

## T5 Backend:

Nome	Tipo	Modifiche
<b>Game/GameController.java</b>		
<b>GameFactoryFunction</b>	Interfaccia	Aggiunta dei parametri per la gestione della scalata.
<b>RegisterGames</b>	Metodo	Aggiunta della registrazione di scalata.
<b>StartGame</b>	Metodo	Aggiunti parametri opzionali per la scalata Aggiunta gestione della modalità "Scalata" con creazione dell'oggetto ScalataGamee restituzione dati utili a UtilEditor. Il controllo della partita esistente tiene conto se la scalata sta effettuando una transizione al round successivo.
<b>TotalScore</b>	Variabile	Fondamentale per calcolare il punteggio totale della scalata.
<b>processScalataParameters</b>	Metodo	Chiamata al metodo implementata ScalataGame.
<b>buildScalataResponse</b>	Metodo	Costruisce la risposta da inviare in caso di partita in modalità scalata.
<b>gestisciPartita</b>	Metodo	Migliorato controllo di fine partita, in modo da distinguere termine sfida e termine scalata. Aggiunta restituzione del punteggio totale della scalata.
<b>Game/GameLogic.java</b>		
<b>PlayTurn</b>	Metodo Astratto	Aggiunto il booleano isRoundEnd per verificare la fine di un round.
<b>CreateGame</b>	Metodo	Aggiornato il vecchio metodo. Aggiunto un CrateGame(int scalataID) per gestire la creazione della partita con un ID Scalata.
<b>GameLogic</b>	Classe	Aggiunta setter/getter. Ritorna il PlayerID, tipo di robot, difficoltà, classe.
<b>Game/ScalataGame.java</b>		
<b>PlayTurn</b>	Metodo	Aggiunto boolean isRound End per controllare la fine del round. Modificato per gestire la logica della scalata nei turni con l'aggiunta del controllo sullo stato della scalata ( IN_PROGRESS, WIN, LOST) e gestione dell'aggiornamento del turno.
<b>GetScore</b>	Metodo	Aggiunta dell'aggiornamento del totalScore.

<b>ScalataGame</b>	Classe	Aggiunta metodi getter/setter per id_scalata, currentRoundID, currentGameID, currentTurnID
<b>ProcessScalataParameters</b>	Metodo	Aggiunta del metodo per “normalizzare” i parametri che sono passati come array json.
<b>CheckGame</b>	Metodo	Viene fatto il controllo mediante round corrente.
<b>nextRoundTransition</b>	Attributo/ Metodo	Permette di notificare all’esterno quando la scalata passa al round successivo.
<b>toggleRoundTransition</b>	Metodo	Imposta il booleano nextRoundTransition rendendolo true quando si passa al round successivo in playTurn di ScalataGame, e false quando viene fatto il controllo al round successivo in startGame nell’HomeController.
<b>isRoundTransition</b>	Metodo	Ritorna il valore di nextRoundTransition
<b>Game/Sfida.java</b>		
<b>PlayTurn</b>	Metodo	Abbiamo effettuato la modifica perché è un metodo ereditato da GameLogic e dato che abbiamo il playTurn in GameLogic abbiamo adattato il metodo con l’aggiunta del booleano isRoundEnd.  Sono state aggiunte le chiamate a createTurn, EndRound ed endGame.
<b>GameCoverage</b>	Variabile	Aggiunta variabile per ottenere la copertura del codice tramite getScore. Aggiunto anche il relativo get.
<b>GetScore</b>	Metodo	Imposta la variabile gameCoverage con la copertura ricevuta in ingresso.
<b>Interfaces/BaseService.java</b>		
<b>CallRestPut</b>	Metodo	Aggiunte per un put di un jsonObject.
<b>Interfaces/T4Service.java</b>		
<b>registerAction("CreateGame")</b>	Registrazione azione REST	Aggiunta del parametro scalataId di tipo Optional<Integer> per la creazione di un gioco, modificando la chiamata REST per includerlo.

