

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

CORSO DI SYSTEM ARCHITECTURE DESIGN PROF.SSA FASOLINO ANNA RITA

A.A. 2022-23

**PROGETTO ENACTEST RELAZIONE GRUPPO #BOH TASK DI RIFERIMENTO: T1-T5**

|  |  |  |
| --- | --- | --- |
| **Studenti**: |  |  |
| Argentino Simone | M63001555 | [si.argentino@stu](mailto:si.argentino@stu)denti.unina.it |
| Borrelli Francesco | M63001559 | [francesco.borrelli6@studenti.unina.it](mailto:francesco.borrelli6@studenti.unina.it) |
| Mauriello Augusto | M63001545 | Au.mauriello@studenti.unina.it |
| Romano Vincenzo | M63001557 | vincenzo.romano31@studenti.unina.it |

Sommario

1. [INTRODUZIONE 4](#_bookmark0)
   1. [Scenario 4](#_bookmark1)
   2. [Requisito specifico assegnato 4](#_bookmark2)
   3. [Metodologia di lavoro 4](#_bookmark3)
2. [SPECIFICA DEI REQUISITI 7](#_bookmark4)
   1. [Attori, funzionalità e attributi 7](#_bookmark5)
   2. [Glossario dei termini 7](#_bookmark6)
3. [USER STORIES 9](#_bookmark7)
   1. [Individuazione delle Storie Utente 9](#_bookmark8)
   2. [Criteri di Accettazione 13](#_bookmark9)
4. [ANALISI DEI REQUISITI 15](#_bookmark10)
   1. [Descrizione informale dei requisiti. 15](#_bookmark11)
   2. [Requisiti funzionali 15](#_bookmark12)
   3. [Requisiti sui dati 16](#_bookmark13)
   4. [Requisiti non funzionali 16](#_bookmark14)
5. [SCELTEDIPROGETTO 17](#_bookmark15)
   1. [Pattern architetturali utilizzati 17](#_bookmark16)
   2. [Stili architetturali utilizzati 17](#_bookmark17)
6. [TECNICHE DI SVILUPPO 19](#_bookmark18)
7. [MODELLO DEI CASI D’USO 24](#_bookmark19)
   1. [Componenti principali 24](#_bookmark20)
   2. [Diagramma dei Casi d’Uso 24](#_bookmark21)
   3. [Scenari Casi d’Uso 25](#_bookmark22)
8. [SYSTEM DOMAIN MODEL 29](#_bookmark23)
9. [MODELLAZIONE DEI DATI 30](#_bookmark24)
10. [CONTEXT DIAGRAM 31](#_bookmark25)
11. [COMPONENT DIAGRAM 32](#_bookmark26)
12. [PACKAGE DIAGRAM 33](#_bookmark27)
13. [API](#_bookmark28)
    1. [Introduzione eproprietà 35](#_bookmark29)
    2. [Descrizione formale delle REST API 36](#_bookmark30)
    3. [REST API in AdminClasseController 36](#_bookmark31)
    4. [REST API in UserClasseController 38](#_bookmark32)
14. [ACTIVITY DIAGRAM 40](#_bookmark33)
    1. [EliminaClasse 40](#_bookmark34)
    2. [DownloadCodiceClasse 41](#_bookmark35)
    3. [AggiungiClasse 41](#_bookmark36)
    4. [RicercaAvanzata 42](#_bookmark37)
    5. [ModificaClasse 42](#_bookmark38)
    6. [VisualizzaElenco 43](#_bookmark39)
15. [SEQUENCE DIAGRAMS 44](#_bookmark40)
    1. [VisualizzaClassi e DownloadCodiceClasse 44](#_bookmark41)
    2. [RicercaAvanzata 45](#_bookmark42)
    3. [GestioneClasse 47](#_bookmark43)
    4. [AggiuntaClasse 48](#_bookmark44)
16. [COMMUNICATION DIAGRAMS 49](#_bookmark45)
    1. [Gestione Classi Communication Diagram 50](#_bookmark46)
    2. [Ricerca Avanzata Communication Diagram 51](#_bookmark47)
17. [STATE DIAGRAM 53](#_bookmark48)
18. [DEPLOYMENT 55](#_bookmark49)
19. [TESTING 58](#_bookmark50)

# INTRODUZIONE

## Requisito specifico assegnato

////////////////////////////////////////////////////////////////////////////////////////

Il requisito specifico assegnato al gruppo, composto dagli studenti Argentino Simone, Borrelli Francesco, Mauriello Augusto e Romano Vincenzo, frequentanti il primo anno magistrale di Ingegneria Informatica presso l'Università di Napoli Federico II, è relativo all’unione di due diversi task:

*1)l’applicazione deve mantenere un insieme di Classi Java da testare e deve offrire la possibilità ai giocatori di consultare l’elenco delle classi disponibili e di fare il download del codice di una di esse. L’applicazione deve permettere ad un amministratore anche di aggiornare l’insieme di classi disponibili mediante aggiunta di classi e relativo salvataggio del file di codice. Sarebbe auspicabile anche prevedere funzioni per la ricerca di classi in base a specifici requisiti, come ad esempio la complessità della classe, o altri attributi.*

*2)* *Il giocatore (dopo essersi autenticato) avvia una nuova partita del Primo Scenario, l’applicazione gli mostra un elenco di classi da testare ed un elenco di Robot disponibili, il giocatore sceglie la classe ed il Robot contro cui confrontarsi. A questo punto il sistema crea la partita con tutte le scelte fatte, le associa un IdPartita, e la salva. Successivamente l’applicazione avvia l’ambiente di editing in cui visualizza la classe da testare e gli offre una finestra in cui può scrivere la classe di test.*

# SPECIFICA DEI REQUISITI

## Attori, funzionalità e attributi

Per prima cosa, risulta fondamentale elencare gli attori, le funzionalità e gli attributi:

-Attori:

-Giocatore;

-Amministratore.

-Funzionalità:

-Mantenere un insieme di classi da testare;

-Consultare un elenco di classi disponibili;

-Download delle classi disponibili;

-Aggiunta di classi;

-Salvataggio del file del codice delle classi;

-Ricerca di classi;

-Avviare una partita;

-Scelta della classe da testare e robot avversario;

-Finestra di editing per testare la classe.

-Attributi:

-Complessità della classe

-Altri attributi della classe da testare( Vedi T1-G3).

## Glossario dei termini

Segue il glossario dei termini, definito al fine di offrire definizioni e spiegazioni chiare dei termini comuni utilizzati in questa relazione.

|  |  |  |
| --- | --- | --- |
| Termine | Descrizione | Sinonimi |
| **Classe Java** | Descrizione degli attributi di un file .java che si vuole caricare nel database, contente la classe da testare | Specifiche file |
| **File Java** | File con estensione .java contenente il codice Java di una classe da testare | Codice della classe |
| **Last Update** | Data dell’ultima modifica della classe Java | Data ultimo Aggiornamento |
| **LOC** | Numero di linee di codice del file Java | Line di Codice |

|  |  |  |
| --- | --- | --- |
| **Complexity** | Complessità del file Java. Le possibilità previste sono:  -Simple  -Moderate  -Complex  -Insane | Complessità della classe |
| **Recommended Opponent** | Avversario consigliato. Le possibilità previste sono:   * RANDOOP * EVOSUITE | avversario consigliato |
| **Download** | Trasferimento di un file in locale a partire dalla rete | Scaricamento |
| **Test** | Parte del ciclo di vita di un software che mira ad individuare la correttezza, la completezza e l’affidabilità dello stesso | Software Testing,  collaudo del software |
| **Amministratore** | Utente che può aggiornare l’insieme delle classi disponibili mediante l’aggiunta di classi e relativo salvataggio del file di codice | Gestore |
| **Giocatore** | Utente che può visualizzare l’elenco delle classi e farne il relativo download | Competitore, Utente registrato |
| **Repository** | Archivio digitale di informazioni centralizzato | Archivio, database |
| **Partita** | Sfida avviata dall’utente, che può portare a vittoria, pareggio o sconfitta | Sfida, Competizione |

# USER STORIES

## Individuazione delle Storie Utente

La storia utente è un concetto chiave nell'Ingegneria del Software, in particolare nel contesto dello Sviluppo Agile.

Una **storia utente** rappresenta un requisito funzionale o un'unità di lavoro che descrive una determinata funzionalità o caratteristica dal punto di vista dell'utente finale del software. In altre parole, una storia utente definisce cosa deve essere fatto dal sistema per soddisfare le esigenze degli utenti.

