare l'esecuzione è necessario collocarsi nella cartella scaricata al passaggio *Download di WMS3* (Sezione 5.3.1) ed eseguire il comando

```
docker compose_G down
```

6 Database$_G$

In questa sezione viene presentato lo schema della base di dati realizzata con PostgreSQL$_G$.

Esso è cosi composto:

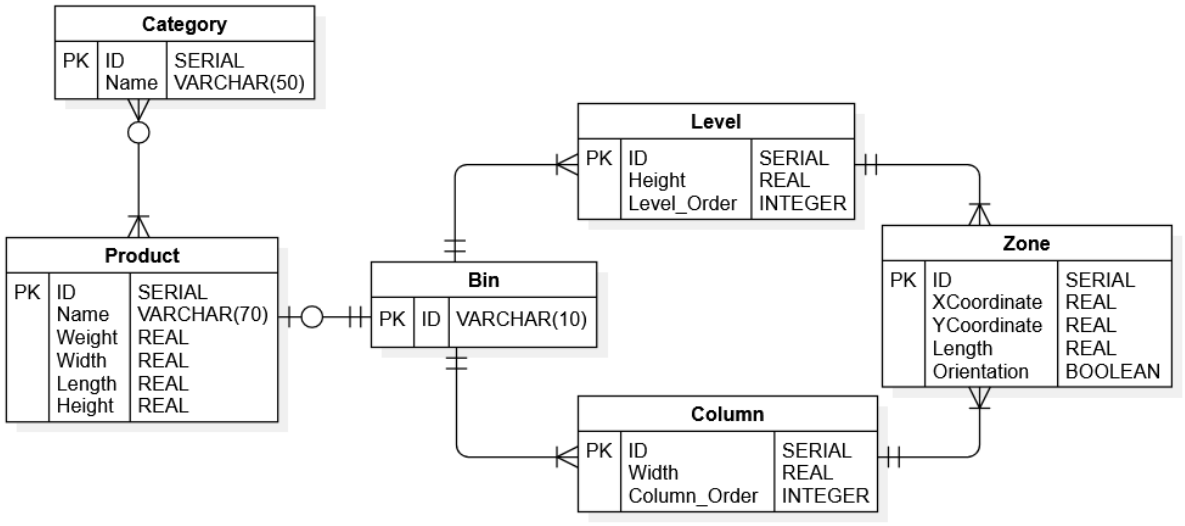

Figura 8: Schema ER del Database$_G$.

6.1 Entità

Il database$_G$ è composto da 6 entità:

- **Product**: rappresenta un prodotto presente all'interno del magazzino$_G$. Composto da:
  - ID: identificativo univoco e seriale di un prodotto;
  - Name: nome del prodotto;
  - Weight: peso del prodotto;
  - Width: larghezza del prodotto;
  - Length: lunghezza del prodotto;
  - Height: altezza del prodotto.
- **Category**: rappresenta la categoria merceologica di appartenenza del prodotto. Composto da:
  - ID: identificativo univoco e seriale di una categoria;
  - Name: nome della categoria.

- **Bin**$_G$: rappresenta uno spazio del magazzino$_G$ in cui è possibile inserire un prodotto. Composto da:
  - ID: identificativo univoco di un bin$_G$, esso è cosi composto:

    ```
    ID zona + _ + Column_Order + _ + Level_Order"
    ```

- **Level**: rappresenta un ripiano$_G$ dello scaffale$_G$. Composto da:
  - ID: identificativo univoco e seriale di un ripiano$_G$;
  - Height: altezza del ripiano$_G$;
  - Level_order: valore incrementale che rappresenta la posizione del ripiano$_G$ all'interno di uno scaffale$_G$. Se il suo valore è 0 allora esso rappresenta una zona a terra.
- **Column**: rappresenta una colonna dello scaffale$_G$. Composto da:
  - ID: identificativo univoco e seriale di una colonna;
  - Width: larghezza della colonna;
  - Column_order: valore incrementale che rappresenta la posizione della colonna all'interno di uno scaffale$_G$.
- **Zone**: rappresenta una zona del piano del magazzino$_G$. Essa può essere sia uno scaffale$_G$ che una zona a terra. Composto da:
  - ID: identificativo univoco e seriale di una zona;
  - XCoordinate: coordinata orizzontale della zona;
  - YCoordinate: coordinata verticale della zona;
  - Length: lunghezza della zona;
  - Orientation: orientamento della zona.