<b>registerAction("CreateRound")</b>	Registrazione azione REST	Aggiunto il parametro robot_id di tipo int per la creazione di un round, per includere l'ID del robot.
<b>registerAction("EndTurn")</b>	Registrazione azione REST	Modifica dei parametri (ordine cambiato da (String, String, int) a (int, String, String)) e aggiunta la gestione del robot ID.
<b>registerAction("GetRobotID")</b>	Registrazione azione REST	Aggiunta di una nuova azione REST per recuperare l'ID del robot.
<b>CreateTurn</b>	Metodo privato REST (chiamata POST)	Aggiunta gestione del response JSON.
<b>EndTurn</b>	Metodo privato REST (chiamata PUT)	Conversione del payload da MultiValueMap<String, String> a JSONObject per inviare dati in formato JSON. L'ID del turno (turnId) ora è una String invece di int.
<b>registerAction("UpdateScalata")</b>	Registrazione azione REST	Aggiunta la registrazione per l'aggiornamento della scalata.
<b>registerAction("CloseScalata")</b>	Registrazione azione REST	Aggiunta la registrazione per la chiusura di una scalata.
<b>CreateGame</b>	Metodo privato REST (chiamata POST)	Aggiunta della logica per inviare i dati di creazione del gioco, inclusa la gestione del parametro scalataId opzionale.
<b>EndGame</b>	Metodo privato REST (chiamata POST)	Metodo rimasto invariato ma invece di utilizzare MultiValueMap<String, String> formData abbiamo utilizzato un JSONObject obj.
<b>CreateRound</b>	Metodo privato REST (chiamata POST)	Aggiunta del parametro robot_id per la creazione di un round.
<b>EndTurn</b>	Metodo privato REST (chiamata PUT)	Modificato per inviare i dati di fine turno come JSON tramite una chiamata PUT.
<b>GetRobotID</b>	Metodo privato REST (chiamata GET)	Nuovo metodo per ottenere l'ID del robot, usando i parametri classUT, robot_type e difficulty in una chiamata GET a /robots.

<b>CreateScalata</b>	Metodo privato REST (chiamata POST)	Nuovo metodo per creare una scalata, che invia i dati come JSON e restituisce l'ID della scalata creata.
<b>UpdateScalata</b>	Metodo privato REST (chiamata PUT)	Aggiunta la logica per aggiornare una scalata, modificando il round corrente e la data di aggiornamento.
<b>CloseScalata</b>	Metodo privato REST (chiamata PUT)	Aggiunta la logica per chiudere una scalata, inviando il risultato finale, il punteggio e la data di chiusura tramite una chiamata PUT.

## T5 Frontend:

Nome	Tipo	Modifiche
<b>js/Button_Editor.js</b>		
<b>DOMContentLoaded</b>	Event Listener	Aggiunto un controllo if per verificare se modalita è "Scalata". Se modalita è "Scalata", aggiunge round_corrente_info nel messaggio.
<b>flush_localStorage</b>	Funzione	Se modalita è "Scalata", vengono eliminati anche i dati specifici della scalata (scalataId, SelectedScalata, gameId, ecc.).
<b>js/REST_Editor.js</b>		
<b>GetGameData</b>	Funzione	Introdotta controllo per modalita == "Scalata", Aggiunto is_scalata_inprogress per verificare se la scalata è in corso, Se la scalata non è iniziata, chiama handleScalataParameters(), ggunti nuovi parametri nel return (scalata_name, scalata_classes, ecc.). Riposizionato controllo sul parametro del link.
<b>\$(document).ready</b>	Event Listener	Aggiunto if(is_scalata_inprogress == false) prima di chiamare startGame(data). Aggiunto try-catch, dove in caso di eccezione viene mostrato un alert che porta al menu principale.
<b>handleResponse</b>	Funzione	Aggiunto totalScore tra i parametri estratti da response.
<b>processCoverage</b>	Funzione (async)	Aggiunto totalScore tra i parametri della funzione.
<b>handleEndGame</b>	Funzione	Modifiche radicali per migliorare la gestione della modalità "Scalata". Aggiunta condizione di vittoria di una partita scalata, e degli alert a seconda della vittoria, sconfitta, passaggio al round successivo. Ad ogni alert è seguito l'inserimento di un timeout di 10 secondi che reindirizza in una pagina di default.
<b>js/Util_Editor.js</b>		
<b>startGame</b>	Funzione async	Se modalita === "Scalata", recupera roundID, gameId, turnID, scalataID dalla risposta JSON e li salva in localStorage.
<b>getScalataClasse</b>	Funzione	Estrae la classe associata al round corrente dalla JSON della scalata.
<b>getScalataRobot</b>	Funzione	Estrae il robot associato al round corrente dalla JSON della scalata.
<b>getScalataDifficulty</b>	Funzione	Estrae e converte la difficoltà del round corrente.
<b>GetDifficulty</b>	Funzione	

