

IronWorks

Utility per la Costruzione di Software Robusto



Swear on Code

swearoncode@gmail.com

Manuale Utente

Versione	2.0.0
Redattori	Stefano Nordio, Francesco Sacchetto Sharon Della Libera, Anna Poletti
Verificatori	Antonio Moz
Responsabili	Mirko Gibin
Uso	Esterno
Distribuzione	Gruppo Swear on Code Prof. Tullio Vardanega Prof. Riccardo Cardin Gregorio Piccoli, Zucchetti S.p.A.

Descrizione

Questo documento contiene il Manuale Utente per il progetto **IronWorks** del gruppo Swear on Code.

Registro delle modifiche

Versione	Data	Autori	Ruolo	Descrizione
2.0.0	2018/08/22	Mirko Gibin	Responsabile	Approvazione
1.1.0	2018/08/10	Antonio Moz	Verificatore	Verifica
1.0.3	2018/08/03	Sharon Della Libera	Programmatore	Modifiche generali ed estensione §3
1.0.2	2018/08/02	Anna Poletti	Programmatore	Estensione §4.2
1.0.1	2018/07/30	Sharon Della Libera	Programmatore	Estensione §3 con procedura d'installazione
1.0.0	2018/07/12	Sharon Della Libera	Responsabile	Approvazione
0.1.0	2018/07/12	Antonio Moz	Verificatore	Verifica
0.0.7	2018/07/11	Stefano Nordio	Programmatore	Stesura §A
0.0.6	2018/07/11	Francesco Sacchetto	Programmatore	Stesura §4
0.0.5	2018/07/10	Stefano Nordio	Programmatore	Inserimento immagini in §3
0.0.4	2018/07/10	Francesco Sacchetto	Programmatore	Stesura §3
0.0.3	2018/07/09	Stefano Nordio	Programmatore	Stesura §2
0.0.2	2018/07/06	Stefano Nordio	Programmatore	Stesura §1
0.0.1	2018/07/05	Francesco Sacchetto	Programmatore	Creazione del documento

Tabella 1: Storico versioni del documento

Indice

1	Introduzione	1
1.1	Scopo del Documento	1
1.2	Scopo del Prodotto	1
1.3	Ambiguità	1
2	Requisiti di Sistema	2
2.1	Requisiti Software	2
2.2	Requisiti Hardware	2
3	Manuale d'Uso	3
3.1	Accesso all'Applicazione	3
3.1.1	Utilizzo in Locale	3
3.1.2	Utilizzo Tramite la Rete	3
3.2	Installazione in Locale dell'Applicazione	4
3.2.1	Prerequisiti	4
3.2.2	Installazione IronWorks	4
3.2.2.1	Installare le Dipendenze	4
3.2.2.2	Avvio IronWorks	4
3.3	Pagina Iniziale	5
3.3.1	Creazione Nuovo Progetto	6
3.3.2	Caricamento Progetto Esistente	7
3.4	Pagina dell'Editor	8
3.4.1	Layout	8
3.4.1.1	Sezione Superiore	8
3.4.1.2	Sezione Centrale	8
3.4.1.3	Sezione Inferiore	9
3.4.2	Tornare alla Pagina Iniziale	9
3.4.3	Salvare il Progetto	9
3.4.4	Scaricare il Codice	10
3.5	Utilizzare l'Editor	11
3.5.1	Aggiungere un Elemento Actor, Boundary o Control	11
3.5.2	Aggiungere un Elemento Entity	12
3.5.3	Modificare un Elemento Actor, Boundary o Control	13
3.5.4	Modificare un Elemento Entity	14
3.5.4.1	Aggiungere un Attributo ad un Elemento Entity	15
3.5.5	Collegare Due Elementi	17
3.6	Utilizzo del Codice Generato	18
3.6.1	Contenuto del File Zip Scaricato	18
3.6.2	Esecuzione dello Script Sql	18
3.6.3	Gestione del Database Tramite Driver JDBC	18
3.6.4	Gestione del Database Tramite Framework Hibernate ORM	19
3.6.5	Compilazione ed Esecuzione	19
3.6.6	Contenuto Tabelle SQL	20

3.6.7	Contenuto Classi Java	21
3.6.8	Contenuto Classi DAO	21
3.6.9	Ulteriori Informazioni	22
4	Risoluzione dei Problemi	23
4.1	Accesso all'Applicazione Non Disponibile	23
4.2	Segnalazione di Bug	23
A	Glossario	24

Elenco delle figure

1	Pagina iniziale	5
2	Nuovo progetto	6
3	Caricamento progetto esistente	7
4	Pagina dell'editor	8
5	Pulsante per mostrare il menù degli elementi	9
6	Aggiunta di un elemento Actor, Boundary o Control	11
7	Aggiunta di un elemento Entity	12
8	Scelta iniziale del modificatore d'accesso per l'entità	12
9	Modifica di un elemento Actor, Boundary o Control	13
10	Modifica di un elemento Entity	14
11	Aggiunta di un attributo	15
12	Tipo dell'attributo	16
13	Linea di associazione	17

1 Introduzione

1.1 Scopo del Documento

Questo documento ha l'obiettivo di descrivere le funzionalità e le modalità di utilizzo dell'applicazione IronWorks. Esso rappresenta sia una guida, sia un riferimento completo per l'utilizzo del prodotto.

1.2 Scopo del Prodotto

Lo scopo del prodotto è quello di fornire un *editor_G* di *diagrammi di robustezza_G* che permetta all'*utente_G* di progettare l'*architettura_G* del proprio software secondo lo *standard_G UML_G*.

A partire dal diagramma disegnato è possibile generare automaticamente il corrispondente codice *Java_G* delle *entità_G* persistenti.

Per ogni *entità_G* viene inoltre generato il codice *SQL_G* per permettere la creazione delle tabelle nel *database_G* relazionale, mentre nei *file_G DAO_G* vengono implementate tutte le operazioni *CRUD_G* per gestire la persistenza dei dati.

È prevista anche la generazione del codice per implementare l'interazione con il database attraverso il middleware *Hibernate_G*.

1.3 Ambiguità

Nella realizzazione di questo documento, vengono sottintesi molti aspetti e tecnicismi riguardanti la programmazione. La motivazione risiede nel fatto che questo prodotto mira ad inserirsi nelle prime fasi della progettazione software. Pertanto, gli utilizzatori saranno *utenti_G* con una conoscenza almeno basilare della programmazione.

Al fine di dipanare qualsiasi dubbio o ambiguità relativa al linguaggio impiegato viene fornito il Glossario in appendice A. Esso contiene le definizioni di tutti i termini scritti in corsivo e marcati con una 'G' pedice.

2 Requisiti di Sistema

IronWorks è una *applicazione web_G* destinata al funzionamento su dispositivi desktop. Non è presente una versione per dispositivi mobile.

2.1 Requisiti Software

Il funzionamento di questo prodotto è legato all'utilizzo di un browser. Di seguito viene fornito un breve elenco con le versioni di tutti i browser sui quali viene garantito il funzionamento del nostro prodotto:

- *Google Chrome_G* versione 67 o superiore;
- *Firefox Quantum_G* versione 61 o superiore;
- *Safari_G* versione 11 o superiore;
- *Microsoft Edge_G* versione 42 o superiore.

Per il corretto funzionamento dell'applicazione è necessario che *Javascript_G* sia abilitato nei browser usati.

2.2 Requisiti Hardware

Non vi sono particolari requisiti hardware per il funzionamento del prodotto, se non quelli relativi al browser utilizzato.

3 Manuale d'Uso

3.1 Accesso all'Applicazione

L'applicazione può essere utilizzata sia in locale che tramite la rete.