Solitamente, una storia utente è composta da una breve descrizione del **bisogno dell'utente**, i **criteri di accettazione** per valutare la sua realizzazione e un ordine di **priorità**.

Le storie utente sono spesso organizzate in un elenco prioritario (Product Backlog) e vengono selezionate per l'implementazione durante le iterazioni o gli Sprint di sviluppo.

Questo approccio permette di focalizzarsi sugli obiettivi più rilevanti e di consegnare valore in modo incrementale agli utenti durante tutto il processo di sviluppo del software.

In [Figura 3.1] vengono mostrate le 8 storie utente alla base del task:

1. In qualità di giocatore, voglio accedere al sistema affinché possa creare una partita;
2. In qualità di giocatore, voglio visualizzare l’elenco di classi disponibili
3. In qualità di giocatore, voglio selezionare un robot ed una classe affinché possa essere avviata una partita;
4. In qualità di giocatore, dopo aver avviato una partita, voglio poter visualizzare l’editor per competere contro il robot selezionato;
5. In qualità di amministratore, voglio aggiornare l’insieme di classi disponibili mediante aggiunta di classi e relativo salvataggio del file di codice;
6. L’applicazione deve mantenere un insieme di classi java da testare in una repository;
7. In qualità di utente(amministratore o giocatore), voglio ricercare delle classi in base a specifici requisiti.
8. In qualità di utente(amministratore o giocatore), voglio fare il download di una delle classi disponibili;

Storie utente da fare con schema simpatico.

# ANALISI DEI REQUISITI

In tale capitolo vengono anzitutto descritti i requisiti al fine di classificarli in funzionali e non, nonché in requisiti sui dati.

## Descrizione informale dei requisiti.

Si vuole realizzare un'applicazione ove sia possibile effettuare le seguenti operazioni:

* mantenere un insieme di classi facilmente consultabili e scaricabili in locale dagli utenti. Tali file hanno alcuni parametri di fondamentale importanza:
  + **ID**: utile per la persistenza sulla base dati, individua univocamente un file;
  + **path**: percorso adoperato per raggiungere un determinato file;

## Requisiti funzionali

Si elencano in seguito i requisiti funzionali.

* **RF-1**. Il sistema deve consentire il mantenimento di tutte le classi di testing caricate

dall’amministratore.

* **RF-2**. Il sistema deve offrire all’amministratore la possibilità di effettuare l’upload di un file .java

che contiene la classe da testare.

* **RF-3.** Il sistema deve offrire all’amministratore la possibilità di specificare le caratteristiche di ciascuna classe di testing caricata.
* **RF-4.** Il sistema deve offrire all’amministratore la possibilità di modificare gli attributi della classe di testing caricata.
* **RF-5.** Il sistema deve consentire all’amministratore di eliminare una classe precedentemente caricata, ed il relativo file .java.
* **RF-6.** Il sistema deve consentire all’utente (giocatore e amministratore) la possibilità di visualizzare tutte la classi di testing collezionate.
* **RF-7.** Il sistema deve consentire all’utente (giocatore e amministratore) di effettuare il download di un file precedentemente caricato.
* **RF-8** Il sistema deve consentire all’utente (giocatore e amministratore) di ricercare un sottoinsieme di classi in base a degli specifici filtri.
* **RF-9** Il sistema deve consentire la registrazione a qualsiasi utente voglia registrarsi.
* **RF-10** Il sistema deve permettere ad un utente registrato di poter fare il login.
* **RF-11** il sistema deve permettere ad un giocatore di poter avviare una partita.
* **RF-12** il sistema deve permettere al giocatore di poter scegliere la classe da testare.
* **RF-13** il sistema deve permettere al giocatore di poter scegliere il robot con il quale competere.
* **RF-14** Il sistema deve permettere al giocatore di poter salvare la partita, la quale verrà identificata mediante l’ausilio di un Id.
* **RF-15** Il sistema deve permettere al giocatore di poter avviare l’editor una volta avviata la partita.
* **RF-16** Il sistema deve permettere ad un utente registrato di poter recuperare le credenziali se si è dimenticato la password.

## Requisiti sui dati

Si elencano in seguito i requisiti sui dati.

* **RD-1**. Ad ogni file .java devono essere associate le seguenti informazioni: nome del file, contenuto e tipo di file
* **RD-2**. Per ogni classe di testing si deve specificare: complessità, avversario consigliato, data

dell’ultima modifica degli attributi e numero di linee di codice.

* **RD-3.** Ad ogni classe di testing deve essere associato il corrispondente file .java.

## Requisiti non funzionali

Si elencano in seguito i requisiti non funzionali.

* **RNF-1**: *Evolvibilità*

-Essendo il software implementato parte di un progetto di dimensioni più ampie, deve essere pensato per poter essere esteso, integrando ulteriori componenti.

* **RNF-2**: *Usabilità*

-Il sistema deve risultare intuitivo e semplice da usare, comprensibile sia al giocatore che

all’amministratore.

-Le operazioni consentite agli attori con cui interagisce il sistema devono essere immediate, per cui è opportuno che sia implementata un’interfaccia utente chiara e coerente.

* **RNF-3**: *Installabilità*

-L’applicazione deve risultare semplice da installare.

-Il processo di installazione deve essere opportunamente guidato dalla documentazione

* **RNF-4**: *Testabilità*

-Il testing dell’applicazione deve essere semplice, in modo da favorire il lavoro di altri task che sono intenzionati ad integrare il presente sistema dopo averlo testato.

* **RNF-5**: *Integrità*

-Il sistema deve garantire l’integrità dei dati contenuti nel database.

-Deve verificare l’integrità degli input prima di effettuare inserimento nel database.

-Il sistema deve garantire un buon grado di sicurezza nella gestione delle credenziali dell’utente.

-Il sistema deve garantire la consistenza dei dati e che gli identificativi siano univoci.

* **RNF-6**: *Modificabilità*

-Il sistema deve garantire la possibilità di modificare i suoi componenti e metodi facilmente.

-L’architettura software deve essere modulare al fine di semplificare un’operazione di modifica e manutenzione.

* **RNF-7**: *Comprensibilità*

-Il codice sorgente deve essere ben strutturato e ben organizzato.

-Il codice sorgente deve inoltre essere descritto da una accurata documentazione.

# SCELTE DI PROGETTO

## Pattern architetturali utilizzati

* **Model-View-Controller**.

Il pattern MVC è una popolare architettura per la progettazione di applicazioni web. Definisce tre componenti principali:

* + *Model*: Definisce i metodi per accedere e modificare i dati; rappresenta lo stato dell’applicazione. Il model può avere la responsabilità di notificare ai componenti view gli aggiornamenti dati da parte del controller, in modo che si aggiornino continuamente.
  + *View*: Delega al controller i processi conseguenti l’input dell’utente. Ha la responsabilità della presentazione dei dati, in modo che i dati siano sempre aggiornati.
  + *Controller*: Realizza l’associazione tra model e view. Incapsula la logica della applicazione, seleziona le schermate della view da utilizzare.

Immagine che contiene testo, diagramma, cerchio, schermata

Descrizione generata automaticamente

## Stili architetturali utilizzati

* **Stile Object Oriented**: i componenti sono oggetti, con dati e operazioni annesse; i connettori sono dati scambiati tramite messaggi ed invocazione di metodi. Ogni oggetto è responsabile dell’integrità della loro rappresentazione interna e questa è nascosta ad altri oggetti. L’adozione di tale stile ci ha consentito di avere interfacce ben definite e di utilizzare principi di progettazione come l’**information hiding** e l’**incapsulazione** che aumentano la **modificabilità**.
* **Stile Client-Server**: l'architettura client-server è un particolare tipo di stile architetturale a due livelli che suddivide un'applicazione in due componenti principali: il client e il server.

Il client implementa la presentation logic, mentre il server gestisce la logica di business, l'elaborazione delle richieste e l'accesso ai dati.

Nel contesto dell'architettura client-server, il pacchetto View rappresenta il lato del client dell'applicazione, che si occupa dell'interfaccia utente e della generazione delle pagine HTML da visualizzare. Il client fa richieste al server per ottenere i dati o eseguire operazioni.

I pacchetti Controller*,* Service *e* Repository rappresentano il lato del server dell'applicazione. Il pacchetto Controller riceve le richieste del client e coordina l'esecuzione delle operazioni appropriate. Il pacchetto Service contiene la logica di business dell'applicazione, mentre il pacchetto Repository gestisce l'accesso ai dati nel database.

