Mitro

μητρώο

RAPPORTO SULLA FASE DI REALIZZAZIONE

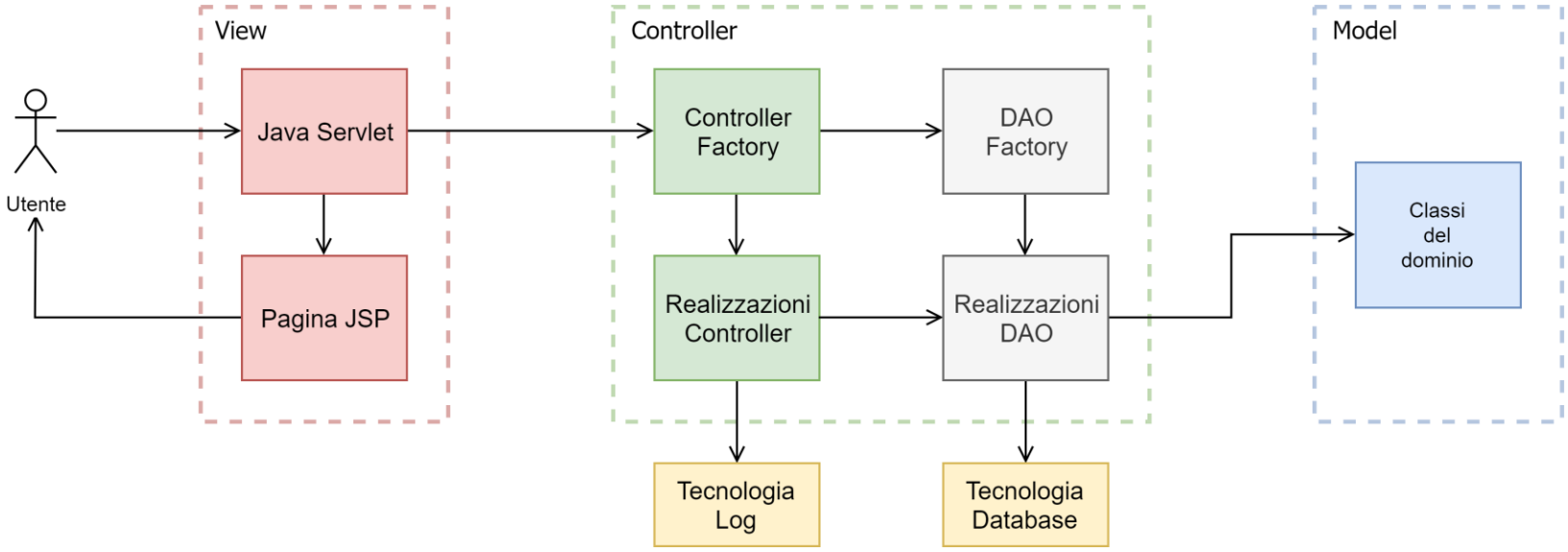
Amir Al Sadi 0000792816
Jason Dellaluce 0000789710
Federico Baldini 0000789767

Che cos'è Mitro

- Registro elettronico destinato ad istituti scolastici italiani di livello secondario
- Servizio installabile e mantenibile dagli istituti scolastici stessi, per abbattere problemi di costi e disponibilità degli attuali sistemi centralizzati
- Profili personalizzati per studenti, professori, enti amministrativi
- Gestione di attività scolastiche, voti, appelli, comunicazioni interne
- Tutela della privacy e sistemi di separazione dei privilegi