		Mappa la difficoltà da Beginner, Intermediate, Advanced a valori numerici 1, 2, 3.
<b>HandleScalataParameters</b>	Funzione	Recupera e stampa i dati relativi alla scalata da localStorage.
<b>HandleScalataNextRound</b>	Funzione	Controlla se la scalata è terminata, Se la scalata è ancora in corso, aggiorna localStorage con i parametri del prossimo round, e Incrementa current_round_scalata e aggiorna il punteggio della scalata per la visualizzazione nel Frontend.
<b>js/Var_Editor.js</b>		
<b>var editor</b>	Variabili globali	Aggiunte le variabili per gestire la modalità Scalata (ID, nome, round corrente, totale round, robot, difficoltà, classi, punteggio totale, stato della scalata).
<b>js/gamemode_scalata.js</b>		
<b>gamedatawriter</b>	AJAX e gestione scalata	Aggiunta chiamata AJAX per ottenere la lista delle scalate e pulsante "Info" per identificare se la scalata è predefinita o personalizzabile.
<b>PressedSubmit</b>	Funzione	Rimossa la chiamata AJAX per salvare la scalata e aggiunto il reindirizzamento all'editor basato su ClassUT in localStorage.
<b>RetrieveScalata</b>	Funzione	Aggiunta per recuperare i dati di una scalata, salvandoli in localStorage (classi, robot, difficoltà, numero di round, ecc.).

**T4:**

Nome	Tipo	Modifiche
<b>round/round.go</b>		
<b>Round</b>	Struct	Aggiunti RobotID e Robot alla struttura.
<b>CreateRequest</b>	Struct	Aggiunta proprietà RobotID con validazione per obbligatorietà.
<b>CreateRequest</b>	Funzione	Controllo sul id del robot.
<b>UpdateRequest</b>	Struct	Aggiunto id del robot.
<b>FromModel</b>	Funzione	Modificata per includere RobotID dalla struttura del modello.
<b>round/Service.go</b>		
<b>Create</b>	Funzione	Ora assegna un solo robot al round, aggiungendo il campo RobotID e utilizzando l'ID del robot nel momento della creazione del round.
<b>Update</b>	Funzione	Aggiunta la modifica del metodo, passando da Updates(req) a Updates(r) e modificato il trattamento dell'errore.
<b>FindById</b>	Funzione	Aggiunta la pre-caricamento del campo Robot.
<b>FindByGame</b>	Funzione	Aggiunta la pre-caricamento del campo Robot.