3.1.1 Utilizzo in Locale

Per poter accedere in locale all'applicazione IronWorks è necessario:

- Aver installato precedentemente nella propria macchina IronWorks. (Le istruzioni sono presenti nella sezione 3.2);
- Collegarsi all'indirizzo <http://localhost:3000/> tramite uno dei browser supportati.

3.1.2 Utilizzo Tramite la Rete

Per poter accedere tramite la rete all'applicazione Iron Works è necessario recarsi all'indirizzo <http://46.101.244.120> utilizzando uno dei browser supportati.

3.2 Installazione in Locale dell'Applicazione

3.2.1 Prerequisiti

L'applicazione necessita del runtime *JavaScript_G* "*Node.js_G*" ed il package manager "*npm_G*" installati nel proprio sistema. La versione minima d'installazione è 8.11.1 LTS.

La versione per il proprio sistema operativo è scaricabile all'indirizzo:

<https://nodejs.org/it/download> .

3.2.2 Installazione IronWorks

3.2.2.1 Installare le Dipendenze

Posizionandosi all'interno della directory "BackEnd", aprire qui un terminale e lanciare il comando "npm install".

Questo comando si occuperà di installare tutte le dipendenze necessarie al funzionamento dell'applicazione.

3.2.2.2 Avvio IronWorks

Per avviare l'applicazione occorre aprire nuovamente un terminale all'interno della directory "BackEnd" e lanciare il comando "npm start".

Una volta eseguito il comando, per accedere all'applicazione, lasciare in esecuzione il terminale ed aprire uno dei browser supportati.

Fatto questo, recarsi all'indirizzo <http://localhost:3000> .

Nel caso in cui l'utente decida di aprire il progetto con un IDE deve configurare manualmente la "run" per l'esecuzione. Affinchè tutto funzioni regolarmente è necessario impostare la directory "BackEnd" come "working directory" e "bin/www.js" per il file di esecuzione.

3.3 Pagina Iniziale

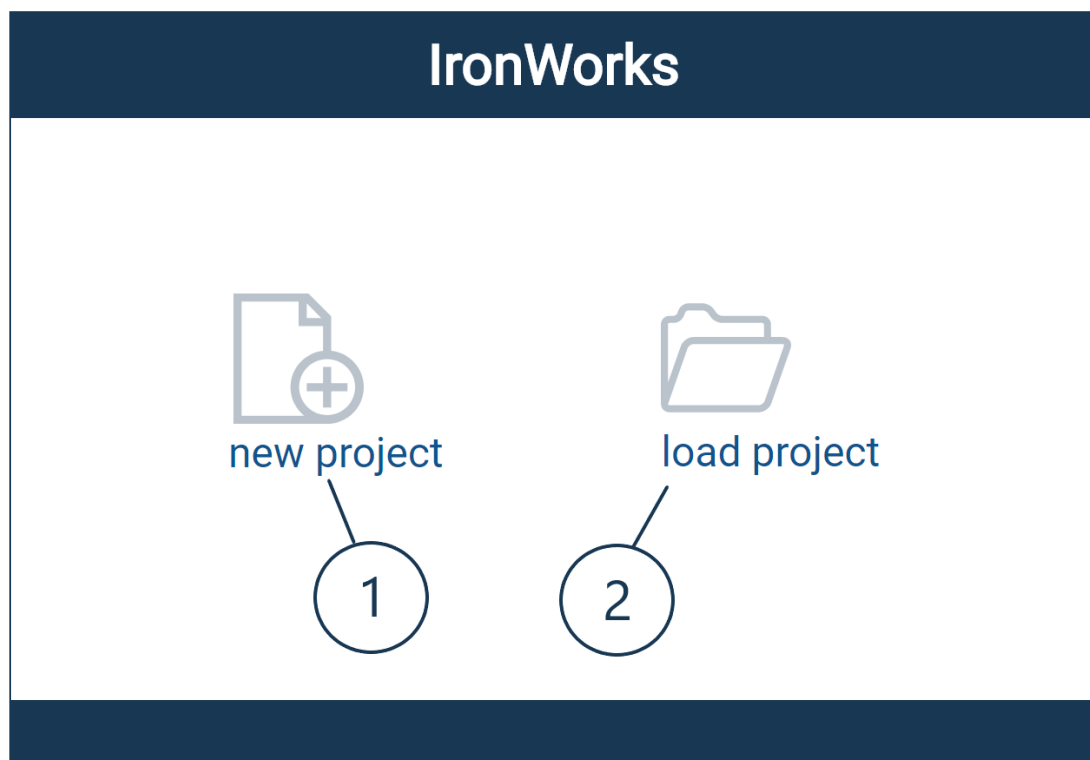


Figura 1: Pagina iniziale

Aprendo l'applicazione, l'*utente_G* si trova nella pagina iniziale. Qui ha la possibilità di scegliere tra due opzioni:

- Creare un nuovo progetto (1 in figura 1);
- Caricare dal filesystem un progetto esistente (2 in figura 1).

3.3.1 Creazione Nuovo Progetto

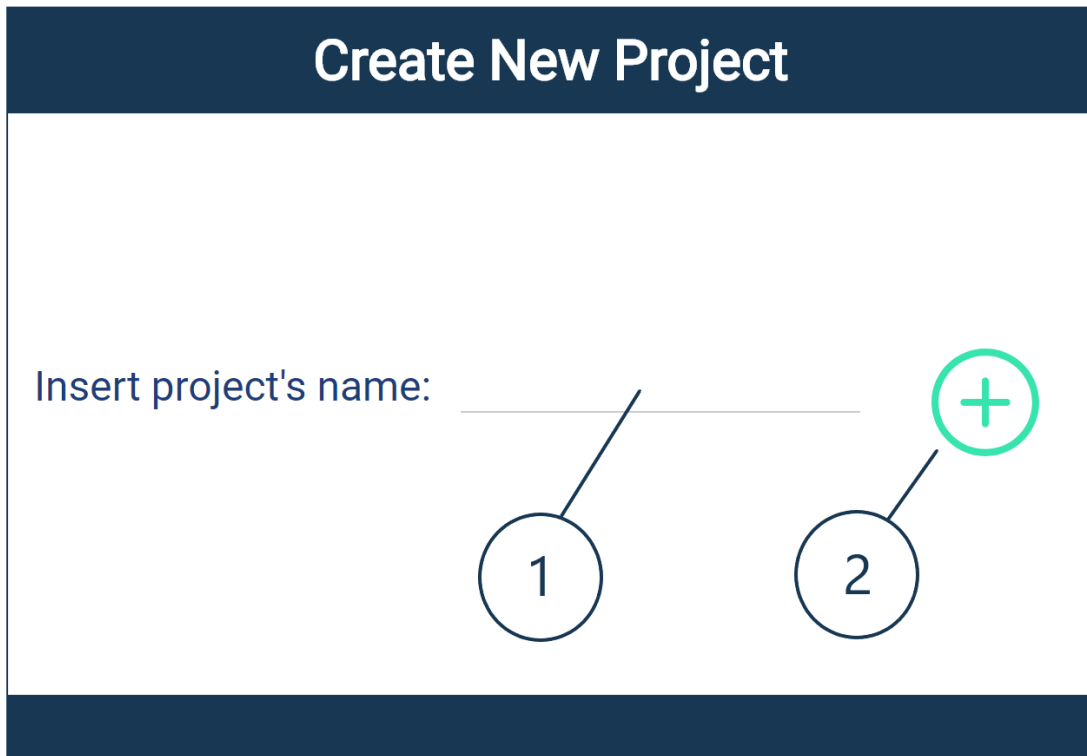


Figura 2: Nuovo progetto

Una volta scelto di creare un nuovo progetto, l'*utente_G* viene indirizzato alla pagina di creazione.