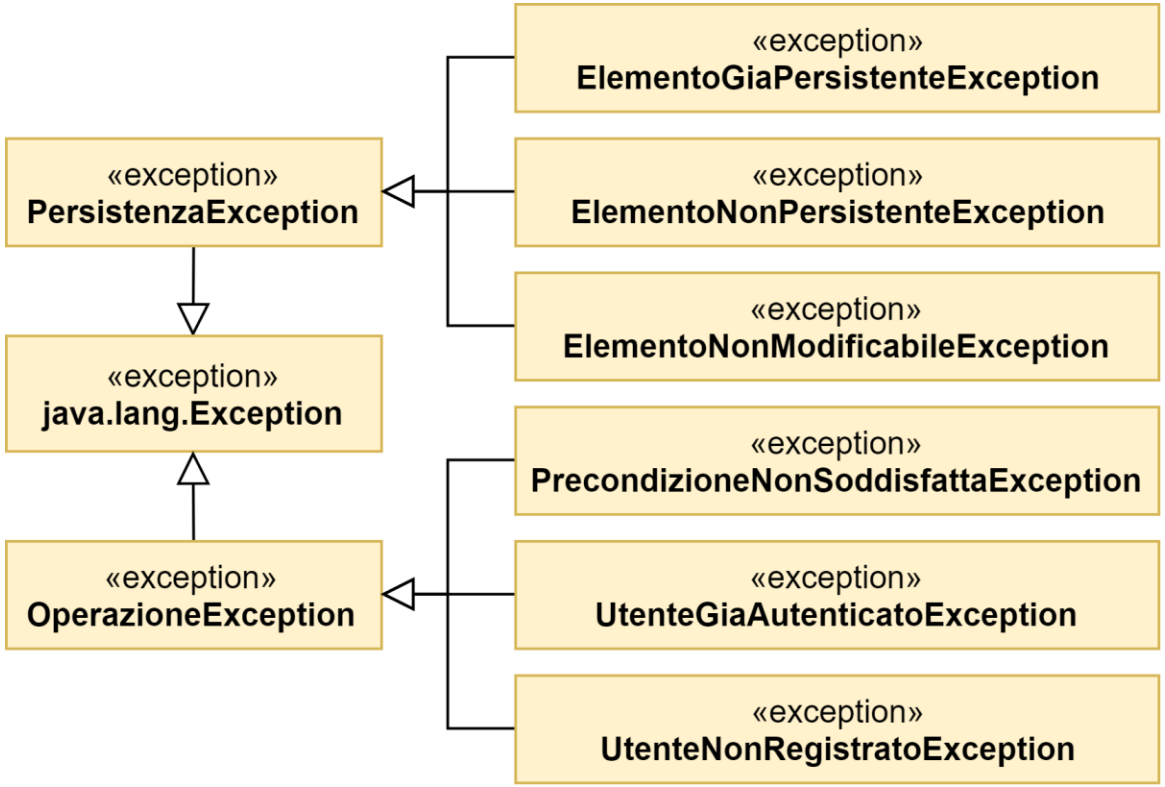
Architettura dell'applicativo

- Utilizzo del pattern architetturale Model View Controller
- Forte disaccoppiamento tra tecnologia di presentazione e logica di business



Gestione degli errori specifici

- Gerarchia interna di eccezioni
- Separazioni di errori del livello di persistenza da quelli della logica di business
- Permettono un facile riconoscimento di problemi specifici e ricorrenti
- Facile presentabilità degli errori agli utenti nelle View

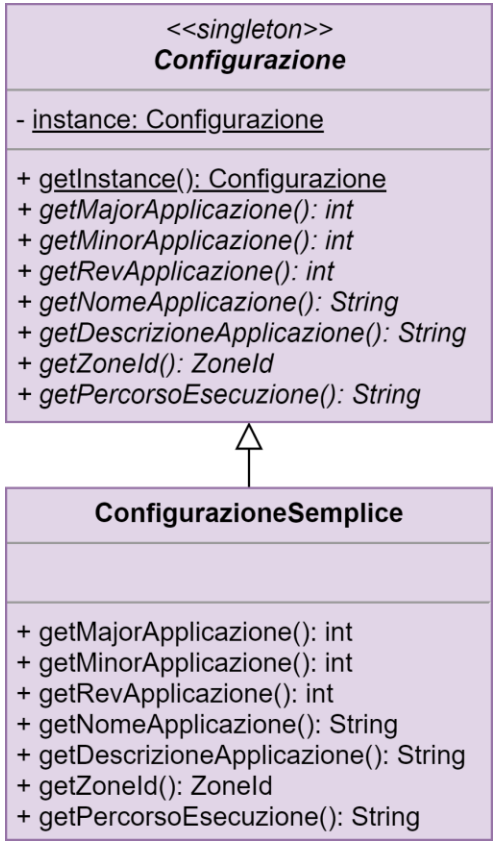


Realizzazione interfacce grafiche

- Progetto strutturato come Web Application con Web Container Apache Tomcat
- View implementate con tecnologie Java Servlet e Java Server Pages
- Utilizzo di un template grafico gratuito per il prototipo
- Dati utente ed informazioni sull'autenticazione mantenuti nello stato di sessione delle Servlet