## T1 Frontend:

Nome	Tipo	Modifiche
<b>js/Scalata.js</b>		
<b>handleRoundsChange, handleConfirm</b>	Funzioni	Modifica effettuata per impedire l'inserimento del numero del round da tastiera. Inoltre tutti i controlli sul nome della scalata e sul numero di round sono stati spostati sul tasto aggiungi in handleRoundsChange(), di conseguenza sono stati tolti da handleConfirm() che posizionava tali controlli solo alla fine della creazione della scalata (click tasto conferma).
<b>SelectClasses</b>	Funzione	Aggiunti gestori di eventi per i menu a tendina delle card con i relativi controlli.
<b>DisplayClasses</b>	Funzione	Modifiche effettuate per mostrare le coperture da battere, robot e difficoltà.
<b>submitScalataData</b>	Funzione	Aggiunta la riabilitazione del tasto “aggiungi” e dei componenti della parte superiore della pagina come il nome, descrizione e numero di round.
<b>ready</b>	Funzione	Aggiunto il codice per modificare il cambiamento di modalità scalata, nel caso di scalata predefinita setta il robot selezionato dall'amministratore.
<b>updateDifficultySelect</b>	Funzione	Aggiunta per cambiare le difficoltà in base al robot selezionato, perché nel caso un robot abbia una difficoltà con coverage nulla non viene proprio mostrata all'amministratore.
<b>\$(document).on('change', 'robot-select')</b>	Event Listener	Quando cambia il robot selezionato viene chiamata updateDifficultySelect.
<b>js/class.js</b>		
<b>EliminaScalata</b>	Funzione	Nuova funzione per eliminare una scalata tramite la chiamata DELETE all' endpoint /delete_scalata/{scalataName}.
<b>EliminaClasse</b>	Funzione	Aggiunto controllo sullo <b>status 409</b> (classe usata in una scalata). Analizza il testo della risposta per estrarre i nomi delle scalate associate. Utilizza <b>SweetAlert (swal)</b> per mostrare un messaggio di conferma all'utente. Se l'utente conferma, elimina prima le scalate associate e poi la classe.
<b>js/scalata_tour.js</b>		



#modeRobotSelect	Elemento UI	Aggiunto nuovo step nel tour per spiegare la selezione tra scalata personalizzabile e predefinita.
localStorage.getItem('tourShown')	Controllo	Ora il tour viene mostrato solo la prima volta grazie all'uso di localStorage.
#summaryButton	ID	Corretto da #summarymButton a #summaryButton.

T1 Backend:

Nome	Tipo	Modifiche
controller/HomeController.java		
EliminaClasse	Metodo	Aggiunto il controllo per verificare se la classe è presente in una scalata prima di eliminarla.
service/AdminService.java		
UploadTest	Metodo	Ora passa l'oggetto ClassUT a RobotUtil.saveRobots invece del solo nome della classe. Aggiunta lista robotList con i robot disponibili (Randoop, Evosuite) e associata all'oggetto ClassUT. Aggiunta lista robotDifficultyList con i livelli di difficoltà (Beginner, Intermediate, Advanced) e associata all'oggetto ClassUT.
modificaClasse	Metodo	Aggiunti i campi robot, robotDifficulty e coverage all'Update della classe. Ora aggiorna anche le liste di robot e difficoltà oltre alle informazioni di base.
service/ScalataService.java		
FindScalataByClassName	Metodo	Aggiunto metodo per ritornare la lista delle scalate che contengono una data classe.
repository/ScalataRepository.java		
FindBySelectedClassesClassName	Metodo	Aggiunto nuovo metodo per trovare tutte le scalate che contengono una classe specifica.
model/ClassUT.java		
RobotList	Variabile	Aggiunta lista per memorizzare i tipi di robot associati alla classe.
RobotDifficulty	Variabile	Aggiunta lista per memorizzare i livelli di difficoltà dei robot.

<b>Coverage</b>	Variabile	Aggiunta lista per memorizzare la copertura dei test della classe.
<b>ClassUT</b>	Costruttore	Modificato per accettare i nuovi parametri robotList, robotDifficulty, e coverage.
<b>getRobotList, setRobotList</b>	Getter/ Setter	Permette di ottenere e impostare la lista dei robot associati alla classe.
<b>getRobotDifficulty, setRobotDifficulty</b>	Getter/ Setter	Permette di ottenere e impostare la lista dei livelli di difficoltà dei robot.
<b>getCoverage, setCoverage</b>	Getter/ Setter	Permette di ottenere e impostare la lista delle coperture di test della classe.
<b>ToString</b>	Metodo	Modificato per includere robotList, robotDifficulty e coverage nella stringa di output.
<b>filesystem/RobotUtil.java</b>		
<b>SaveRobots</b>	Metodo	Ora riceve un oggetto ClassUT invece del solo className, permettendo di aggiornare direttamente i dati della classe. Aggiunta gestione delle coperture (coverageList[]) per memorizzare i risultati dei test in MongoDB. Ora imposta le coperture dei test (setCoverage(...)) direttamente su classUT.