# Documentazione riepilogativa – Modulo Corsi / Esami / Studenti

## 1. Introduzione

Questo documento riassume la struttura del modulo applicativo che gestisce corsi, esami e studenti basato sul framework PSGLibrary / PSGExt. L'architettura segue un modello MVC proprietario composto da:
- Bean di dominio persistenti (Model)
- Controller W3 (Controller)
- JSP con tag PSG (View)
- File XML di mapping per la persistenza
- Classi di Query personalizzate

## 2. Classi di dominio (Model)

**Corso** – rappresenta un corso scolastico. Campi principali: code, description, teacher, startDate, endDate, remote. Gestisce inoltre le relazioni con gli esami (List<Esame>) e con i legami corso–studente (List<CorsoStudente>). Implementa Persistent, ValidableInterface, DynamicInfoInterface, InteractiveInterface e utilizza un JDBCMemento per la persistenza.

**Esame** – rappresenta un esame associato a un corso. Campi: id, corso (codice del corso), description, date, maxStudent. È un bean persistente PSGLibrary con validazione e metadata dinamici, mappato su SA_SARD_ESAME.

**Studente** – anagrafica studente. Campi tipici: id, name, surname, startDate, endDate, cittadinanza. Mappato sulla tabella SA_SARD_STUDENTE e usato come entità principale per la gestione delle iscrizioni e ricerche.

**CorsoStudente** – associazione tra Corso e Studente. Campi: id, corso (codice corso), studente (id studente), voto. Rappresenta la partecipazione di uno studente a un corso e l'eventuale voto. Mappato sulla tabella SA_SARD_CORSO_STUDENTE.

## 3. Controller W3 (Controller)

**WizardW3** – controller W3 che gestisce un wizard di creazione corso. Riceve i dati da wizard.jsp, verifica i permessi utente tramite ProfileManager, crea un Corso e una serie di Esame in base alla frequenza scelta (mensile, trimestrale, semestrale, annuale). Utilizza ConnectionManager, ServiceManager e SequenceManager per persistenza e generazione del codice corso.

**CorsoW3** – controller principale per la gestione dei corsi. Estende W3ControllerPersistent e implementa le operazioni CRUD su Corso, la navigazione tra liste e dettagli, l'uso di CorsoQuery per la ricerca e il caricamento di esami e studenti associati.

**EsameW3** – controller W3 dedicato alla gestione CRUD dell'entità Esame. Collega le JSP di dettaglio esame alle operazioni di Store/Retrieve/Remove e al bean Esame.

**StudenteW3** – controller W3 per la gestione dell'anagrafica studenti. In init() configura la query StudenteQuery, imposta le pagine di ricerca (studenteFind.jsp) e di editing (studenteEdit2.jsp), stabilisce lo stato iniziale a FIND e abilita l'uso dei servizi e del profiler per la classe Studente.

**CorsoStudenteW3** – controller W3 per la gestione delle associazioni corso–studente (CorsoStudente). Permette di creare, modificare e cancellare le iscrizioni degli studenti ai corsi e gestisce eventuali voti.

## 4. Classi di Query personalizzate

**CorsoQuery** – classe di servizio che costruisce SQL dinamico per la ricerca dei corsi. Utilizza un oggetto filtro Corso e JDBCDataMapper per tradurre i campi in clausole WHERE, con supporto per ordinamenti e filtri multipli.

**StudenteQuery** – implementa una CustomQueryProfile per la ricerca degli studenti. Definisce i filtri formali (idFilter, cognomeFilter, nomeFilter, soloIscritti), le colonne restituite (matricola, nome completo) e costruisce la SQL string applicando condizioni sul cognome, nome e sul periodo di iscrizione tramite ST_START_DATE e ST_END_DATE.

## 5. File XML di mapping (Persistenza)

**Corso.xml** – mappa il bean Corso sulla tabella SAD_SARD_CORSO. Definisce code come chiave primaria e mappa description, teacher, startDate, endDate, remote con relativi tipi e lunghezze.