Realizzazione configurazione

- Nuova classe Singleton **mitro.deployment.Configurazione**
- Informazioni comuni a tutte le componenti dell'applicativo circa dettagli del deployment e configurazione
- Utile per tenere traccia delle versioni del software e sincronizzare tutte le istanze replicate del servizio sullo stesso fuso orario
- Nel prototipo implementata con informazioni hard-coded
- Realizzazioni future: lettura da file o connessione remota



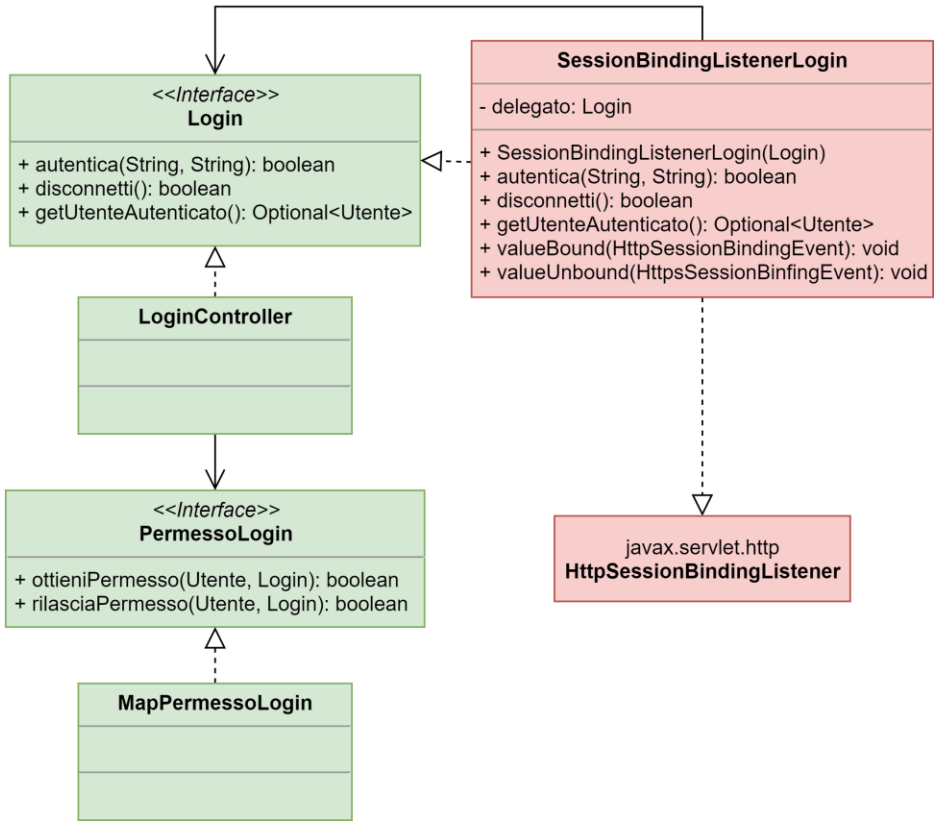
Utilizzo del pattern Factory

- **mitro.controller.ControllerFactory**
  - Implementazioni dei Controller seguono il pattern Dependency Injection nei costruttori, perfavorire configurabilità e collaudo
  - Classe Factory realizzata come Singleton astratto, permettendo realizzazioni multiple

- **mitro.persistenza.DAOFactory**
  - Classe Factory realizzata come Singleton astratto, permettendo realizzazioni multiple
  - Creazione e configurazione di tutte le realizzazioni delle interfacce DAO

Realizzazione gestione delle autenticazioni

- Autenticazione effettuata interrogando la persistenza con le credenziali
- Mantenimento del **Login** in stato di sessione
- Interfaccia **PermessoLogin** come Monitor per coordinamento centralizzato degli accessi
  - Realizzato con una collezione sincronizzata locale
  - Realizzabile con connessioni remote in caso di server replicati
- **SessionBindingListenerLogin** con pattern Decorator e Observer
  - Ascolto di eventi sulla sessione Servlet, per disconnessione automatica in caso di invalidazione