In questa architettura, il client interagisce con il server tramite richieste API REST. Il server elabora le richieste, recupera i dati necessari, esegue le operazioni richieste e restituisce le risposte al client.

Alcuni vantaggi che questa scelta architetturale ha fornito sono i seguenti:

* + - *Separazione delle responsabilità*: L’architettura client-server permette di separare chiaramente le responsabilità tra i due componenti. Il client si occupa dell'interazione con l'utente, della presentazione dei dati e dell'interfaccia utente, mentre il server gestisce la logica di business, l'elaborazione delle richieste e l'accesso ai dati.
    - *Riutilizzo del codice***:** La separazione tra client e server consente il riutilizzo del codice. La logica di business e le operazioni complesse possono essere implementate nel server e utilizzate da diversi client. Questo riduce la duplicazione del codice e favorisce la manutenibilità dell'applicazione
    - *Manutenibilità e evolvibilità***:** L’architettura client-server semplifica la manutenibilità dell'applicazione. I cambiamenti nella logica di business possono essere effettuati nel server senza dover apportare modifiche significative al client. Inoltre, è possibile aggiornare o sostituire il server senza influire sull'interfaccia utente o sui client esistenti.
* **Stile REpresentational State Transfer (REST)**: le API REST (Representational State Transfer) sono uno stile architetturale per progettare e sviluppare servizi web. L'approccio REST si basa su un insieme di principi che promuovono l'interazione tra client e server attraverso richieste e risposte HTTP, utilizzando concetti come risorse, URL (Uniform Resource Locator) e metodi HTTP (GET, POST, PUT, DELETE).

Nel sistema realizzato le API REST sono utilizzate per accedere ai metodi del package View, e ai metodi del Controller. Quindi è lo strumento che connette il front-end della applicazione, ed il rispettivo back-end. Grazie ai suoi standard aperti e alla sua natura leggera, le API REST possono essere **facilmente integrate** con diverse piattaforme, linguaggi di programmazione e framework. Consentono di realizzare il **principio di separazione tra client e server** che permette una maggiore modularità e manutenibilità dell'architettura.

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

Le scelte di progetto nello sviluppo software si riferiscono alle decisioni prese durante la pianificazione e progettazione di un'applicazione. Queste decisioni riguardano l'architettura del sistema, la scelta del linguaggio di programmazione, delle librerie e dei framework da utilizzare, il design dell'interfaccia utente e molti altri aspetti. Le scelte di progetto influenzano la scalabilità, la manutenibilità e le prestazioni del software, nonché la sua facilità di utilizzo da parte degli utenti finali.

# TECNICHE DI SVILUPPO

Conclusa la progettazione, risulta importante focalizzarsi sulla scelta delle tecnologie da adoperare al fine di muovere i primi passi verso l’implementazione.

##### HTML, JAVASCRIPT



Figura 6.1 HTML Logo

HTML, AJAX e JavaScript sono fondamentali per lo sviluppo del frontend di un'applicazione web. HTML (HyperText Markup Language) viene utilizzato per creare la struttura e il layout della pagina, definendo gli elementi come titoli, paragrafi, tabelle e form. JavaScript, invece, permette di rendere la pagina interattiva, manipolando gli elementi HTML, gestendo gli eventi e implementando logiche di business.

##### VISUAL STUDIO CODE

##### Immagine che contiene schermata, Elementi grafici, simbolo, linea Descrizione generata automaticamente

##### Figura 6.2 Visual Studio Code Logo

##### Visual Studio è uno strumento di sviluppo potente che è possibile usare per completare l'intero ciclo di sviluppo in un'unica posizione. Si tratta di un ambiente di sviluppo integrato completo (IDE) che è possibile usare per scrivere, modificare, eseguire il debug e compilare codice e quindi distribuire l'app. Oltre alla modifica e al debug del codice, Visual Studio include compilatori, strumenti di completamento del codice, controllo del codice sorgente, estensioni e molte altre funzionalità per migliorare ogni fase del processo di sviluppo software. Grazie alla varietà di funzionalità e linguaggi supportati in Visual Studio, è possibile passare dalla scrittura del primo programma "Hello World" allo sviluppo e alla distribuzione di app. Ad esempio, compilare, eseguire il debug e testare le app .NET e C++, modificare ASP.NET pagine nella visualizzazione progettazione Web, sviluppare app per dispositivi mobili e desktop multipiattaforma con .NET o creare interfacce utente Web reattive in C#.

##### THYMELEAF



Figura 6.2 Thymeleaf Logo

Thymeleaf è un motore di template per Java che consente di integrare la logica del backend con il frontend in modo semplice ed efficace. Utilizzando Thymeleaf, è possibile incorporare tag ed espressioni all’interno di file HTML per generare dinamicamente il contenuto della pagina. Questo permette di manipolare i dati provenienti dal server e renderizzarli direttamente nel template HTML. Thymeleaf supporta anche funzionalità avanzate come cicli, condizioni e iterazioni, consentendo una gestione dinamica e flessibile del frontend. Inoltre, Thymeleaf offre un’ottima integrazione con framework come Spring, rendendolo una scelta popolare per lo sviluppo web basato su Java.

##### MYSQL



Figura 6.3 MySQL Logo

MySQL è un sistema di gestione di database relazionali ampiamente utilizzato nel backend delle applicazioni web. Per utilizzare MySQL nel backend, è necessario stabilire una connessione al database utilizzando le credenziali di accesso corrette. Una volta stabilita la connessione, è possibile eseguire query per inserire, recuperare, aggiornare o eliminare dati dal database. MySQL offre un’ampia gamma di funzionalità, tra cui l’indicizzazione dei dati, le transazioni, le viste e molto altro, che consentono di gestire e manipolare i dati in modo efficiente. È possibile utilizzare librerie o framework come JDBC o ORM (Object-Relational Mapping) per semplificare l’interazione con il database MySQL nel backend delle applicazioni web.

##### XAMPP



Figura 6.4 XAMPP Logo

XAMPP è una distribuzione di software open-source che semplifica l'installazione e la configurazione di un ambiente di sviluppo per il backend. Utilizzando XAMPP, è possibile eseguire localmente un server web (Apache), un database (MySQL), un interprete di scripting (PHP) e altro ancora, il tutto in un'unica applicazione. Questo consente agli sviluppatori di creare e testare applicazioni backend senza dover configurare manualmente ciascun componente separatamente. XAMPP offre un'interfaccia utente intuitiva per la gestione dei servizi e delle configurazioni, semplificando l'avvio e la sospensione del server. Inoltre, XAMPP è multi-piattaforma e supporta sistemi operativi come Windows, macOS e Linux, rendendolo uno strumento versatile per lo sviluppo back-end su macchine diverse.

##### POSTMAN



Figura 6.5 Postman Logo

Postman è uno strumento di sviluppo API che consente di testare, documentare e collaborare con le API nel processo di sviluppo backend. Utilizzando Postman, è possibile inviare richieste http personalizzate agli endpoint delle API, visualizzarne le risposte e analizzare i dati restituiti. Inoltre, Postman offre funzionalità avanzate come l’automazione dei test, la generazione di documentazione API e la condivisione di collezioni di richieste con il team di sviluppo. È uno strumento essenziale per la fase di testing e debugging delle API, consentendo agli sviluppatori di garantire la correttezza e la robustezza delle loro applicazioni backend.

##### DOCKER



Figura 6.6 Docker Logo

Docker è una piattaforma di containerizzazione che consente di creare e gestire facilmente ambienti di sviluppo e produzione per il backend delle applicazioni. Utilizzando Docker per il backend, è possibile creare immagini dei componenti software, inclusi il server web, il database e altre dipendenze, in modo da poterli eseguire in modo isolato e riproducibile su diversi ambienti. Docker semplifica il processo di distribuzione delle applicazioni, garantendo che l’ambiente di sviluppo sia identico a quello di produzione, riducendo così i problemi di compatibilità. Inoltre, Docker permette di scalare facilmente l’applicazione in base al carico di lavoro, sia verticalmente che orizzontalmente, grazie alla gestione dei container.