**Esame.xml** – mappa il bean Esame sulla tabella SA_SARD_ESAME. Definisce id come chiave primaria e le proprietà corso, description, date, maxStudent.

**CorsoStudente.xml** – mappa il bean CorsoStudente sulla tabella SA_SARD_CORSO_STUDENTE. Proprietà: id, corso, studente, voto.

**Studente.xml** – mappa il bean Studente sulla tabella SA_SARD_STUDENTE con proprietà id, name, surname, startDate, endDate, cittadinanza.

## 6. JSP principali (View)

Di seguito un riepilogo delle pagine JSP collegate ai controller W3:

**wizard.jsp** – vista del wizard per la creazione di un corso. Collega le proprietà di WizardW3 (description, teacher, frequenza, dataEsameAnnuale) e invia l'azione CREA_CORSO.

**corsoFind.jsp** – pagina di ricerca corsi, utilizzata da CorsoW3 assieme a CorsoQuery.

**corsoEdit.jsp** – pagina di editing/dettaglio corso, legata a CorsoW3.

**corsoStudenteDetail.jsp** – pagina di dettaglio per la relazione corso–studente (gestione iscrizioni / voti).

**esameDetail.jsp** – dettaglio di un esame, usata da EsameW3.

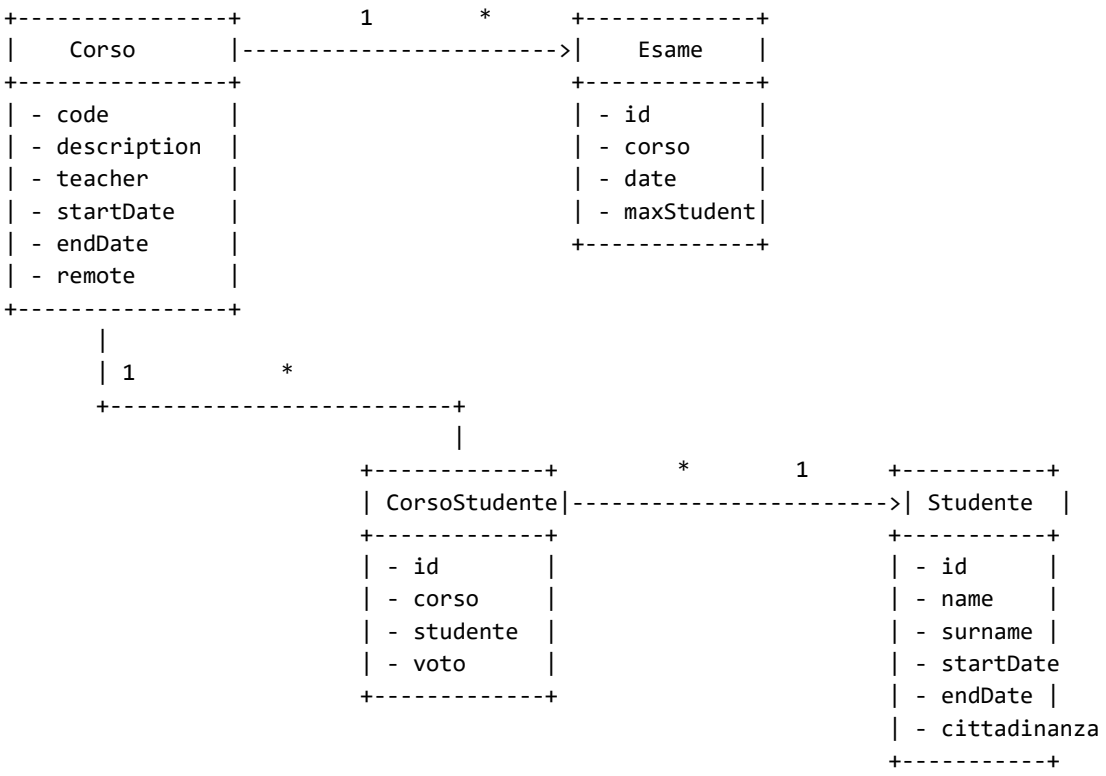
**esameDetailGrid.jsp** – griglia di visualizzazione esami associati a un corso.