Cliccando nell'area evidenziata (1 in figura 2) è possibile inserire il nome che si vuole assegnare al progetto. Una volta inserito il nome, cliccare sul segno "+" (2 in figura 2) per confermare. Si viene successivamente indirizzati alla pagina dell'*editor_G*.

Attenzione: Non è possibile modificare il nome del progetto una volta creato.

3.3.2 Caricamento Progetto Esistente

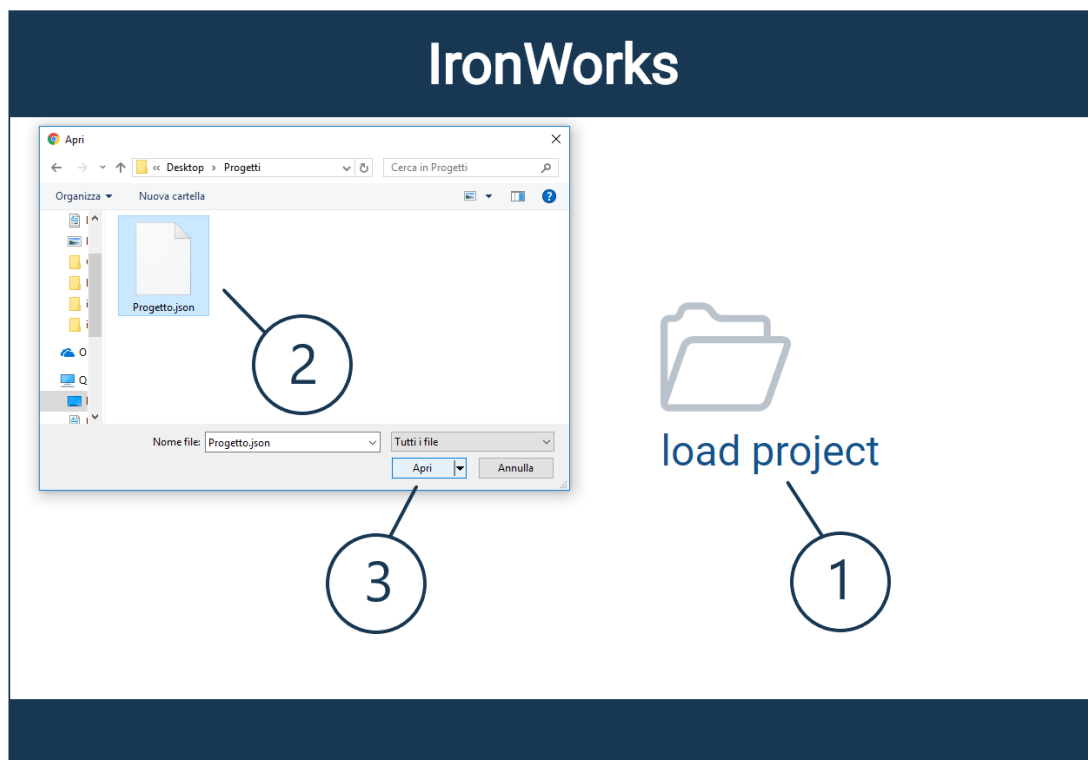


Figura 3: Caricamento progetto esistente

Se invece si desidera continuare a lavorare su un progetto già esistente, è possibile caricarne le informazioni.

Cliccando sull'apposito pulsante (1 in figura 3), compare una finestra da cui è possibile caricare il progetto. Una volta trovato il *file_G* nel filesystem locale è necessario:

- Selezionare il *file_G*, cliccandolo col tasto sinistro del mouse (2 in figura 3);
- Cliccare su "Apri" (3 in figura 3).

Una volta fatto questo, si viene indirizzati alla pagina dell'*editor_G* con le informazioni (nome del progetto, elementi del diagramma, dati *entità_G* etc.) del progetto caricato, già inserite.

Attenzione: Assicurarsi di caricare un *file_G* creato precedentemente da questa applicazione, con estensione ".json".

3.4 Pagina dell'Editor

Una volta scelto se creare un nuovo progetto o caricarne uno già esistente, l'*utente_G* viene indirizzato alla pagina in figura 4.

3.4.1 Layout

Il layout della pagina è organizzato in tre sezioni.

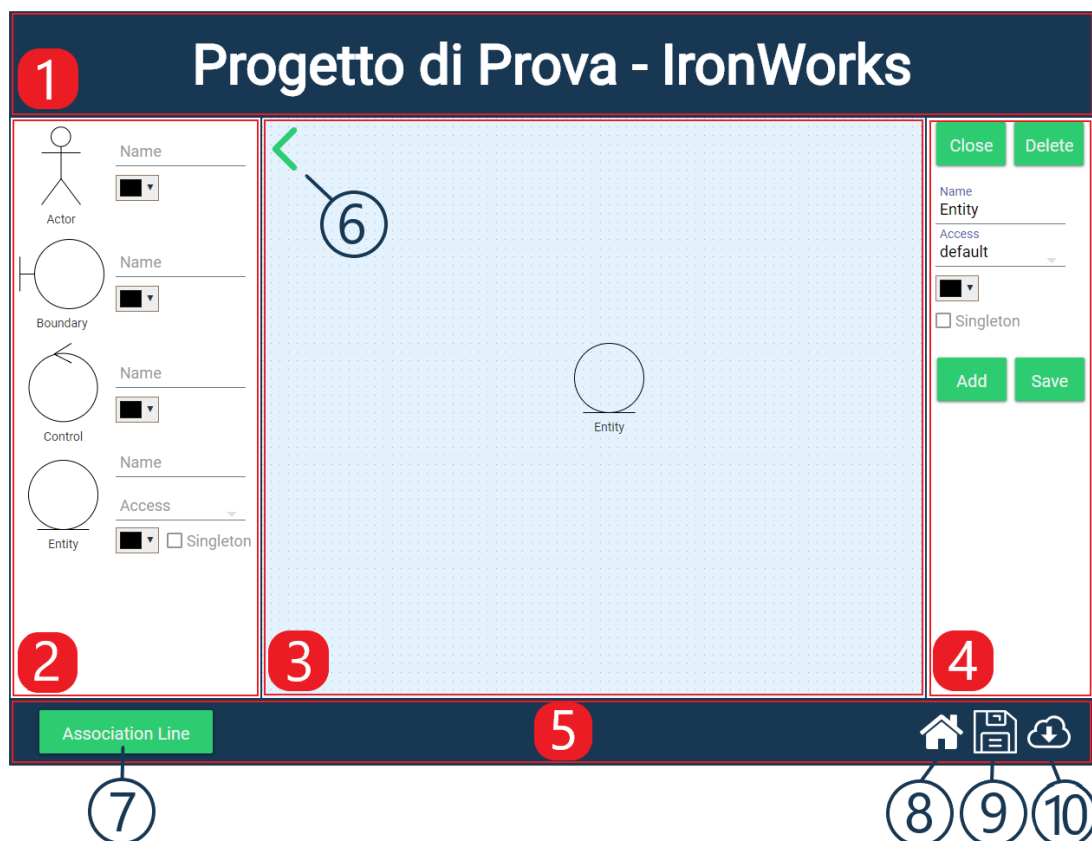


Figura 4: Pagina dell'editor

3.4.1.1 Sezione Superiore

Nella barra orizzontale in alto (1 in figura 4) è presente il nome del progetto scelto in fase di creazione.