##### HIBERNATE



Figura 6.7 Hibernate Logo

Hibernate è un framework di Object-Relational Mapping (ORM) che semplifica l’interazione con il database nel backend delle applicazioni. Utilizzando Hibernate, è possibile mappare gli oggetti Java alle tabelle del database in modo trasparente, senza la necessità di scrivere query SQL manualmente. Hibernate gestisce automaticamente la persistenza degli oggetti nel database, permettendo di eseguire operazioni CRUD (Create, Read, Update, Delete) in modo semplice e intuitivo. Inoltre, Hibernate gestisce le relazioni tra gli oggetti, consentendo di definire associazioni come uno-a-uno, uno-a-molti e molti-a-molti. Grazie a Hibernate, le operazioni di accesso al database diventano più semplici e manutenibili, consentendo agli sviluppatori di concentrarsi sulla logica di business dell’applicazione.

##### MAVEN



Figura 6.8 Maven Logo

Maven è uno strumento di gestione delle dipendenze e di automazione della compilazione per i progetti software. Utilizzando Maven, è possibile definire il progetto tramite un file di configurazione denominato “pom.xml”, specificando le dipendenze esterne necessarie per il progetto. Maven scaricherà automaticamente le dipendenze richieste e le includerà nel progetto. Inoltre, esso semplifica la compilazione, il testing e il packaging del progetto, consentendo di generare file JAR, WAR o altri formati. Maven offre anche la possibilità di definire script per eseguire compiti personalizzati, come la pulizia del progetto o l’esecuzione di test specifici. In generale, Maven semplifica notevolmente il processo di gestione e compilazione dei progetti software, fornendo una struttura coerente e prevedibile per lo sviluppo.

##### VISUAL PARADIGM



Figura 6.9 Visual Paradigm Logo

Visual Paradigm è uno strumento di modellazione e progettazione che consente di creare diagrammi UML (Unified Modeling Language) e diagrammi di flusso per il processo di sviluppo software. Utilizzando Visual Paradigm, è possibile visualizzare l'architettura e la struttura del sistema, definire classi, interfacce, relazioni e altre entità concettuali. Inoltre, esso supporta la generazione automatica di codice da modelli UML, facilitando l'implementazione del software. Il tool offre anche funzionalità per la collaborazione e la gestione dei requisiti, consentendo a team di sviluppatori di lavorare in modo coordinato e di mantenere traccia delle modifiche nel ciclo di sviluppo. In sintesi, Visual Paradigm aiuta a creare modelli e documentazione di qualità per il processo di sviluppo software.

# MODELLO DEI CASI D’USO

## Componenti principali

Relativamente al task da implementare, è possibile definire la seguente classificazione.

* *Attori primari*:
  + Amministratore;
  + Utente registrato;
  + Utente;
* *Attori secondari*:
  + Browser Web;
  + Repository.
* *Casi d’uso*:
  + **UC1**: Registrazione;
  + **UC2**: Login;
  + **UC3**: Visualizza elenco;
  + **UC4**: Avvia partita;
  + **UC5**: Salva partita;
  + **UC6**: Scelta robot;
  + **UC7**: Scelta classe;
  + **UC8­:** Download classe;
  + **UC9:** Gestione classi;
  + **UC10:** Modifica classe;
  + **UC11:** Aggiungi classe;
  + **UC12:** Elimina classe.

## Diagramma dei Casi d’Uso

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

## Scenari Casi d’Uso.

A conclusione di tale capitolo, segue un'analisi approfondita di ciascun caso d'uso con indicazioni sugli attori (primari e secondari), una breve descrizione, pre e post-condizioni, sequenze di eventi principali e alternative.

**UC1: Login**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Login** |
| **Attore primario** | Utente registrato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | L’utente inserisce i dati per il login |
| **Pre-Condizioni** | L’utente deve essere registrato |
| **Sequenza di eventi Principale** | 1. L’utente inserisce username e password per il login. |
| **Sequenza di eventi**  **alternativi** | 1. Se l’utente inserisce un username o password sbagliata, l’applicazione restituisce un messaggio di errore. |
| **Post-Condizioni** | L’utente registrato effettua il login con successo. |

**UC2: View Classes**

|  |  |
| --- | --- |
| **Caso d’uso:** | **View Classes** |
| **Attore primario** | Utente autenticato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Il giocatore visualizza l’elenco delle classi Java da testare |
| **Pre-Condizioni** | L’elenco delle classi da testare non è vuoto |
| **Sequenza di eventi Principale** | 1. Il giocatore seleziona l’opzione “Visualizza Elenco” sulla home page dell’applicazione. 2. Il sistema mostra l’elenco delle Classi Java da testare presenti nel repository. |
| **Post-Condizioni** | L’elenco viene correttamente mostrato dall’interfaccia web dell’applicazione. |

**UC3: Avvia partita**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Avvia partita** |
| **Attore primario** | Utente autenticato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Il giocatore seleziona le impostazioni della partita. |
| **Pre-Condizioni** | L’elenco delle classi da testare non è vuoto. |
| **Sequenza di eventi Principale** | 1. Il giocatore seleziona la classe Java da testare. 2. Il giocatore seleziona il robot con cui competere. 3. Il giocatore clicca su “Avvia partita” con le impostazione da lui selezionate. |
| **Post-Condizioni** | La partita viene avviata con successo. |

**UC4: choose robot**

|  |  |
| --- | --- |
| **Caso d’uso:** | **choose robot** |
| **Attore primario** | Utente autenticato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Il giocatore seleziona il robot con cui competere. |
| **Pre-Condizioni** | La lista dei robot non è vuota. |
| **Sequenza di eventi Principale** | 1. Il giocatore seleziona un robot tra quelli disponibili. |
| **Post-Condizioni** | Il robot è stato selezionato correttamente. |

**UC5: choose class**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Scelta classe** |
| **Attore primario** | Utente autenticato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Il giocatore seleziona la classe da testare. |
| **Pre-Condizioni** | La lista delle classi non è vuota. |
| **Sequenza di eventi Principale** | 1. Il giocatore seleziona una classe tra quelle disponibili. |
| **Post-Condizioni** | Il robot è stato selezionato correttamente. |

**UC6: Download class**

|  |  |  |
| --- | --- | --- |
| **Caso d’uso:** |  | **Download class** |
| **Attore primario** |  | Utente autenticato, Amministratore |
| **Attore secondario** |  | Browser Web, Repository |
| **Descrizione** |  | Il giocatore effettua il download di una delle classi Java disponibili |
| **Pre-Condizioni** |  | L’elenco delle classi da testare non è vuoto |
| **Sequenza di eventi Principale** |  | 1. Il giocatore o l’amministratore seleziona la classe Java da scaricare. 2. Il giocatore o l’amministratore sceglie l’opzione “Download” per la classe selezionata 3. Il sistema fornisce il file .java della classe scelta. |
| **Post-Condizioni** |  | Il file .java viene correttamente scaricato, dal repository, nella cartella Download. |

**UC7: Update class**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Modifica classe** |
| **Attore primario** | Amministratore |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | L’amministratore aggiunge una nuova classe |
| **Pre-Condizioni** | La classe è già presente del database |
| **Sequenza di eventi principale** | 1. L’amministratore seleziona la classe da modificare. 2. L’amministratore apporta le modifiche alla classe e le salva. |
| **Post-Condizioni** | La lista delle classi è aggiornata con la classe modificata |

**UC8: Upload class**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Upload class** |
| **Attore primario** | Amministratore |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | L’amministratore aggiunge una nuova classe |
| **Pre-Condizioni** | La classe non è già presente del database |
| **Sequenza di eventi principale** | 1. L’amministratore fa l’upload della nuova classe |
| **Post-Condizioni** | La lista delle classi è aggiornata con la nuova classe. |

**UC9: Delete class**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Delete class** |
| **Attore primario** | Amministratore |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | L’amministratore elimina una classe tra quelle disponibili |
| **Pre-Condizioni** | La classe è già presente del database |
| **Sequenza di eventi principale** | 1. L’amministratore seleziona una classe 2. L’amministratore clicca su “Elimina classe” |
| **Post-Condizioni** | La lista delle classi è aggiornata con la classe eliminata |

**UC10: Search classes**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Search classes** |
| **Attore primario** | Utente autenticato |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Ricerca si una classe in base a certe caratteristiche |
| **Pre-Condizioni** | La classe è già presente del database |
| **Sequenza di eventi principale** | 1. L’utente clicca su “Search classes”. 2. L’utente inserisce le caratteristiche della classe da cercare. 3. L’applicazione gli mostra le classi con quella caratteristica. |
| **Post-Condizioni** | L’applicazione restituisce le classi con le caratteristiche inserite dall’utente |

**UC11: Manage Classes**

|  |  |
| --- | --- |
| **Caso d’uso:** | **Manage Classes** |
| **Attore primario** | Amministratore |
| **Attore secondario** | Browser Web, Repository |
| **Descrizione** | Il giocatore visualizza l’elenco delle classi Java da testare |
| **Pre-Condizioni** | L’elenco delle classi da testare non è vuoto |
| **Sequenza di eventi Principale** | 1. L’amministratore seleziona l’opzione “Manage classes” sulla home page dell’applicazione. 2. Il sistema mostra l’elenco delle Classi Java da testare presenti nel repository. |
| **Post-Condizioni** | L’elenco viene correttamente mostrato dall’interfaccia web dell’applicazione. |

# SYSTEM DOMAIN MODEL

Il System Domain Model **aiuta a comprendere e definire i concetti chiave, le entità, le relazioni** e i comportamenti all'interno del dominio specifico del sistema. Un System Domain Model è un modello concettuale di un sistema che **descrive le varie entità** che fanno parte o hanno rilevanza nel sistema e le loro relazioni. Il modello di dominio è utile per mettere a fuoco i concetti fondamentali di un sistema e definire un vocabolario specifico per il sistema. In particolare, il System Domain Model rappresenta il **modello di dominio di un sistema software**.