**studenteFind.jsp** – pagina di ricerca studenti, collegata a StudenteW3 e StudenteQuery.

**studenteEdit2.jsp** – pagina di editing/dettaglio dell'anagrafica studente.

## 7. Diagramma UML (vista logica)

Di seguito una rappresentazione UML testuale delle principali classi di dominio e delle relazioni tra loro.




I controller W3 (CorsoW3, EsameW3, StudenteW3, CorsoStudenteW3, WizardW3) operano come layer di presentazione e non sono rappresentati qui nel dettaglio UML di dominio, ma si appoggiano alle classi sopra rappresentate per le operazioni di business.

## 8. Schema riepilogativo delle classi

| Nome | Tipo | Ruolo principale | Tabella / Vista | Relazioni chiave |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Corso | Bean di dominio | Gestione corsi scolastici | SAD_SARD_CORSO | 1-* con Esame, 1-* con CorsoStudente |
| Esame | Bean di dominio | Esami di un corso | SA_SARD_ESAME | Molti esami per un Corso |
| Studente | Bean di dominio | Anagrafica studenti | SA_SARD_STUDENTE | Molti CorsoStudente per uno Studente |
| CorsoStudente | Bean di dominio | Relazione corso–studente + voto | SA_SARD_CORSO_STUDENTE | Molti per uno Studente e uno Corso |
| WizardW3 | Controller W3 | Wizard creazione corso + esami | wizard.jsp | Usa Corso, Esame |
| CorsoW3 | Controller W3 | CRUD corso + ricerche | corsoFind.jsp / corsoEdit.jsp | Usa Corso, CorsoQuery, Esame, CorsoStudente |
| EsameW3 | Controller W3 | CRUD esame | esameDetail.jsp / esameDetailGrid.jsp | Usa Esame |
| StudenteW3 | Controller W3 | CRUD studente + ricerche | studenteFind.jsp / studenteEdit2.jsp | Usa Studente, StudenteQuery |
| CorsoStudenteW3 | Controller W3 | Gestione iscrizioni e voti | corsoStudenteDetail.jsp | Usa CorsoStudente, Corso, Studente |
| CorsoQuery | Query | Ricerca corsi | - | Usata da CorsoW3 |
| StudenteQuery | Query | Ricerca studenti | - | Usata da StudenteW3 |

| | | | | |
|---|---|---|---|---|
| Corso.xml | XML mapping | Mappatura bean Corso | SAD_SARD_CORSO | Definisce campi e chiave |
| Esame.xml | XML mapping | Mappatura bean Esame | SA_SARD_ESAME | Definisce campi e chiave |
| CorsoStudente.xml | XML mapping | Mappatura CorsoStudente | SA_SARD_CORSO_STUDENTE | Definisce campi e chiave |
| Studente.xml | XML mapping | Mappatura Studente | SA_SARD_STUDENTE | Definisce campi e chiave |

**1. Il Modello (Model)**

Il Modello è composto da due elementi principali: il **Bin** (POJO) e la **Classe Query**.