6.2 Relazioni

All'interno del database$_G$ le relazioni fra le differenti entità sono del tipo:

- **Zero..One to One** per quanto riguarda le entità:
  - Product e Bin$_G$.
- **One to Many** per quanto riguarda le entità:
  - Bin$_G$ e Level;
  - Bin$_G$ e Column;
  - Level e Zone;
  - Column e Zone.
- **Many to Zero..Many** per quanto riguarda le entità:
  - Product e Category.

6.2.1 Interrogazione del database$_G$

Il database$_G$ viene utilizzato dall'applicazione per il caricamento, il posizionamento e la visualizzazione dei prodotti all'interno del magazzino$_G$. In nessun caso il database$_G$ verrà modificato dall'applicazione.

7 Requisiti soddisfatti

Di seguito vengono riportati i requisti funzionali e di qualità soddisfatti dall'applicazione.

Per una visione più completa sui requisiti si rimanda al documento *Analisi dei Requisiti*$_G$ *v2.0.1*.

7.1 Requisiti funzionali soddisfatti

Codice	Descrizione	Stato
FM–1	L'utente$_G$ deve poter configurare un ambiente 3D all'avvio della sessione d'uso	Soddisfatto
FD–2	L'utente$_G$ deve avere la possibilità di scegliere tra diverse modalità di configurazione del magazzino$_G$	Soddisfatto
FM–3	Deve essere fornita una modalità di configurazione dell'ambiente 3D per la rappresentazione di un magazzino$_G$ con pianta rettangolare	Soddisfatto
FD–4	Deve essere fornita una modalità di configurazione dell'ambiente 3D per la rappresentazione di un magazzino$_G$ con planimetria importata da file SVG$_G$	Soddisfatto
FM–5	L'utente$_G$ deve poter indicare la larghezza della planimetria rettangolare	Soddisfatto
FM–6	L'utente$_G$ deve poter indicare la lunghezza della planimetria rettangolare	Soddisfatto
FM–7	L'utente$_G$ deve visualizzare un errore se la larghezza indicata non è positiva ($\leq$0)	Soddisfatto
FM–8	L'utente$_G$ deve visualizzare un errore se la lunghezza indicata non è positiva ($\leq$0)	Soddisfatto
FD–9	L'utente$_G$ deve poter caricare un file SVG$_G$ da usare come planimetria qualora abbia scelto di definire la planimetria a partire da un file SVG$_G$	Soddisfatto
FD–10	Il file SVG$_G$ deve essere sanificato prima dell'importazione	Soddisfatto
FD–11	Il file SVG$_G$ deve contenere almeno un elemento grafico tra path, rect, circle, ellipse, line, polyline, polygon, text, g per essere considerato valido	Non soddisfatto
FD–12	L'utente$_G$ deve ricevere un messaggio di errore qualora avesse caricato un file SVG$_G$ privo di elementi grafici (path, rect, circle, ellipse, line, polyline, polygon, text, g)	Non soddisfatto
FD–13	Il file SVG$_G$ deve essere validato	Soddisfatto
FD–14	L'utente$_G$ deve ricevere un messaggio di errore qualora avesse caricato un file SVG$_G$ non valido o corrotto	Soddisfatto
FD–15	L'utente$_G$ che abbia scelto la modalità di configurazione a partire da un file SVG$_G$, deve poter indicare il solo lato maggiore del magazzino$_G$ per configurare la planimetria	Soddisfatto