3.4.1.2 Sezione Centrale

Nella fascia centrale vi è l'*editor_G* vero e proprio, organizzato in 3 sottoparti:

- Una barra verticale sulla sinistra (2 in figura 4) contenente gli elementi trascinabili del diagramma. Tale barra compare soltanto cliccando sul pulsante a forma di freccia (1 in figura 5);



Figura 5: Pulsante per mostrare il menù degli elementi

- Il paper in cui viene realizzato il diagramma (3 in figura 4);
- Una barra verticale sulla destra con tutte le informazioni modificabili dell'elemento (4 in figura 4). Tale barra compare soltanto cliccando due volte sull'elemento desiderato;
- Un pulsante a forma di freccia (6 in figura 4) per nascondere il menù degli elementi (2 in figura 4).

3.4.1.3 Sezione Inferiore

Nella barra orizzontale situata a fondo pagina (5 in figura 4) sono presenti dei pulsanti che permettono di:

- Creare una linea di associazione tra due elementi (7 in figura 4);
- Tornare alla pagina iniziale (8 in figura 4);
- Salvare il progetto con le informazioni correnti (9 in figura 4);
- Generare il codice relativo al diagramma e scaricarlo in *file_G* zip (10 in figura 4).

3.4.2 Tornare alla Pagina Iniziale

Cliccando sul pulsante a forma di casa (8 in figura 4) è possibile tornare alla pagina iniziale. Una volta cliccato appare un messaggio che richiede la conferma dell'azione. Se si procede tutto il lavoro non salvato viene perduto.

3.4.3 Salvare il Progetto

Cliccando sull'icona a forma di floppy disk (9 in figura 4) viene eseguito istantaneamente il download di un archivio "nomeprogetto.json". Tale *file_G* contiene tutte le informazioni salvate nel diagramma dell'*editor_G*. L'*utente_G* ha quindi la possibilità di salvare in qualsiasi momento l'avanzamento del proprio progetto.

3.4.4 Scaricare il Codice

Cliccando sul pulsante con la freccia verso il basso (10 in figura 4) è possibile generare e scaricare il codice relativo alle *entità_G* presenti nel diagramma. Al click viene eseguito istantaneamente il download di un archivio "nomeprogetto.zip" avente all'interno due tipi di *file_G*:

- *File_G* con estensione ".sql", contenente lo script *SQL_G* per la creazione delle tabelle all'interno del *database_G*;
- *File_G* con estensione ".java". Ne viene generato uno per ogni *entità_G* del diagramma. All'interno di ognuno è presente la classe *Java_G* contenente le informazioni (e i metodi per usufruirne) dell'*entità_G*.

3.5 Utilizzare l'Editor

3.5.1 Aggiungere un Elemento Actor, Boundary o Control

Per aggiungere un elemento al diagramma è sufficiente trascinare, con il tasto sinistro del mouse, una delle icone presenti nella barra verticale (1 in figura 6) nel paper dell'*editor_G* (2 in figura 6) e rilasciare nella posizione desiderata.

Questa funzionalità di Drag&Drop è attiva per ogni elemento del diagramma e funziona anche su elementi già presenti nel paper.

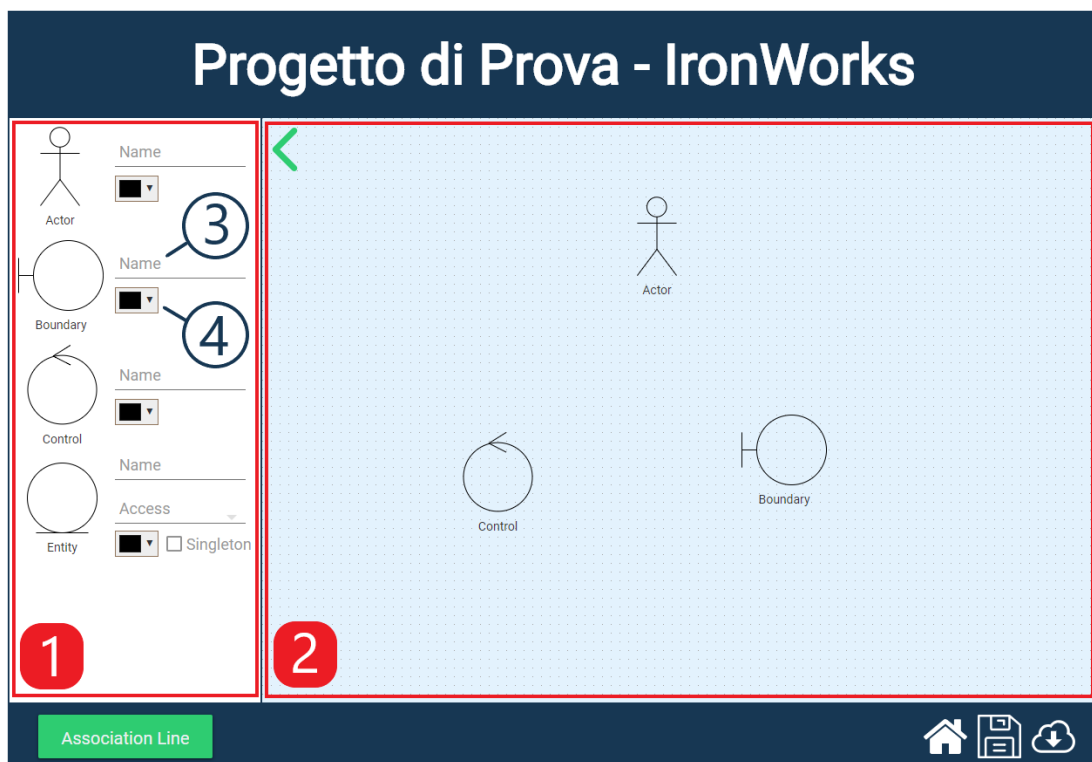


Figura 6: Aggiunta di un elemento Actor, Boundary o Control

E' possibile assegnare determinate caratteristiche all'elemento prima ancora del suo inserimento nel paper. Accanto all'icona dell'elemento sono presenti due zone cliccabili:

- La casella di testo "Name" (3 in figura 6), dove è possibile assegnare un nome personalizzato all'elemento;
- Il selettore del colore (di default nero) (4 in figura 6) con il quale è possibile assegnare un colore all'elemento. Dopo il click si apre un pop-up in cui basterà selezionare il colore desiderato e premere "choose".

Una volta inseriti il nome e il colore desiderati, basta trascinare l'elemento sul paper dell'*editor_G* per rendere effettive le modifiche.

3.5.2 Aggiungere un Elemento Entity

L'aggiunta di un elemento *Entity_G* segue quanto spiegato nella sezione 3.5.1.

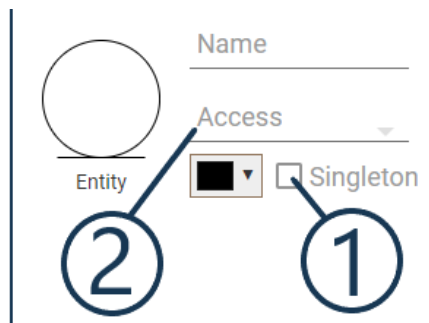


Figura 7: Aggiunta di un elemento Entity

Oltre alle precedenti, vi sono due funzionalità aggiuntive:

- Cliccando sulla checkbox "*Singleton_G*" (1 in figura 7) è possibile scegliere se l'*entità_G* da creare sia considerata come *Singleton_G*;
- Cliccando su "Access" (2 in figura 7) è possibile scegliere quale modificatore d'accesso assegnare all'*entità_G*. Le scelte possibili sono indicate nella figura 8.