**A. Bin (Plain Old Java Object)**

Il Bin rappresenta un singolo record della tabella (per le pagine di Edit/Insert).

| **Elemento** | **Dettaglio** | **Fonte** |
|---|---|---|
| **Classe Base** | Una semplice classe Java (POJO) con proprietà, getter e setter. | |
| **Interfacce Obbligatorie** | Deve implementare Persistent, ValidableInterface, e DynamicInfoInterface. | |
| **Metodi Core** | I metodi per la persistenza (retrieve, store, remove) sono implementati dall'interfaccia Persistent. | |
| **XML di Mapping** | Richiede un file XML gemello (es. Corso.xml) per mappare le proprietà del Bin (<property name="..."\>) alle colonne del DB (field="..."). | |

**B. Classe Query**

La Classe Query è usata per costruire lo statement SQL (per le pagine di Find/Results).

| **Elemento** | **Dettaglio** | **Fonte** |
|---|---|---|

| Elemento | Dettaglio | Fonte |
|---|---|---|
| **Estende** | Deve estendere la classe astratta CustomQueryProfile. | |
| **Logica** | Contiene il metodo getCustomSQLString che ritorna la SELECT. | |
| **Filtri/Colonne** | Nel costruttore, si dichiarano i filtri attesi (formFilter) e le colonne di risultato (fieldName, fieldCaption). | |

---

**2. La Vista (View - JSP)**

Le View sono realizzate usando semplici pagine JSP. Di default, ne servono due:

| Pagina | Scopo | Note di Implementazione | Fonte |
|---|---|---|---|
| **FIND.jsp** | Mostra i filtri di ricerca. | La pagina dei risultati (Results Page) è generata automaticamente dal framework in base a quanto definito nella Classe Query. | |
| **EDIT.jsp** | Usata per le operazioni di modifica (EDIT) e inserimento (INSERT). | Si usa lo stesso file JSP per entrambe le modalità. | |

Il collegamento tra la View e il Modello (il **Binding**) avviene tramite custom Tag Libraries, in particolare <psg:linkProperty>.

- Su FIND.jsp, usi <psg:linkProperty propertyName="codeFilter" isFilter="Y"> per legare un campo input a un filtro della Query Class.
- Su EDIT.jsp, usi <psg:linkProperty propertyName="code" useBeanInfo="Y"> per legare un campo input a una proprietà del Bin.

---

**3. Il Controller**

Il Controller gestisce le azioni dell'utente e funge da collante tra Model e View.

| Elemento | Dettaglio | Fonte |
|---|---|---|
| **Estende** | W3ControllerPersistent (per la testata) o W3ControllerPersistentDetail (per il dettaglio). | |
| **Metodo Base** | Deve implementare il metodo init(HttpServletRequest req). | |

| Elemento | Dettaglio | Fonte |
|---|---|---|
| **Inizializzazione** | In init, deve: | |
| Query | Inizializzare la Classe Query tramite ServiceResultBuilder, specificando la PK (code per l'esempio Corso). | |
| View Paths | Dichiarare le JSP di Find e Edit tramite setReturnPage. | |
| Stato Iniziale | Impostare lo stato iniziale (di solito STATUS_FIND). | |
| Bin | Abilitare il Bin associato tramite enableProfiler(Corso.class.getClass()). | |

**Esempio di Paths nel Controller:**

```
setReturnPage(STATUS_FIND, "/sianc/sisar/corso/corsoFind.jsp");
setReturnPage(STATUS_EDIT, "/sianc/sisar/corso/corsoEdit.jsp");
```

---

**4. Collegamento (Mapping)**

Per rendere il Controller accessibile, è necessario definire un **Mapping** nel file di Properties del progetto (es. controller.properties in GSP/WebInf/controller/).

Il framework (tramite l'unica servlet dichiarata, il Controller Dispatcher) intercetta le URL che terminano in .do e utilizza il Mapping per istanziare la classe Controller corretta.

Il Mapping tipico è:

```
mapping.sisar.corso.do=sianc.sisar.corso.CorsoW3
bin.sianc.sisar.corso.CorsoW3=sianc.sisar.corso.Corso
```

- La prima riga associa l'URL (sisar.corso.do) alla classe Controller (CorsoW3).
- La seconda riga associa la classe Controller al Bin corrispondente (Corso) per fini interni al framework.