FD–16	Il sistema deve determinare il valore del lato minore a partire dal rapporto di aspetto del file SVG$_G$ e dai dati forniti dall'utente$_G$	Soddisfatto
FD–17	L'utente$_G$ deve visualizzare un errore se il valore indicato come lato maggiore non è positivo ($\leq$0)	Soddisfatto
FM–18	L'utente$_G$ deve poter riconfigurare la planimetria dell'ambiente 3D corrente	Soddisfatto
FM–19	A seguito della riconfigurazione della planimetria, le modifiche a zone, bin$_G$ e prodotti non devono subire variazioni	Soddisfatto
FO–20	L'utente$_G$ deve poter visualizzare un'anteprima delle modifiche alla planimetria prima di confermare l'operazione	Soddisfatto
FM–21	L'utente$_G$ deve poter ridefinire la larghezza dell'ambiente 3D corrente	Soddisfatto
FM–22	L'utente$_G$ deve poter ridefinire la lunghezza dell'ambiente 3D corrente	Soddisfatto
FD–23	L'utente$_G$ che abbia configurato un ambiente 3D a partire da file SVG$_G$ non può definire un valore di lunghezza inferiore a quello corrente	Soddisfatto
FD–24	L'utente$_G$ che abbia configurato un ambiente 3D a partire da file SVG$_G$ non può definire un valore di larghezza inferiore a quello corrente	Soddisfatto
FM–25	L'utente$_G$ deve visualizzare un errore se il nuovo valore di larghezza indicato non è positivo ($\leq$0)	Soddisfatto
FM–26	L'utente$_G$ deve visualizzare un errore se il nuovo valore di lunghezza indicato non è positivo ($\leq$0)	Soddisfatto
FD–27	L'utente$_G$ deve poter disporre di una griglia di aggancio come aiuto al posizionamento delle zone nell'ambiente 3D	Soddisfatto
FD–28	Il passo della griglia deve essere configurabile	Soddisfatto
FD–29	L'utente$_G$ deve poter disattivare la griglia di posizionamento	Soddisfatto
FD–30	La griglia deve essere configurabile durante le normali operazioni sull'ambiente 3D, non esclusivamente durante la configurazione dell'ambiente	Soddisfatto
FD–31	Se il passo di griglia non è nullo, il collocamento delle zone deve agganciarsi ad essa	Soddisfatto
FD–32	L'utente$_G$ deve poter importare le zone da un database$_G$	Soddisfatto
FD–33	L'utente$_G$ deve poter importare le zone da un database$_G$ durante la fase di configurazione dell'ambiente 3D	Soddisfatto
FD–34	Le zone importate devono essere collocate automaticamente nell'ambiente 3D, nella posizione descritta dal database$_G$	Soddisfatto

FD–35	I bin$_G$ delle zone devono essere importati contestualmente all'importazione delle zone	Soddisfatto
FD–36	L'importazione delle zone può avvenire solo se l'utente$_G$ ha configurato un ambiente 3D a partire da file SVG$_G$	Soddisfatto
FD–37	L'utente$_G$ deve visualizzare un messaggio di errore nel caso l'importazione non dovesse andare a buon fine	Soddisfatto
FD–38	L'utente$_G$ deve poter importare i prodotti da database$_G$	Soddisfatto
FD–39	Quando l'utente$_G$ importa zone e prodotti da un database$_G$, i prodotti devono essere collocati nei rispettivi bin$_G$ di appartenenza	Soddisfatto
FM–40	L'utente$_G$ deve poter alterare il proprio POV sull'ambiente 3D	Soddisfatto
FM–41	L'utente$_G$ deve poter ruotare il proprio POV attorno all'asse longitudinale	Soddisfatto
FM–42	L'utente$_G$ deve poter traslare il proprio POV lungo l'asse orizzontale	Soddisfatto
FM–43	L'utente$_G$ deve poter effettuare zoom$_G$-in	Soddisfatto
FM–44	L'utente$_G$ deve poter effettuare zoom$_G$-out	Soddisfatto
FM–45	L'utente$_G$ deve poter configurare un nuovo ambiente 3D	Soddisfatto
FM–46	La configurazione di un nuovo ambiente 3D deve cancellare tutti i dati della sessione corrente	Soddisfatto
FM–47	Il sistema non deve offrire la persistenza dei dati importati	Soddisfatto
FM–48	Il sistema non deve offrire la persistenza dei dati generati durante la sessione corrente	Soddisfatto
FM–49	La lista delle movimentazioni di prodotti richieste durante la sessione corrente deve essere scartata contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–50	Le aggiunte alle zone devono essere scartate contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–51	Le modifiche alle zone devono essere scartate contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–52	Le cancellazioni delle zone devono essere scartate contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–53	La configurazione della planimetria deve essere scartata contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–54	Le informazioni sui prodotti devono essere scartate contestualmente alla riconfigurazione dell'ambiente 3D	Soddisfatto
FM–55	L'utente$_G$ deve poter creare nuove zone	Soddisfatto
FD–56	L'utente$_G$ deve poter indicare una sequenza numerica come codice identificativo delle nuove zone create	Soddisfatto