Realizzazione gestione dei Log

- Classi astratte **mitro.controller.ControllerAstratto** e **mitro.view.ViewAstratta** si occupano della corretta formattazione dei Log
- Interfacce **LoggerOperazioni** e **LoggerMessaggi** implementate utilizzando classi Writer di Java, per facile collaudo e cambiamento (Open/Close Principle)
- Scrittura delle voci su file locale
- Utilizzo del pattern Strategy per gli algoritmi di analisi delle anomalie

Realizzazione persistenza

- Utilizzo del pattern DAO per nascondere la tecnologia scelta e favorire sostituibilità e mantenimento
- Realizzazioni basate su tecnologie DBMS e framework JDBC
  - Realizzazione modello Entità-Relazione in tabelle SQL
  - Dipendenza dall'interfaccia **DataSource** per sostituibilità del DBMS
  - Utilizzo della libreria SQLite nel prototipo, per salvataggio semplificato su file
- Implementazioni di classi DAO Mock
  - Mantengono internamente una collezione di dati
  - Utili per collaudo dell'applicazione e test unitari

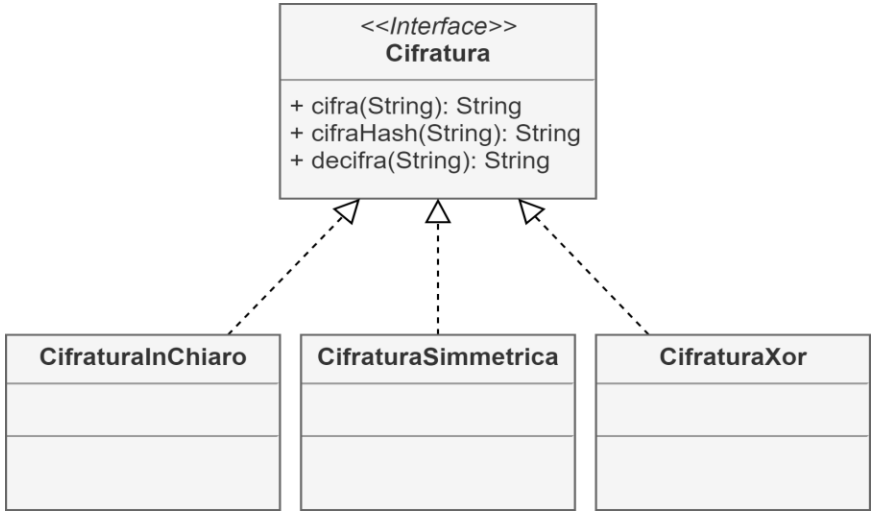
Realizzazione cifratura dei dati

- Uso del pattern Strategy per l'algoritmo di cifratura
- Interfaccia **Cifratura** utilizzata nelle classi DAO per proteggere dati sensibili
- Implementazioni differenziate da usare in base ai requisiti di performance e sicurezza

Struttura database | Naviga nei dati | Modifica Pragmas | Esegui SQL

Tabella: UTENTI

	Id	Username	Password	Ruolo	Nome	Cognome
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
5	5	#$%4}e	1##'?"4◀◀◀◀...	ST	↔1"9?	●%##?
6	6	#$%4}f	1##'?"4◀◀◀◀...	ST	‼891"1	▂5""1"9
7	7	#$%4}g	1##'?"4◀◀◀◀...	ST	▲93?<¢	●9339
8	8	#$%4}h	1##'?"4◀◀◀◀...	ST	▲93?<¢	↔1"9>?
9	9	#$%4}i	1##'?"4◀◀◀◀...	ST	∟1%"1	●%##?



Collaudo delle componenti

- Oltre 30 classi di test, per oltre 50 test unitari
- Utilizzo di Dependency Injection e classi Mock per collaudare le classi Controller
- Utilizzo di file temporanei SQLite per collaudare le realizzazioni DAO SQL.
- Utilizzo di Reflection per il collaudo delle classi del dominio, progettate come Java Bean