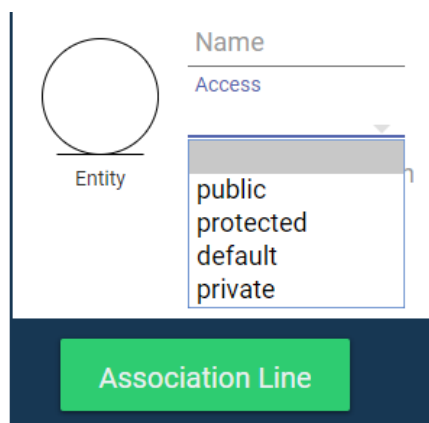


Figura 8: Scelta iniziale del modificatore d'accesso per l'entità

3.5.3 Modificare un Elemento Actor, Boundary o Control

Dopo aver inserito un elemento è possibile modificarlo. Cliccando due volte col tasto sinistro del mouse su di esso compare sul lato destro l'apposita sezione di modifica.

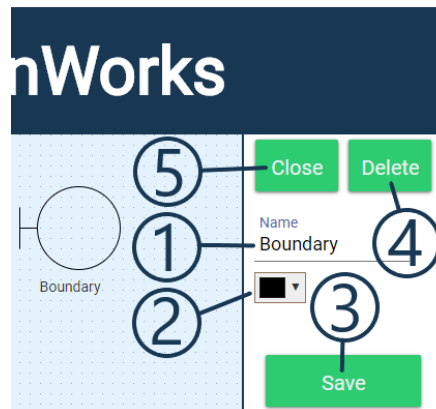


Figura 9: Modifica di un elemento Actor, Boundary o Control

In questo pannello sono presenti quattro zone cliccabili:

- La casella di testo con la scritta "Name" (1 in figura 9) permette di modificare il nome dell'elemento;
- Il selettore del colore (2 in figura 9) permette di modificare il colore dell'elemento;
- Il pulsante "Save" (3 in figura 9) permette di confermare le modifiche;
- Il pulsante "Delete" (4 in figura 9) permette di eliminare l'elemento; una volta cliccato, un pop-up chiede di confermare la scelta;
- Il pulsante "Close" (5 in figura 9) permette la chiusura di questo pannello.

Attenzione: Le modifiche non confermate vengono perse.

3.5.4 Modificare un Elemento Entity

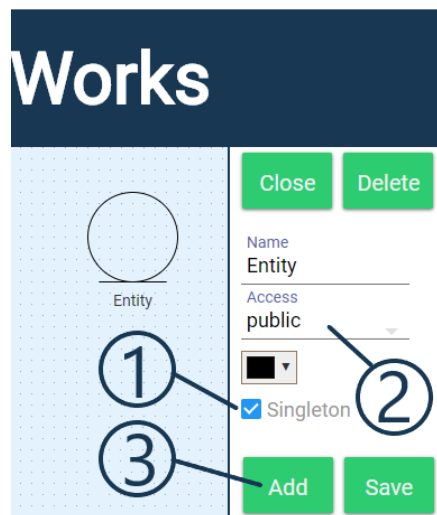


Figura 10: Modifica di un elemento Entity

Per gli elementi $Entity_G$, oltre a quanto descritto in figura 3.5.3, si può:

- Cliccare sulla checkbox " $Singleton_G$ " (1 in figura 10): è possibile scegliere se l' $entità_G$ debba essere modificata in $Singleton_G$ o meno;
- Cliccare su "Access" (2 in figura 10): è possibile scegliere nuovamente quale modificatore d'accesso (gli stessi visualizzabili nella figura 8) assegnare all' $entità_G$;
- Cliccare su "Add" (3 in figura 10): dopo il click compare una nuova sezione di questo pannello, come visibile in figura 11, che permette l'aggiunta di un attributo.

3.5.4.1 Aggiungere un Attributo ad un Elemento Entity

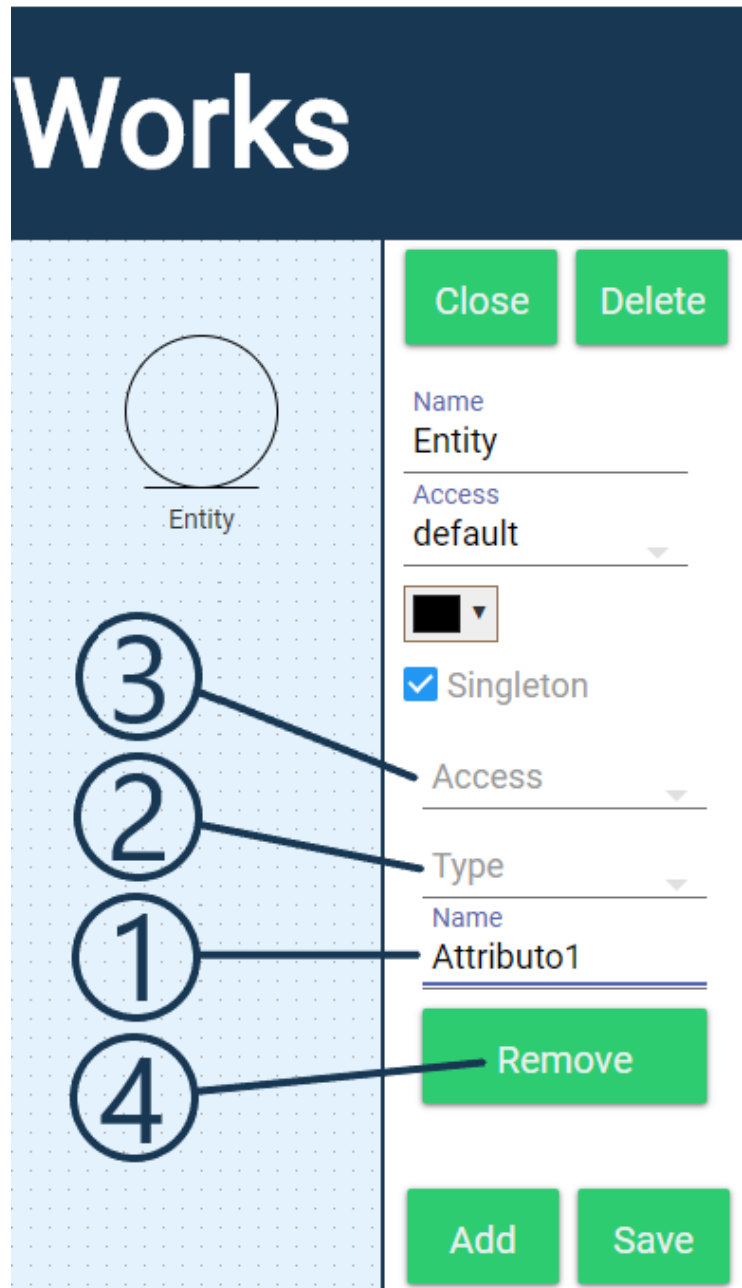


Figura 11: Aggiunta di un attributo

Nella nuova sezione del pannello, sono presenti quattro zone cliccabili:

- "Name" (1 in figura 11) in cui inserire il nome dell'attributo;
- "Type" (2 in figura 11) in cui inserire il tipo dell'attributo, scegliendo dalla lista che compare in seguito al click (figura 12);

- "Access" (3 in figura 11) in cui inserire il modificatore d'accesso, scegliendo dalla lista che compare in seguito al click (gli stessi della figura 8);
- "Remove" (4 in figura 11) per rimuovere l'attributo corrente.

Nella figura 12 è possibile vedere i tipi d'attributo attualmente supportati.