FD–57	L'utente$_G$ deve visualizzare un errore qualora avesse indicato un codice identificativo già in uso	Soddisfatto
FM–58	L'utente$_G$ deve indicare la lunghezza della nuova zona da creare	Soddisfatto
FM–59	L'utente$_G$ deve visualizzare un errore se la lunghezza indicata non è positiva ($\leq$0)	Soddisfatto
FD–60	L'utente$_G$ deve poter scegliere tra "NS" e "WE" come orientamento della zona da creare	Soddisfatto
FM–61	L'utente$_G$ deve indicare il numero di colonne della nuova zona	Soddisfatto
FM–62	Una zona deve contenere almeno 1 colonna	Soddisfatto
FD–63	L'identificazione delle colonne deve avvenire tramite lettere crescenti in senso lessicografico a partire da "A"	Soddisfatto
FM–64	L'utente$_G$ deve visualizzare un errore se il numero di colonne della nuova zona non è almeno pari a 1	Soddisfatto
FD–65	L'utente$_G$ deve poter personalizzare la larghezza delle colonne della nuova zona	Soddisfatto
FD–66	L'utente$_G$ deve poter suddividere la larghezza della nuova zona in colonne di equa larghezza	Soddisfatto
FD–67	L'utente$_G$ deve indicare la larghezza complessiva della nuova zona, qualora avesse richiesto la suddivisione della stessa in colonne di equa larghezza	Soddisfatto
FD–68	L'utente$_G$ deve poter suddividere la larghezza della nuova zona in colonne di larghezza specifica	Soddisfatto
FD–69	L'utente$_G$ deve poter indicare la larghezza di ciascuna colonna, qualora avesse richiesto la suddivisione della nuova zona in colonne di larghezza specifica	Soddisfatto
FD–70	Il sistema deve determinare il valore della larghezza della zona dalla somma delle larghezze delle singole colonne	Soddisfatto
FD–71	L'utente$_G$ deve visualizzare un errore se la larghezza indicata per la singola colonna non è positiva ($\leq$0)	Soddisfatto
FM–72	L'utente$_G$ deve poter personalizzare il numero di livelli della nuova zona da creare	Soddisfatto
FM–73	L'utente$_G$ deve visualizzare un errore se il numero di livelli della nuova zona non è almeno pari a 1	Soddisfatto
FM–74	L'utente$_G$ deve poter personalizzare l'altezza dei singoli livelli della zona	Soddisfatto
FM–75	Una zona deve contenere almeno 1 livello	Soddisfatto
FM–76	La numerazione dei livelli deve partire da 0 ("piano terra")	Soddisfatto
FM–77	Il sistema deve determinare il valore dell'altezza della zona dalla somma delle altezze dei singoli livelli	Soddisfatto