Il System Domain Model rappresentato comprende tre entità principali: Classe, File e Response.

La relazione fra Classe e File è di tipo 1:1 ed è collegata tramite la chiave esterna fileid in Classe e il campo id presente in File.

Gli attributi di tipo enumerazione dell’entità Classe sono due: Level e Opponent. Entrambi si compongono di una stringa selezionabile fra quattro alternative per il tipo Level e due alternative per il tipo Opponent. L’entità Response invece non corrisponde a nessuna tabella del database me è un oggetto utilizzato dal software come tipo di ritorno di alcuni metodi.

Immagine che contiene testo, diagramma, schermata, Piano

Descrizione generata automaticamente

Figura 8.1 System Domain Model.

# MODELLAZIONE DEI DATI

Il diagramma di modellazione dei dati è un tipo di diagramma che **rappresenta la struttura di un database e le relazioni tra le tabelle**. Questo tipo di diagramma è utilizzato per rappresentare i dati e le relazioni tra di essi in modo visuale e intuitivo. In particolare, il diagramma di modellazione dei dati può essere utilizzato **per definire la struttura di un database relazionale e per descrivere le relazioni tra le tabelle**. Il diagramma di modellazione dei dati può essere utilizzato anche per definire la struttura di altri tipi di database come i database ad oggetti.

Il database risulta essere composto da due tabelle, ossia collezioni di dati organizzata in righe e colonne possedenti **una chiave primaria.**

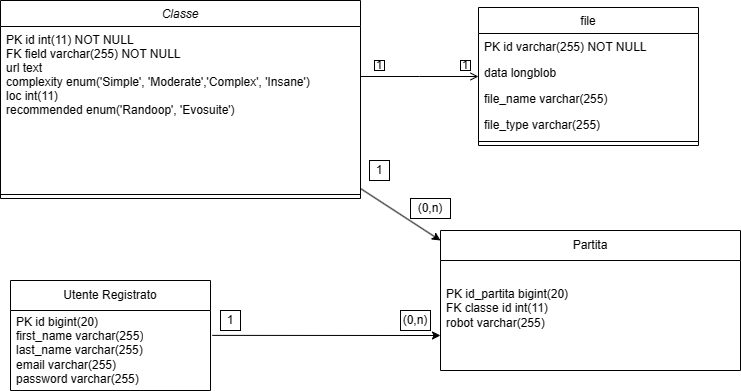


Figura 9.1 Diagramma Modellazione dei Dati.

# CONTEXT DIAGRAM

Il Context Diagram è un **ulteriore diagramma ad alto livello** che rappresenta il contesto operativo del sistema e mette in evidenza le interazioni con gli attori esterni. Fornisce, dunque, **una visione panoramica dell'architettura senza entrare nei dettagli interni del sistema**.

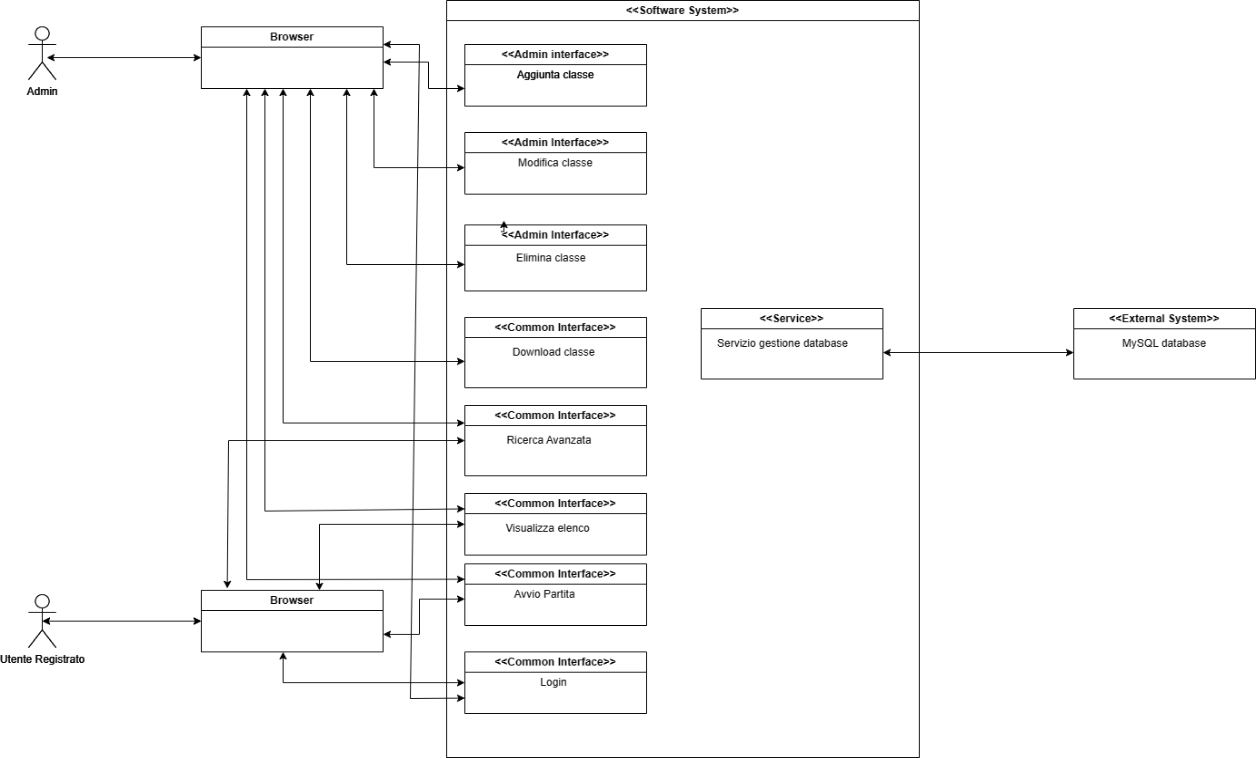


Figura 10.1 Context Diagram.

# COMPONENT DIAGRAM

Il diagramma dei componenti viene utilizzato per **descrivere l'architettura software** di un sistema, esso mostra i componenti del sistema e le relazioni tra di essi.

Immagine che contiene diagramma, Piano, Disegno tecnico, testo

Descrizione generata automaticamente

Di seguito inoltre vengono mostrati il contenuto dei package View, Model e templates.