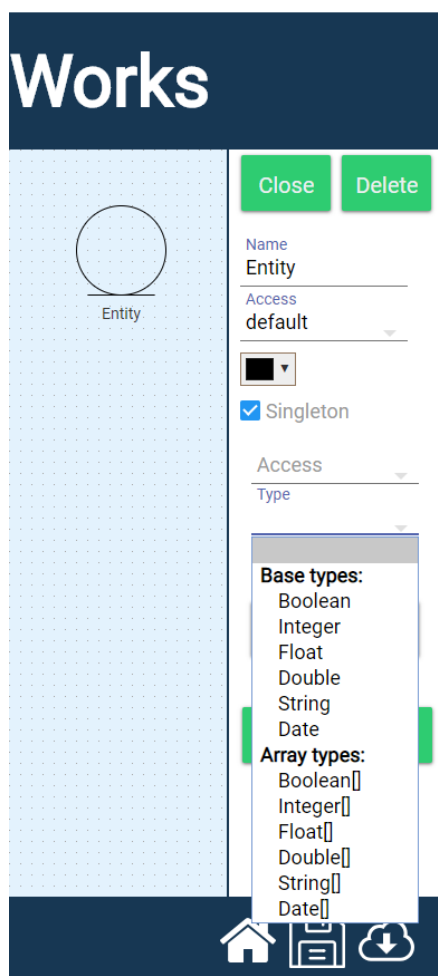


Figura 12: Tipo dell'attributo

3.5.5 Collegare Due Elementi

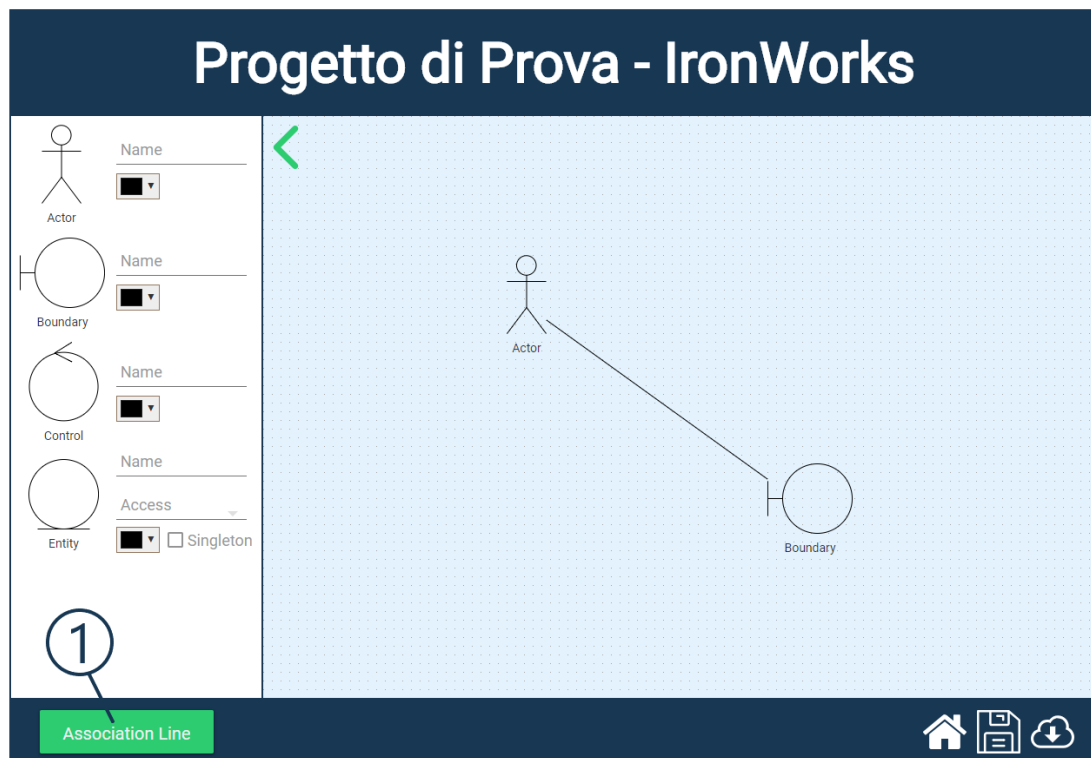


Figura 13: Linea di associazione

Per collegare due elementi del diagramma è necessario cliccare sul pulsante "Association Line" (1 in figura 13) e in seguito:

- Cliccare una volta soltanto, con il tasto sinistro del mouse, sull'elemento da cui far partire la linea di associazione;
- Cliccare sul secondo elemento, destinazione della linea di associazione.

Una volta cliccati i due elementi, compare in automatico la linea di associazione.

Attenzione: Rispettare sempre le regole del *diagramma di robustezza_G* nell'associazione di due elementi.

3.6 Utilizzo del Codice Generato

3.6.1 Contenuto del File Zip Scaricato

La zip scaricata al suo interno contiene:

- **scriptTables.sql**: script SQL che genera le tabelle e i trigger necessari per la gestione del database;
- **config.properties**: file di configurazione contenente le informazioni necessarie all'applicazione Java per connettersi al database utilizzando unicamente il driver JDBC;
- **hibernate.cfg.xml**: file di configurazione contenente le informazioni necessarie all'applicazione Java per connettersi al database attraverso il framework Hibernate ORM;
- **app.java**: file contenente unicamente una classe App con il metodo main, modificabile per eseguire operazioni;
- **cartella hibernate**: contiene tutti i file ".java" necessari a gestire il database attraverso Hibernate ORM;
- **cartella jdbc**: contiene tutti i file ".java" necessari a gestire il database attraverso il solo utilizzo del driver JDBC.

3.6.2 Esecuzione dello Script Sql

Lo script SQL è stato scritto seguendo la sintassi MySQL e testato per funzionare in un database MySQL con storage engine InnoDB.

Una volta creato il database che si desidera popolare, è sufficiente eseguire lo script per creare tabelle e trigger: da linea di comando MySQL digitare `source path-to/scriptTables.sql`. Le tabelle sono relative alle entità create nel diagramma, mentre i triggers permettono di mantenere e gestire eventuali singleton e gli indici delle tabelle relative agli array.

3.6.3 Gestione del Database Tramite Driver JDBC

I file ".java" relativi alla gestione del database tramite driver JDBC sono organizzati nei seguenti package:

- **jdbc.entities**: contiene semplici classi Java (anche dette POJO) con attributi e relativi metodi getter e setter;
- **jdbc.dao**: contiene le classi Java che interagiscono con il database (nello specifico, classi DAO);
- **jdbc.helpers**: contiene le classi necessarie a creare la connessione al database.

È innanzitutto necessario impostare url, username e password del database che si vuole gestire, modificando gli opportuni campi del file "config.properties".

Per funzionare, l'applicazione necessita della libreria "MySQL connector" reperibile al link <https://dev.mysql.com/downloads/connector/j/>.

Attenzione: in caso di segnalazione di errori relativi al Time-Zone durante la connessione al database, potrebbe essere necessario aggiungere in fondo all'url la seguente stringa:

```
?autoReconnect=true&useSSL=false&
useLegacyDatetimeCode=false&serverTimezone=UTC
```

3.6.4 Gestione del Database Tramite Framework Hibernate ORM

I file ".java" relativi alla gestione del database tramite framework sono organizzati nei seguenti package:

- **hibernate.entitites:** contiene le classi Java con attributi e relativi metodi get. Sugli attributi troviamo le annotazioni Hibernate che permettono il mapping tra questi e le tabelle SQL;
- **hibernate.dao:** contiene le classi Java che interagiscono con il database.

È innanzitutto necessario impostare url, username e password del database che si vuole gestire, modificando gli opportuni campi del file "hibernate.cfg.xml".