FM–78	L'utente$_G$ deve visualizzare un errore se l'altezza indicata per il singolo livello non è positiva ($\leq$0)	Soddisfatto
FM–79	L'utente$_G$ deve poter modificare una zona già creata	Soddisfatto
FM–80	L'utente$_G$ deve poter modificare una zona importata da database$_G$	Soddisfatto
FM–81	L'utente$_G$ deve poter rimuovere una singola colonna, purché l'operazione non elimini una colonna con almeno un bin$_G$ occupato	Soddisfatto
FM–82	L'utente$_G$ deve poter rimuovere una singola colonna, purché l'operazione non elimini una colonna con indice inferiore all'indice di una colonna con almeno un bin$_G$ occupato	Soddisfatto
FM–83	L'utente$_G$ deve visualizzare un errore se l'operazione di rimozione di una colonna è impossibile per i vincoli individuati	Soddisfatto
FM–84	L'utente$_G$ deve poter rimuovere un singolo livello, purché l'operazione non elimini un livello con almeno un bin$_G$ occupato	Soddisfatto
FM–85	L'utente$_G$ deve poter rimuovere un singolo livello, purché l'operazione non elimini un livello con indice inferiore all'indice di un livello con almeno un bin$_G$ occupato	Soddisfatto
FM–86	L'utente$_G$ deve visualizzare un errore se l'operazione di rimozione di un livello è impossibile per i vincoli individuati	Soddisfatto
FM–87	L'operazione di creazione di una nuova zona è da ritenersi conclusa solo con il corretto collocamento della stessa nell'ambiente 3D	Soddisfatto
FM–88	L'operazione di modifica di una zona è da ritenersi conclusa solo con il corretto collocamento della stessa nell'ambiente 3D	Soddisfatto
FM–89	L'utente$_G$ deve poter eliminare qualsiasi zona	Soddisfatto
FM–90	I prodotti collocati in una zona rimossa non devono essere cancellati	Soddisfatto
FM–91	L'utente$_G$ deve visualizzare un messaggio di avviso prima di procedere con l'eliminazione di una zona	Soddisfatto
FM–92	L'utente$_G$ deve poter ispezionare una zona a partire dall'ambiente 3D	Soddisfatto
FM–93	L'utente$_G$ deve poter visualizzare l'ID della zona ispezionata	Soddisfatto
FM–94	L'utente$_G$ deve poter visualizzare la larghezza della zona ispezionata	Soddisfatto
FM–95	L'utente$_G$ deve poter visualizzare la lunghezza della zona ispezionata	Soddisfatto
FM–96	L'utente$_G$ deve poter visualizzare l'altezza della zona ispezionata	Soddisfatto
FM–97	La zona ispezionata deve essere evidenziata graficamente nell'ambiente 3D	Soddisfatto

FM–98	L'utente$_G$ deve poter visualizzare la lista dei bin$_G$ inclusi nella zona ispezionata	Soddisfatto
FM–99	L'utente$_G$ deve poter visualizzare l'ID dei bin$_G$ inclusi nella zona ispezionata	Soddisfatto
FM–100	L'utente$_G$ deve poter visualizzare lo stato di occupazione dei bin $_G$ inclusi nella zona ispezionata	Soddisfatto
FM–101	L'utente$_G$ deve poter collocare una zona creata nello spazio 3D	Soddisfatto
FM–102	L'utente$_G$ deve poter collocare una zona modificata nello spazio 3D	Soddisfatto
FM–103	Il sistema deve evidenziare graficamente una zona in una posizione non occupabile	Soddisfatto
FM–104	Il sistema deve impedire il collocamento di una zona su una posizione non occupabile	Soddisfatto
FM–105	Il sistema deve impedire il collocamento di una zona su di un'altra, ovvero deve impedire la compenetrazione tra zone	Soddisfatto
FM–106	Il sistema deve impedire il collocamento di una zona al di fuori del perimetro dell'ambiente 3D	Soddisfatto
FM–107	L'utente$_G$ deve poter visualizzare la lista delle zone contenute nell'ambiente 3D	Soddisfatto
FM–108	L'utente$_G$ deve poter visualizzare l'ID delle zone incluse nella lista	Soddisfatto
FD–109	L'utente$_G$ deve poter cercare le zone in base all'ID	Soddisfatto
FD–110	Le zone che rispondono ai criteri di ricerca devono essere evidenziate graficamente	Soddisfatto
FM–111	L'utente$_G$ deve poter ispezionare un bin$_G$ a partire dall'ambiente 3D	Soddisfatto
FM–112	L'utente$_G$ deve poter visualizzare l'ID del bin$_G$ ispezionato	Soddisfatto
FM–113	L'utente$_G$ deve poter visualizzare la lunghezza del bin$_G$ ispezionato	Soddisfatto
FM–114	L'utente$_G$ deve poter visualizzare la larghezza del bin$_G$ ispezionato	Soddisfatto
FM–115	L'utente$_G$ deve poter visualizzare l'altezza del bin$_G$ ispezionato	Soddisfatto
FM–116	Il bin$_G$ ispezionato deve essere evidenziato graficamente	Soddisfatto
FM–117	L'utente$_G$ deve poter visualizzare le informazioni associate al prodotto eventualmente contenuto nel bin$_G$	Soddisfatto
FM–118	Ogni bin$_G$ può contenere al massimo 1 prodotto	Soddisfatto
FD–119	L'utente$_G$ può richiedere lo spostamento del POV sulla zona ispezionata	Non soddisfatto
FD–120	L'utente$_G$ può richiedere lo spostamento del POV sul bin$_G$ ispezionato	Non soddisfatto