Immagine che contiene testo, schermata, design

Descrizione generata automaticamente

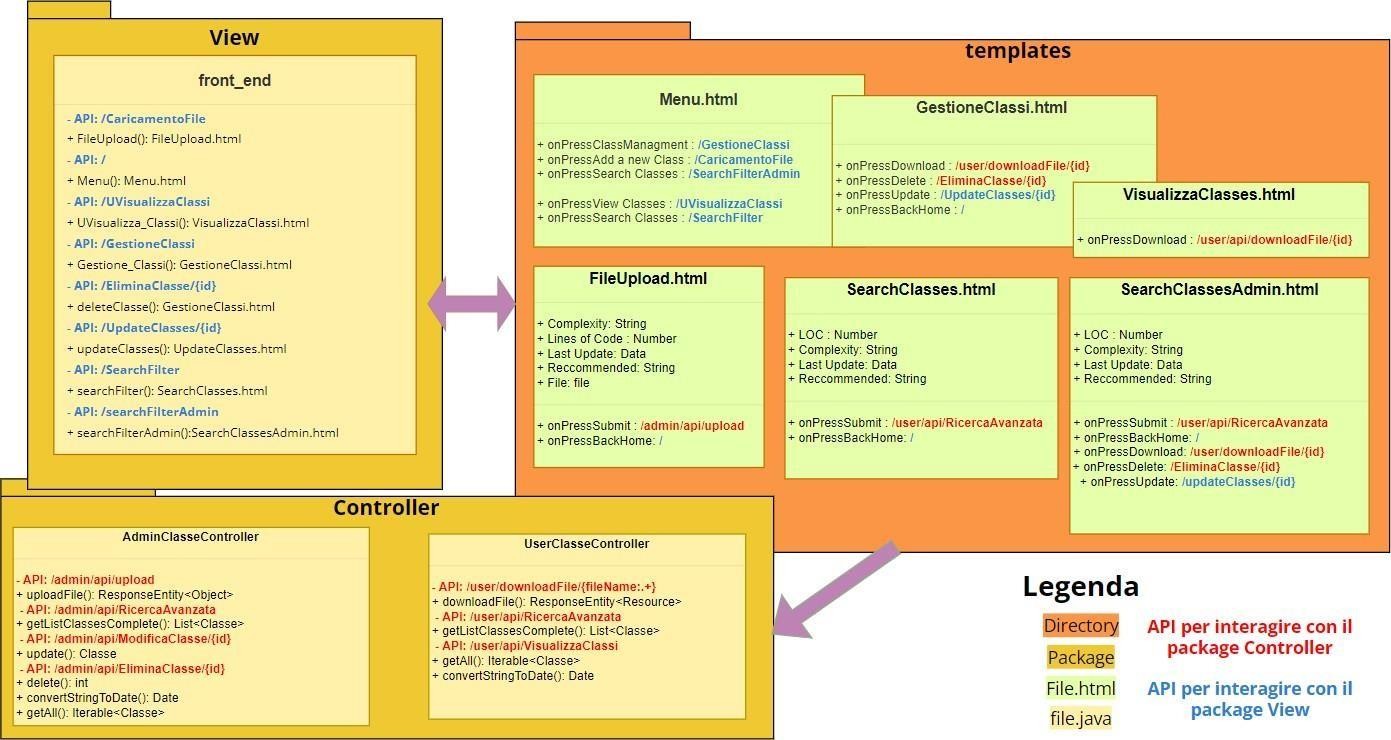
# PACKAGE DIAGRAM

Un Package Diagram è un tipo di diagramma UML che permette di descrivere la struttura di un sistema attraverso dei grafi in cui i nodi rappresentano i package e gli archi le dipendenze tra di essi. Lo scopo dei diagrammi dei package UML è organizzare gli elementi in gruppi per fornire una struttura migliore per un modello di sistema e illustrare le dipendenze tra i diversi package di questo sistema. Un package è il blocco predefinito di un diagramma e rappresenta un raggruppamento di elementi di un modello. Questi elementi possono essere diagrammi, documenti, classi o anche altri package.

Immagine che contiene testo, schermata, Rettangolo, Stampa

Descrizione generata automaticamente

Di seguito si riporta una rappresentazione estesa del contenuto dei package View e Controller e della cartella contente i file .html, per rendere più chiara l’interazione tra essi. I file .html all’interno contengono script in linguaggio JavaScript che richiamano i metodi del controller o della classe front\_end tramite le API REST. Analogamente i metodi contenuti nella classe front\_end del package View sono richiamati dal Browser Web attraverso le corrispondenti API, e in base al metodo richiamto viene selezionato l’apposito file HTML.



# API

## Introduzione e proprietà

Le REST API, acronimo di "Representational State Transfer Application Programming Interface", sono un insieme di principi e protocolli che consentono la comunicazione e lo scambio di dati tra sistemi software. Le REST API si basano sull'architettura REST, che è un approccio comune nello sviluppo di servizi web. I vantaggi delle REST API nel contesto dello sviluppo software sono diversi:

* + - **Scalabilità e flessibilità**: Le REST API permettono di creare servizi web altamente scalabili, in grado di gestire un gran numero di richieste simultanee da parte degli utenti. L'architettura REST si basa surisorse indipendenti, consentendo l'aggiunta o la rimozione di risorse senza influire sulle altre. Le REST API consentono di distribuire il carico su più server o risorse. Ciò significa che è possibile allocarele risorse in modo ottimale, garantendo che ogni richiesta venga gestita in modo efficiente e senza sovraccaricare un singolo server.
    - **Interoperabilità**: Le REST API utilizzano protocolli di comunicazione standard come HTTP e HTTPS, che sono ampiamente supportati e comprensibili da diverse tecnologie e piattaforme. Ciò consente a sistemi diversi di comunicare e scambiare dati senza problemi.
    - **Separazione dei concetti**: Le REST API seguono il principio di separazione tra client e server. Ciò significa che il server fornisce l'accesso ai dati e ai servizi, mentre il client è responsabile della presentazione dei dati e dell'interazione con l'utente. Questa separazione permette di sviluppare e modificare indipendentemente il client e il server, facilitando la **manutenzione** e l'**evoluzione** del sistema.
    - **Semplicità**: Le REST API utilizzano una serie di operazioni standard, come GET, POST, PUT e DELETE, per accedere e manipolare le risorse. Questo rende l'interfaccia delle API intuitiva e facile da usare per gli sviluppatori, riducendo la curva di apprendimento e accelerando lo sviluppo delle applicazioni.

Le applicazioni delle REST API sono ampie e diverse. Alcuni esempi comuni includono:

* + - **Integrazione di sistemi**: Le REST API consentono a diverse applicazioni o sistemi di comunicare tra loro, scambiando dati e informazioni. Ad esempio, un'applicazione mobile può utilizzare le REST APIper inviare richieste a un server remoto e ottenere i dati necessari per fornire una funzionalità specifica.
    - **Sviluppo di servizi web**: Le REST API sono ampiamente utilizzate per lo sviluppo di servizi web che forniscono accesso a dati e funzionalità tramite Internet. Le applicazioni possono utilizzare queste API per ottenere informazioni da diverse fonti, come social media, servizi di pagamento o dati di terzeparti.
    - **Creazione di applicazioni distribuite**: Le REST API consentono lo sviluppo di applicazioni distribuite, in cui diverse parti del sistema possono essere sviluppate indipendentemente e comunicare tra loro tramite API. Questa flessibilità consente di creare applicazioni complesse, in cui diverse funzionalitàpossono essere sviluppate separatamente e integrate successivamente.

In sintesi, le REST API offrono una soluzione **efficiente** e **scalabile** per la comunicazione tra sistemi software, consentendo lo sviluppo di applicazioni distribuite, l'integrazione di servizi e l'accesso a dati e funzionalità di terze parti. Grazie alla loro **semplicità** e **flessibilità,** le REST API sono diventate uno degli standard più diffusi nello sviluppo di applicazioni web e mobile.

## Descrizione formale delle REST API

In questa sezione si fornisce la descrizione contenente le informazioni essenziali sulle API implementate, inclusi i percorsi (paths) e i relativi metodi HTTP consentiti, nonché i parametri richiesti o opzionali per ciascun percorso.

La documentazione delle API è di fondamentale importanza per aiutare gli sviluppatori a comprendere come interagire correttamente con un software attraverso le API. Essa fornisce una guida chiara e strutturata sulle funzionalità disponibili, sui parametri richiesti, sui formati di dati accettati e sulle eventuali restrizioni. Attraverso questa descrizione delle API, gli sviluppatori possono esplorare i percorsi disponibili, comprendere i metodi consentiti (come GET, POST, PUT e DELETE) e i parametri richiesti per ciascuna richiesta. Questo aiuta gli sviluppatori a integrare correttamente le tue API nel loro software, inviando richieste valide e gestendo le risposte in modo appropriato.

## TEST REST API in AdminClassController

Di seguito si riporta il testing delle API di AdminClassController:

• /admin/api/upload

Immagine che contiene schermata, testo, software, Software multimediale

Descrizione generata automaticamente

• /admin/api/VisualizzaClassi

Immagine che contiene schermata, testo, software, Software multimediale

Descrizione generata automaticamente

• /admin/api/ModificaClasse/{id}

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

• /admin/api/MatchStart

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

• /admin/api/EliminaClasse/{id}

Immagine che contiene schermata, testo, software, Software multimediale

Descrizione generata automaticamente

• /admin/api/RicercaAvanzata

Immagine che contiene schermata, testo, software, Software multimediale

Descrizione generata automaticamente

## TEST REST API in UserClassController

• /admin/api/downloadFile/{filename:.+}

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

# SEQUENCE DIAGRAMS

I diagrammi di sequenza sono uno strumento di **modellazione utilizzato per rappresentare l'interazione tra oggetti o componenti** di un sistema nel tempo. Sono composti da una linea verticale che rappresenta il tempo e da frecce che mostrano il flusso di controllo tra gli oggetti. I diagrammi di sequenza consentono di **visualizzare le azioni, i messaggi e le chiamate di metodo** tra gli oggetti, fornendo una panoramica chiara del comportamento dinamico del sistema. Sono utili per **comprendere l'ordine delle operazioni**, le dipendenze temporali e le collaborazioni tra gli elementi del sistema.