Per funzionare, l'applicazione necessita delle seguenti librerie:

- **libreria MySQL connector:** reperibile al link <https://dev.mysql.com/downloads/connector/j/>;
- **Framework Hibernate ORM:** reperibile al link <http://hibernate.org/orm/releases/5.3/>. Le librerie necessarie sono tutte all'interno della cartella "required".
- **Attenzione:** se si utilizza Java 9 o superiore è necessaria anche la libreria `java.xml.bind` reperibile al sito <https://jar-download.com/artifacts/javax.xml.bind>

Attenzione: in caso di segnalazione di errori relativi al Time-Zone durante la connessione al database, potrebbe essere necessario aggiungere in fondo all'url la seguente stringa:

```
?autoReconnect=true&useSSL=false&
useLegacyDatetimeCode=false&serverTimezone=UTC
```

3.6.5 Compilazione ed Esecuzione

Il codice Java generato è utilizzabile inserendo nel file `App.java`, contenente il metodo `main`, le istruzioni che si vogliono eseguire.

È necessario modificare gli import in base all'uso di solo JDBC o di Hibernate. Gli import sono già presenti nel file, è sufficiente commentare quelli non necessari.

Di seguito si riportano le istruzioni per compilare ed eseguire l'applicazione da terminale.

Assumendo che la cartella scaricata dall'editor si chiami `Project`:

- Da dentro la cartella `Project`, aprire il terminale e digitare:

- Linux o Mac: `find -name "*.java" > sources.txt`
- Windows: `dir /s /B *.java > sources.tx`

In questo modo viene creato il file `sources.txt` che contiene tutti i path delle classi che devono essere compilate;

- creare dentro Project una cartella rinominata ad esempio `out`;
- creare dentro Project una cartella rinominata ad esempio `javaliib`;
- copiare dentro `javaliib` tutte le librerie necessarie sopra indicate, sia per JDBC che per Hibernate;
- da terminale, dentro Project, eseguire: `textttjavac -cp "javaliib/*" -d ./out/ @sources.txt`
In questo modo vengono compilate le classi Java, e il bytecode sarà disponibile all'interno della cartella `out`;
- per eseguire l'applicazione da terminale, sempre dentro Project:
 - in Java 8:
 - * Linux o Mac: `java -cp ../javaliib/*: App`
 - * Windows: `java -cp ../javaliib/*; App`
 - in Java 9 o superiore:
 - * Linux o Mac: `java -cp ../javaliib/*: -add-modules java.xml.bind App`
 - * Windows: `java -cp ../javaliib/*; -add-modules java.xml.bind App`

3.6.6 Contenuto Tabelle SQL

Le tabelle SQL relative alle entità hanno la seguente struttura:

- Nome uguale alla variabile che contiene l'istanza;
- Campo `db_id` generato automaticamente, impostato come Primary Key con politica auto-increment;
- Lista di attributi aventi nomi uguali a quello inserito nel diagramma.

Le tabelle SQL relative agli array hanno la seguente struttura:

- Nome uguale al nome dell'array;
- Campo `db_id`, gestito da trigger o dai metodi Java per mantenere la posizione di un elemento all'interno dell'array;
- Campo chiave esterna avente nome uguale alla variabile che contiene l'istanza dell'entità, seguito da `"id"`;
- Campo `element`, contenente i valori dell'array.

La chiave primaria è composta dai campi `db_id` e dalla chiave esterna.

I trigger hanno lo scopo di mantenere le classi marcate come singleton e gestire gli indici degli array in modo da mantenere correlazione tra istanza, array e posizione degli elementi all'interno dell'array.

3.6.7 Contenuto Classi Java

Le classi POJO mettono a disposizione il costruttore di default senza parametri e i metodi `getter` e `setter` per gestire gli attributi dell'entità. Assumendo che il nome dell'attributo sia "Prova" i metodi sono invocabili con:

- `getProva()`, restituisce il valore di "Prova";
- `setProva(value)`, imposta il valore di "Prova" con l'elemento passato come parametro.

Viene automaticamente inserito un campo dati `Db_id` che ha lo scopo di mantenere la correlazione tra entità Java e i record nel database. Sono disponibili i relativi metodi `getter` e `setter` qualora lo si volesse manipolare.

Modifiche al campo `Db_id` non comporteranno modifiche al relativo campo nella tabella per non perdere la correlazione tra entità e oggetto.

Vengono forniti dei metodi di utilità di gestione degli array qualora questi fossero presenti. Assumendo che il nome dell'array sia "Example":

- `popFromExample()`, rimuove l'elemento in coda all'array e lo restituisce;
- `removeFromExample(index)`, rimuove l'elemento nella posizione indicata e lo restituisce;
- `pushToNomeExample(value)`, aggiunge in coda l'elemento passato come parametro e restituisce un `Boolean`;
- `editExampleElement(index, value)`, permette di sostituire l'elemento nella posizione indicata con l'elemento passato come parametro e restituisce un `Boolean`.

3.6.8 Contenuto Classi DAO

Le classi Dao offrono le seguenti funzionalità:

- `insert(Object)`, permette di inserire (o aggiornare qualora la chiave primaria del database fosse la stessa) l'oggetto `Object` nel database restituendo l'oggetto aggiunto;
- `read(id)`, fornito l'id dell'oggetto, il metodo restituisce un'istanza creata con i parametri prelevati dal database. In caso di errori viene restituito il valore `null`;
- `readAll()`, restituisce un `ArrayList` contenente tutti gli elementi letti dal database;
- `delete(Object)`, rimuove dal database l'elemento passato come parametro. Elimina anche tutti gli elementi dagli array e dalle relative tabelle in una politica CASCADE;

- `lastDb_id()`, restituisce l'indice più alto all'interno del database. Viene utilizzato per inizializzare il campo dati `Db_id` delle classi POJO durante l'inserimento nel database.

In aggiunta, le classi Dao che utilizzano unicamente il driver JDBC mettono a disposizione anche i seguenti metodi privati, utilizzati dai metodi appena descritti per mantenere la consistenza con gli array.

Assumendo il nome dell'array "Example" troviamo:

- `deleteFromExample(id, i, connection)`, rimuove dalla tabella tutti gli elementi collegati all'entità di indice `id` e con il loro `id >= i`. Viene utilizzato per eliminare gli elementi in eccesso dalla tabella qualora l'array venisse ridotto di dimensione;
- `deleteFromExample(id, connection)`, invoca il metodo `deleteFromExample(id, i, connection)`, passando come indice `i` il valore '0'. Lo scopo è di eliminare tutti gli elementi della tabella relativa all'array dell'entità di indice `id`. Viene invocato generalmente in seguito alla delete dell'entità, eliminando prima gli elementi della tabella relativa agli array in una politica CASCADE;
- `sqlAddExample(Object, connection)`, inserisce o aggiorna la tabella relativa agli array dell'elemento `Object`, sfruttando la connessione creata dal metodo chiamante;
- `tableExampleLength(id, connection)`, restituisce il numero di elementi all'interno della tabella. Viene generalmente utilizzato dai chiamanti per confrontare il numero di elementi nella tabella con quello negli array per scegliere l'operazione da effettuare per gestire la consistenza;
- `readExample(id, connection)`, restituisce un `ArrayList` contenente gli elementi collegati ad una entità, ricevendo il suo `id` e la connessione creata dal chiamante.

È messa a disposizione una classe `Database` che legge il file "config.properties" e che restituisce la connessione invocando il metodo `getConnection()`.

3.6.9 Ulteriori Informazioni

È possibile in ogni momento passare dalla gestione del database con il codice che utilizza il solo driver JDBC al codice che utilizza il framework Hibernate e viceversa.