FD–121	L'utente$_G$ deve poter visualizzare le informazioni associate ad un prodotto importato da database$_G$	Soddisfatto
FD–122	L'utente$_G$ deve poter visualizzare l'ID del prodotto ispezionato	Soddisfatto
FD–123	L'utente$_G$ deve poter visualizzare il nome del prodotto ispezionato	Soddisfatto
FD–124	L'utente$_G$ deve poter visualizzare la categoria del prodotto ispezionato	Soddisfatto
FD–125	L'utente$_G$ deve poter visualizzare la larghezza del prodotto ispezionato	Soddisfatto
FD–126	L'utente$_G$ deve poter visualizzare la lunghezza del prodotto ispezionato	Soddisfatto
FD–127	L'utente$_G$ deve poter visualizzare l'altezza del prodotto ispezionato	Soddisfatto
FD–128	L'utente$_G$ deve poter visualizzare il peso del prodotto ispezionato	Soddisfatto
FD–129	L'utente$_G$ deve poter visualizzare la lista dei prodotti importati da database$_G$	Soddisfatto
FD–130	L'utente$_G$ deve poter distinguere tra prodotti collocati in un bin $_G$ e non collocati	Soddisfatto
FD–131	L'utente$_G$ deve poter visualizzare la lista dei prodotti collocati	Soddisfatto
FD–132	L'utente$_G$ deve poter visualizzare la lista dei prodotti non collocati	Soddisfatto
FD–133	L'utente$_G$ deve poter visualizzare il nome del prodotto nella lista dei prodotti (collocati e non)	Soddisfatto
FD–134	L'utente$_G$ deve poter visualizzare l'ID del prodotto nella lista dei prodotti (collocati e non)	Soddisfatto
FD–135	L'utente$_G$ deve poter visualizzare la categoria del prodotto nella lista dei prodotti (collocati e non)	Soddisfatto
FD–136	L'utente$_G$ deve poter visualizzare l'ID della zona di appartenenza di un prodotto nella lista dei prodotti collocati	Non soddisfatto
FD–137	L'utente$_G$ deve poter visualizzare l'ID del bin$_G$ di appartenenza di un prodotto nella lista dei prodotti collocati	Non soddisfatto
FD–138	L'utente$_G$ deve poter filtrare la lista dei prodotti collocati in base all'ID	Non soddisfatto
FD–139	L'utente$_G$ deve poter filtrare la lista dei prodotti non collocati in base all'ID	Non soddisfatto
FD–140	L'utente$_G$ deve poter filtrare la lista dei prodotti collocati in base al nome	Soddisfatto
FD–141	L'utente$_G$ deve poter filtrare la lista dei prodotti non collocati in base al nome	Soddisfatto