Si usano inoltre per rappresentare l'interazione tra i componenti del **pattern Model-View-Controller (MVC)**. Essi consentono di visualizzare chiaramente come avvengono le chiamate di **metodo e i messaggi tra il Model, la View e il Controller**. Questo aiuta a comprendere il flusso delle operazioni e le dipendenze tra i componenti, facilitando la progettazione e la comprensione del comportamento dinamico del sistema. I **diagrammi di sequenza possono evidenziare l'ordine temporale** delle azioni, inclusi gli eventi scatenati dall'utente, le richieste del Controller al Model e il successivo aggiornamento della View.

## VisualizzaClassi e DownloadCodiceClasse

Immagine che contiene schermata, diagramma, design

Descrizione generata automaticamente

## RicercaAvanzata

Immagine che contiene testo, diagramma, Piano, Parallelo

Descrizione generata automaticamente

## GestioneClasse

Immagine che contiene testo, diagramma, Parallelo, linea

Descrizione generata automaticamente

## AggiuntaClasse

Immagine che contiene schermata, diagramma

Descrizione generata automaticamente

## Login

Immagine che contiene schermata, diagramma, design

Descrizione generata automaticamente

## Avvio Partita

Immagine che contiene testo, schermata, Parallelo, diagramma

Descrizione generata automaticamente

# DEPLOYMENT

Il Deployment, nell'ambito del software e delle tecnologie dell'informazione, si riferisce al processo di **distribuzione e messa in esercizio di un'applicazione** o di un sistema informatico in un ambiente operativo. È **una fase cruciale dello sviluppo software** in cui il prodotto finale viene reso disponibile agli utenti finali.

Durante il deployment, l'applicazione o il sistema viene installato, configurato e attivato su uno o più server o dispositivi di destinazione. Questo può **includere sia l'hardware fisico sia le risorse virtuali,** come i server cloud. L'obiettivo principale del deployment è consentire agli utenti di interagire con l'applicazione o il sistema in modo efficiente e affidabile.

Il processo di deployment può coinvolgere molteplici attività, come la **preparazione dell'ambiente di produzione, la gestione delle dipendenze e delle configurazioni, il controllo di versione, il testing** delle funzionalità, la configurazione dei parametri di sicurezza e molto altro. È importante garantire che l'applicazione o il sistema siano pronti per l'uso in termini di prestazioni, affidabilità e sicurezza. Il deployment può essere automatizzato utilizzando strumenti e pratiche di Continuous Integration e Continuous Deployment (CI/CD), che consentono di ridurre gli errori umani e di semplificare il processo complessivo. Inoltre, le moderne architetture di deployment, come i contenitori e le tecnologie di orchestrazione, **come Docker-Compose**, offrono una maggiore scalabilità e flessibilità per gestire ambienti complessi e distribuiti.

In conclusione, **il deployment è un processo essenziale** per portare un'applicazione o un sistema informatico dallo sviluppo all'ambiente di produzione, garantendo che sia configurato correttamente, funzioni in modo **affidabile** e soddisfi le esigenze degli utenti finali. Un deployment efficace richiede pianificazione, automazione, monitoraggio e gestione continua per garantire il successo e il corretto funzionamento dell'applicazione o del sistema.

Nella seguente sezione, si descrive **la procedura di deployment utilizzata per distribuire il software in oggetto**:

1. Aprire **Docker Desktop**.
2. Dal seguente link **GitHub**, scaricare la versione aggiornata del progetto. <https://github.com/Testing-Game-SAD-2023/T1-G3>
3. Estrarre il file **zip** scaricato (seguire la seguente guida in caso di difficoltà).

[*https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-*](https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-8294-e1d3-703ed85e7ebc#%3A~%3Atext%3DApri%20Esplora%20file%20e%20trova%2Csulla%20cartella%20compressa%20per%20aprirla)[*8294-e1d3-*](https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-8294-e1d3-703ed85e7ebc#%3A~%3Atext%3DApri%20Esplora%20file%20e%20trova%2Csulla%20cartella%20compressa%20per%20aprirla)

[*703ed85e7ebc#:~:text=Apri%20Esplora%20file%20e%20trova,sulla%20cartella%20compressa%20*](https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-8294-e1d3-703ed85e7ebc#%3A~%3Atext%3DApri%20Esplora%20file%20e%20trova%2Csulla%20cartella%20compressa%20per%20aprirla)[*per%20aprirla*](https://support.microsoft.com/it-it/windows/comprimere-e-decomprimere-file-f6dde0a7-0fec-8294-e1d3-703ed85e7ebc#%3A~%3Atext%3DApri%20Esplora%20file%20e%20trova%2Csulla%20cartella%20compressa%20per%20aprirla)

1. Aprire un terminale da amministratore e posizionarsi sul percorso dove è stato estratto il progetto, successivamente digitare:

> docker-compose up

Il comando viene utilizzato per avviare i servizi definiti in un file di configurazione docker- compose.yml già presente all'interno della cartella. Viene creata l’immagine del container e viene

eseguito il running all'intero di docker.

Per visualizzare la corretta esecuzione del comando digitare docker-compose ps che fornisce lo stato dei container definiti nel file docker-compose.yml.

1. Arrivati a questo punto l'applicazione è utilizzabile senza ulteriori sforzi, basterà recarsi sul proprio

**browser web** e digitare il seguente comando: http://localhost:8080/.

Attraverso il menù laterale sarà possibile aprire schermate diverse con le diverse funzionalità richieste al task T1.

1. Per la gestione e configurazione del database è possibile scegliere fra due strade alternative. La prima è rappresentata da *Database Client* e rappresenta l'alternativa consigliata per semplicità e usabilità. Per scaricare questa estensione, aprire VSCode (se non si è in possesso recarsi al seguente sito per scaricare la versione aggiornata: [https://www.dz-techs.com/it/windows-vs-code-set-up-](https://www.dz-techs.com/it/windows-vs-code-set-up-guide) [guide](https://www.dz-techs.com/it/windows-vs-code-set-up-guide)), e dal menù laterale premere sull'icona "Extension", ricercare e scaricare dalla barra di ricerca l'estensione *Database Client*.

Una volta scaricata l'estensione, chiudere la finestra di VSCode e riavviare il PC, al termine del riavvio aprire una nuova finestra di VSCode; sul menù laterale dovrebbe essere apparsa una sezione database, cliccare quella sezione e accedere al database chiamato db\_classi.

Tramite i diversi comandi MySQL (alter table, create table, …) è possibile modificare le

tabelle classe e file o crearne di nuove. Cliccare “Execute” per renderle attive.

La seconda alternativa invece, prevede di agire direttamente dall'app Docker. Digitare docker exec -it <nome\_container o ID\_container> bash per entrare all'interno di un container Docker in esecuzione e avviare una shell interattiva al suo interno.

Successivamente digitare mysql -u root -p per accedere all'interfaccia della riga di comando di MySQL e connettersi al database "db\_classi" utilizzando l'utente root.

Alternativamente digitare interamente docker exec -it my-mysql-container mysql

-u root -p db\_classi.

Questo comando si connetterà direttamente all'interfaccia della riga di comando di MySQL all'interno del container my-mysql-container e selezionerà il database db\_classi. Verrà richiesto di inserire la password per l'utente root. Nel caso specifico con la password vuota, si può semplicemente premere nuovamente Invio senza inserire alcuna password.

ATTENZIONE: è necessario lasciare il database attivo durante tutta la durata delle operazioni di gestione del database

* 1. INSTALL VIEW

Immagine che contiene schermata, diagramma, Rettangolo, testo

Descrizione generata automaticamente

* 1. DIAGRAMMA DI DEPLOYMENT

Il diagramma di deployment è una **rappresentazione grafica** dell'architettura di un sistema software che illustra come i suoi **componenti hardware e software** sono distribuiti e interconnessi. Mostra la disposizione fisica dei nodi di elaborazione, come server, client, dispositivi mobili, e le connessioni tra di essi. Il **diagramma evidenzia la distribuzione delle applicazioni**, dei servizi e delle risorse di rete all'interno del sistema, consentendo una visione chiara delle relazioni tra i componenti e dei flussi di dati.