Infatti, grazie all'utilizzo del costrutto `try-with-resources`, è possibile gestire le eccezioni e chiudere in modo automatico le connessioni, indipendentemente dalla loro implementazione.

I metodi che ricevono la connessione come parametro utilizzano se necessario il costrutto `try-with-resources`, senza utilizzo del `catch`, lasciando la gestione dell'eccezione al chiamante.

Molto importante è la gestione dell'atomicità delle operazioni ottenuta grazie ai `commit` in caso di successo e ai `rollback` in caso di fallimento.

Attenzione: si ricorda che in SQL gli indici partono da 1, mentre da Java da 0. Prestare attenzione nella gestione degli array (e delle rispettive tabelle SQL).

4 Risoluzione dei Problemi

4.1 Accesso all'Applicazione Non Disponibile

Consigliamo di consultare il sito <http://www.isitdownrightnow.com> in caso di problemi di accesso all'applicazione per verificare se il problema riguarda il vostro provider oppure il nostro *server_G*. In quest'ultimo caso vi preghiamo di riprovare l'accesso in un secondo momento.

4.2 Segnalazione di Bug

IronWorks potrebbe contenere qualche errore o *bug_G* inaspettato. È possibile segnalare qualsiasi malfunzionamento scrivendo direttamente alla nostra e-mail swearoncode@gmail.com.

Si prega di segnalare soltanto errori relativi all'applicazione IronWorks. Questo prodotto opera all'interno di un browser e, pertanto, non può essere imputato colpevole di malfunzionamenti non interni ad esso.

Si prega dunque di **non** segnalare errori relativi al proprio sistema, come ad esempio:

- Malfunzionamenti del sistema operativo;
- Difficoltà di avvio del browser o errori dello stesso.

Sono invece ben accette segnalazioni relative a:

- Impossibilità di collegarsi all'applicazione tramite la rete all'indirizzo:
<http://46.101.244.120> ;
- Errori relativi al file ".json" da caricare come progetto esistente;
- Errori di utilizzo dell'editor, come impossibilità di aggiungere nuovi elementi o modificarli;
- Errori relativi al download del file ".json" di progetto;
- Errori relativi al download del file ".zip" e dei files al suo interno, contenenti il codice generato;
- Malfunzionamenti di vario genere, relativi a comportamenti inattesi rispetto a quanto riportato in questo manuale.

A Glossario

A

Applicazione web

Indica genericamente un'applicazione distribuita, basata su tecnologie per il web e accessibile via web per mezzo di un network.

Attore

Elemento esterno al sistema che interagisce con quest'ultimo mediante determinate regole e limitato da specifici vincoli.

Architettura

Insieme di componenti, a loro volta suddivisi in unità e in *moduli_G*, che formano il sistema, definendone l'organizzazione e l'interazione tra questi.

B

Boundary

Chiamata anche interfaccia, rappresenta elementi software come schermate, report, pagine *HTML_G* o interfacce di sistema con cui gli *attori_G* interagiscono.

Bug

Identifica un errore di funzionamento di un sistema, di un programma, o in generale un errore nella scrittura del codice sorgente di un programma software.

C

Control

Chiamato anche "Controller", implementa la logica necessaria per gestire i vari elementi e le loro interazioni fungendo da collante tra oggetti *boundary_G* ed *entità_G*.

CRUD

Acronimo di "Create Read Update Delete", indica le quattro funzioni di base di un sistema informatico che associa *utente_G* e risorse.

D

Dao

Acronimo di Data Access Object, indica una classe che possiede i metodi per rappresentare un'entità in un database relazionale, creando un maggiore livello di astrazione ed una più facile manutenibilità.

Database

Insieme organizzato di dati, strutturati e collegati tra loro secondo un determinato modello logico. Possono adottare un modello relazionale ove i dati sono salvati su tabelle interconnesse tra loro, oppure un modello non relazionale che salva i dati su documenti strutturati (ad esempio su *file_G JSON_G*).

Diagramma di robustezza

Diagramma *UML_G* semplificato che utilizza dei simboli grafici per rappresentare 5 concetti principali: *Actor_G*, *Boundary_G*, *Control_G*, *Entity_G*, Linea di associazione.

E

Editor

Software che permette di utilizzare, verificare, riorganizzare, modificare e salvare informazioni testuali o grafiche.

Entità

Insieme di elementi che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse.

F

File

Viene utilizzato per riferirsi a un contenitore di informazioni/dati in formato digitale, tipicamente presenti su un supporto digitale di memorizzazione opportunamente formattato in un determinato *file*_G system.

Firefox Quantum

Browser web *open source*_G e multiplatforma, è mantenuto da Mozilla Foundation.

G

GitHub

Piattaforma di *hosting*_G web per lo sviluppo di progetti software basata sul *controllo versione*_G di *Git*_G.

Google Chrome

Browser web sviluppato da Google, oggi giorno ricopre una delle prime posizioni tra i browser web più utilizzati a livello mondiale.

H

Hibernate

Piattaforma middleware open source per lo sviluppo di applicazioni Java che fornisce un servizio di ORM, ovvero gestisce la persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su database relazionale di un sistema di oggetti Java.

I

IDE

Acronimo di "Integrated Development Environment", è un software che assiste il programmatore nello sviluppo del proprio software.

InnoDB

Motore per il salvataggio di dati per MySQL.

J

Java

Linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione.

JavaScript

Linguaggio orientato agli oggetti e agli eventi utilizzato nella programmazione Web.

JDBC

Acronimo di "Java DataBase Connectivity", è un driver per database che consente l'accesso e la gestione della persistenza dei dati da qualsiasi programma in linguaggio Java, indipendentemente dal tipo di database utilizzato.

M

Microsoft Edge

Browser web sviluppato da Microsoft e incluso in Windows 10, sostituendo Internet Explorer come browser predefinito di Windows.

MySQL

Sistema di gestione di database relazionale composto da un client a riga di comando e un server.

N

Node.js

Web server open source, fornisce un framework basato su JavaScript utilizzato per la gestione degli eventi.

npm

Acronimo di "Node Package Manager", viene utilizzato per la gestione dei pacchetti Node.js.

O

ORM

Acronimo di "Object-Relational Mapping", è un tecnica di programmazione che favorisce l'integrazione tra software che utilizzano linguaggi orientati ad oggetti e database relazionali, riducendo la complessità del codice.

P

POJO

Acronimo di "Plain Old Java Object", indica un oggetto semplice costruito in linguaggio Java, la cui classe non utilizza estensioni o implementazioni di interfacce o annotazioni particolari.

S

Safari

Browser web sviluppato dalla Apple Inc. per i sistemi operativi iOS e macOS.

Server

Insieme di componenti per l'elaborazione e la gestione del traffico di informazioni attraverso una rete di computer o un sistema informatico.

Singleton

Pattern creazionale che ha lo scopo di garantire che di una determinata classe venga creata una ed una sola istanza e di fornire un punto di accesso globale a tale istanza.

SQL

Acronimo di "Structured Query Language", linguaggio di programmazione per la gestione e l'amministrazione di *database*_G relazionali.

Standard

Documento approvato da un ente riconosciuto che fornisce le regole, le linee guida, le specifiche tecniche o le caratteristiche di alcune attività.

T

Try-with-resource

Costrutto particolare di tipo try che dichiara le risorse aperte e utilizzate e si assicura che esse vengano correttamente chiuse alla fine dell'esecuzione.

U

UML

Acronimo di "Unified Modeling Language", è un linguaggio di modellazione e specifica orientato agli oggetti basato su una notazione semi-grafica e semi-formale.

Utente

Fruitore finale di un prodotto software.