FD–142	L'utente$_G$ deve poter filtrare la lista dei prodotti collocati in base alla categoria	Soddisfatto
FD–143	L'utente$_G$ deve poter filtrare la lista dei prodotti non collocati in base alla categoria	Soddisfatto
FD–144	I filtri di ricerca devono essere mutuamente esclusivi	Soddisfatto
FD–145	L'utente$_G$ deve poter inserire un ordine di movimentazione di un prodotto dalla lista dei prodotti ad un bin$_G$	Non soddisfatto
FM–146	L'utente$_G$ deve poter inserire un ordine di movimentazione di un prodotto da un bin$_G$ ad un altro tramite drag and drop	Soddisfatto
FM–147	Ciascun ordine di movimentazione deve inviare una richiesta alla API$_G$ per la convalida dell'operazione	Soddisfatto
FM–148	La API$_G$ deve ricevere almeno l'ID del bin$_G$ di destinazione	Soddisfatto
FM–149	La API$_G$ deve rispondere con stato HTTP 200 se l'operazione è stata convalidata	Soddisfatto
FM–150	La API$_G$ deve rispondere con stato HTTP 4XX se l'operazione è stata rifiutata	Soddisfatto
FM–151	La API$_G$ convalida o rifiuta le operazioni in maniera casuale	Soddisfatto
FM–152	Il sistema deve impedire l'inserimento di un ordine di movimentazione verso un bin$_G$ occupato	Soddisfatto
FM–153	L'utente$_G$ deve visualizzare l'esito dell'operazione di convalida da parte dell'API$_G$	Soddisfatto
FM–154	L'utente$_G$ deve visualizzare un errore di connessione se l'accesso all'API$_G$ non è possibile	Soddisfatto
FD–155	Quando un ordine di movimentazione è convalidato, esso viene inserito in una cronologia delle operazioni accessibile dall'utente$_G$	Soddisfatto
FM–156	Quando un ordine di movimentazione è rifiutato, il prodotto oggetto dell'operazione ritorna nella posizione di partenza	Soddisfatto
FD–157	L'utente$_G$ deve poter visualizzare la cronologia degli ordini di movimentazione convalidati	Soddisfatto
FD–158	L'utente$_G$ deve poter visualizzare l'ID del bin$_G$ di partenza degli ordini di movimentazione convalidati se l'operazione è partita da un bin$_G$	Soddisfatto
FD–159	L'utente$_G$ deve poter visualizzare l'ID del bin$_G$ di destinazione dell'ordine di movimentazione convalidato	Soddisfatto
FD–160	L'utente$_G$ deve poter visualizzare l'ID del prodotto oggetto dell'ordine di movimentazione convalidato	Soddisfatto
FD–161	L'utente$_G$ deve poter visualizzare il nome del prodotto oggetto dell'ordine di movimentazione convalidato	Soddisfatto

FD–162	L'utente$_G$ deve poter visualizzare l'ID del bin$_G$ di partenza degli ordini di movimentazione convalidati se l'operazione è partita da un bin$_G$	Soddisfatto
FD–163	L'utente$_G$ può poter ispezionare un singolo ordine di movimentazione convalidato	Soddisfatto
FM–164	Il sistema deve evidenziare graficamente il bin$_G$ di destinazione dell'ordine di movimentazione ispezionato	Soddisfatto
FM–165	Se l'ordine di movimentazione ispezionato si è originato da un bin$_G$, il sistema deve evidenziare graficamente il bin$_G$ di partenza nell'ambiente 3D	Soddisfatto

Tabella 5: Requisiti funzionali

7.2 Requisiti di qualità soddisfatti

Codice	Descrizione	Stato
QM–1	Deve essere rispettato quanto previsto dal documento *Norme di Progetto*$_G$ *v1.30.1*	Soddisfatto
QM–2	Deve essere rispettato quanto previsto dal documento *Piano di Qualifica*$_G$ *v1.5.0*	Soddisfatto
QM–3	Il codice sorgente deve essere consegnato utilizzando un repository$_G$ GitHub$_G$ pubblico	Soddisfatto
QM–4	Devono essere consegnati i diagrammi UML$_G$ degli UC$_G$	Soddisfatto
QM–5	Deve essere consegnata la lista dei bug$_G$ risolti	Soddisfatto
QM–6	Deve essere fornito un manuale d'uso per l'utente$_G$	Soddisfatto
QO–7	Deve essere consegnato lo schema del DB$_G$	Soddisfatto
QO–8	Deve essere consegnata la documentazione$_G$ delle API$_G$ realizzate	Soddisfatto
QM–9	Deve essere fornita la documentazione$_G$ dell'architettura del prodotto	Soddisfatto

Tabella 6: Requisiti di qualità