Come illustrato sul diagramma, la piattaforma di containerizzazione è Docker che al suo interno possiede due container: Il primo contenente il **Server MySQL**, il secondo con l’immagine dell’applicazione web.

I Container lavorano su un sistema operativo Windows 11 e comunicano con il client per mezzo del porto **80 attraverso il protocollo HTTP**. Il client si collega su tale porto attraverso il browser Web a disposizione delle macchine UniNA utilizzate.

Immagine che contiene Rettangolo, testo, diagramma, schermata

Descrizione generata automaticamente

# 16. TESTING

Il testing è un processo essenziale nello sviluppo del software che mira a valutare la qualità, l'affidabilità e il corretto funzionamento di un sistema o di un'applicazione. Durante il testing, vengono eseguite una serie di attività per identificare difetti, errori e problemi nel software. Ciò viene fatto attraverso l'esecuzione di input specifici, la comparazione dei risultati attesi con quelli effettivi e l'analisi del comportamento del sistema in diversi scenari. I test possono essere automatizzati o eseguiti manualmente e includono test di unità, test di integrazione, test funzionali, test di regressione e molti altri. L'obiettivo finale del testing è garantire che il software soddisfi i requisiti stabiliti e funzioni in modo affidabile per gli utenti finali.

Per il testing dell’interfaccia, vanno eseguiti test funzionali per verificare che tutte le funzionalità della GUI

siano implementate correttamente e che rispondano in modo adeguato agli input.

Le classi che popolano il database sono state prelevate dalla repository del Team Software Architecture Design. [Path: Documenti/General/EsempiDiClassiDaTestare/generated\_tests\_2016.zip]

Di seguito vengono riportati i link a cui si fa riferimento nella colonna input:

1. http://localhost:8080/Admin\_ManageClass
2. http://localhost:8080/Admin\_UploadClass
3. http://localhost:8080/Admin\_AdvancedSearchClasses
4. http://localhost:8080/User\_ViewClass
5. <http://localhost:8080/User_SearchClasses>
6. <http://localhost:8080/Admin_ViewClasses>
7. <http://localhost:8080/Admin_UpdateClasses/1>
8. http://localhost:8080/User\_Play

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Test Case***  ***Id*** | ***Descrizione*** | ***Precondizio ni*** | ***Input*** | ***Output Ottenuti*** | ***Postcondizioni Ottenute*** | ***Esito (FAIL;***  ***PASS)*** |
| 1 CM | L’amministrator  e visualizza  l’elenco delle  classi disponibili | C’è almeno una classe nel database. | L’amministratore accede  al link [1] | Restituzione a video elenco classi. | L’amministrat ore riesce a visualizzare le  classi disponibili | PASS |
| 2 CM | L’amministrator  e visualizza il  database vuoto. | Non ci sono  classi nel database | L’amministrator  e accede al link:  [1] | Restituzione a  video di elenco vuoto. | / | PASS |
| 1 D | L’amministrator e fa il download di una delle classi disponibili. | C’è almeno una classe nel database. | L’amministatore nella pagina [1] o [3] o [6] preme sul tasto “Download”. | File java scaricato nei Download del Browser Web. | L’amministrat ore riesce ad accedere al  download del file. | PASS |
| 1 U | Si modificano i parametri di una classe disponibile. | C’è almeno una classe nel database. | L’amministatore nella pagina [1] preme sull’icona della matita blu.  L’amministrator e accede alla pagina [7] e riconfigura i parametri della classe  selezionata. | Viene mostrato un pop-up con su scritto “Class Update  Successfully” | La classe è modificata. | PASS |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Preme poi sul pulsante  “SUBMIT” |  |  |  |
| 1 DEL | Si vuole eliminare una classe tra quelle disponibili | C’è almeno una classe nel database. | L’amministatore nella pagina [1] preme sull’icona del cestino rosso. | Viene mostrato un messaggio con su scritto “La cancellazione è andata a buon fine”. | La classe viene rimossa dal database.  Viene mostrato  l’elenco delle classi rimanenti. | PASS |
| 1ADD | Si vuole aggiungere una nuova classe. | Si accede al link [2] cliccando sulla sezione “Add a new class”. | L'amministratore accede alla pagina [2] e aggiunge I LOC della classe, complexity, Last update, ooponent recommended e il file da caricare | Viene mostrato un messaggio: “File uploaded Successfully” | La classe è presente fra quelle disponibili nel database | PASS |
| 2ADD | Si sbaglia ad inserire lines of codes: float, null, nonvalid. | Si accede al link [2] cliccando sulla sezione “Upload class”. | L'amministratore accede alla pagina [2] e aggiunge I LOC della classe di tipo float, null o non valid. | Viene mostrato un messaggio: “Insert Loc Greater than 0” | La classe non viene aggiunta al database. | PASS |
| 3ADD | Si sbaglia ad inserire last update:float, null, nonvalid. | Si accede al link [2] cliccando sulla sezione “Upload class”. | L'amministratore accede alla pagina [2] e aggiunge last update della classe di tipo float, null o non valid. | Viene mostrato un messaggio: “Insert a valid last update” | La classe non viene aggiunta al database. | PASS |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4ADD | Si sbaglia ad inserire il file java: non si inserisce. | Si accede al link [2] cliccando sulla sezione “Update class”. | L'amministratore accede alla pagina [2] e non carica un file. | Viene mostrato un messaggio: “file is not uploaded. Error Occurred ” | La classe non viene aggiunta al database. | PASS |
| 5ADD | Si sbaglia ad inserire il file java: si inserisce un file duplicato. | Si accede al link [2] cliccando sulla sezione “Upload class”.  Il file da inserire sia già presente nel database con lo stesso  nome. | L’amministratore inserisce le features della classe da cercare. | Viene mostrato un messaggio: “Url already exist ” | La classe non viene aggiunta al database. | PASS |
| 1SE | Si ricerca attraverso appositi filtri una classe specifica fra quelle disponibili | Si accede al link [3] cliccando sulla sezione “Search Class”.  La classe esiste fra quelle disponibili nel database | L’amministratore inserisce le features della classe da cercare. | Restituzione a video della classe specificata. | La classe può essere scaricata, eliminata o modificata | PASS |
| 2SE | Si ricerca attraverso appositi filtri una classe non presente fra quelle disponibili | Si accede al link [3] cliccando sulla sezione “Search  class”.  La classe non esiste fra quelle disponibili  nel database | L’amministratore inserisce le features della classe da cercare. | Restituzione a video  dell’elenco di  classi vuoto. |  | PASS |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3SE | Si ricerca attraverso appositi filtri una serie di classi fra quelle disponibili con paremetri maggiori o uguali di quelli specificati. | Si accede al link [3] cliccando sulla sezione “Search Class”.  Esiste almeno una classe fra quelle disponibili con paremetri maggiori o uguali di  quelli specificati. | L’amministratore inserisce le features della classe da cercare. | Restituzione a video delle classi che rispettano i vincoli. | La classi possono essere scaricate, eliminate o modificate. | PASS |
| 1 V | L’utente  visualizza  l’elenco delle  classi disponibili | C’è almeno una classe nel  database. | L’utente accede  al link [4] | Restituzione a video elenco classi. | L’utente riesce a visualizzare le classi  disponibili | PASS |
| 2 V | L’utente non  visualizza  l’elenco delle  classi disponibili | Non c’è alcuna classe nel database. | L’utente accede  al link [4] | Restituzione a video elenco classi vuoto. | L’utente non riesce a visualizzare le  classi disponibili | PASS |
| 1 D | L’utente fa il  download di una  delle classi disponibili. | C’è almeno  una classe  nel database. | L’utente nella pagina [4] o [5] preme sul tasto  “Download”. | File java scaricato nei  Download del Browser Web. | L’utente riesce  ad accedere al  download del file. | PASS |
| 1 P | L’utente seleziona la classe da testare e il robot tramite due menu e salva le sue scelte cliccando sul pulsante “Play” | C’e almeno una classe da testare e un robot nel database | L’utente nella pagina [8] inserisce la classe da testare e il robot avversario e clicca sul pulsante “Play” | Si avvia l’editor per testare la classe. | L’utente riesce a giocare | PASS |