Università degli Studi di Milano

FACOLTÀ DI SCIENZE E TECNOLOGIE
Corso di Laurea Magistrale in Informatica

Un applicativo per la gestione di gare a distanza: analisi, revisione e implementazione di nuove funzionalità

Relatore:
Violetta Lonati

Candidato:
Riccardo Macoratti
Matricola 884553

Correlatore:
Mattia Monga

Anno Accademico 2016–2017

Indice

1 Introduzione **1**

2 Il Bebras dell'Informatica in Italia **4**
2.1 Caratteristiche della gara 4
2.2 Il sistema di gara 8
2.3 Soggetti coinvolti 11
2.4 I quesiti Bebras 14

3 Vincoli progettuali imposti dal sistema esistente **17**
3.1 Contesto e criticità 17
3.2 Modalità di sviluppo del sistema di gara 19
3.3 Realizzazione di nuove funzionalità 21
3.4 Esportazione dei quesiti 23

4 Analisi del sistema di gara **27**
4.1 Architettura del sistema 28
4.2 Tecnologie impiegate 32
4.3 Dettagli implementativi 36
4.3.1 Quesiti e gruppi di quesiti 40
4.3.2 Scuole, docenti e squadre 47
4.3.3 Gare 49

5 Revisione e implementazione **59**
5.1 Il modulo *helpdesk* 60
5.1.1 Stato dell'arte e possibili sviluppi 60
5.1.2 Progetto e funzionalità 64
5.1.3 Implementazione e dettagli 68
5.2 Utilizzo dei quesiti anche al di fuori delle gare 75
5.2.1 Architettura 76
5.2.2 Dettagli tecnici 82

6 Conclusione e sviluppi futuri **90**

Bibliografia **92**

Elenco delle figure

2.1 Interfaccia grafica del sito di monitoraggio che viene utilizzato dallo *staff* Bebras. 8
2.2 Esempio del quesito, come viene mostrato alla squadra durante la gara. . . 14
2.3 Esempio della spiegazione della risposta e della sezione “Anche questa è informatica” che viene mostrata alla squadra in fase di revisione della gara. 16

3.1 I due tipi di decomposizione di un sistema in moduli illustrati da Parnas in [1]. 22
3.2 Il *formato standard* per l’esportazione dei quesiti utilizzato come interfaccia per i nuovi strumenti. 26

4.1 Schema generale dell’architettura del sistema e delle interazioni tra le sue componenti. 28
4.2 Schema generale delle tecnologie impiegate in ogni componente del sistema. 32
4.3 Schema delle tabelle e delle relazioni contenute nella base di dati. 37
4.4 Diagramma delle risorse coinvolte nella modellazione di un quesito standard. 41
4.5 Diagramma delle risorse coinvolte nella modellazione di un quesito non standard. 45
4.6 Diagramma dello scambio di dati tra la parte *server* e la parte *client* del modulo *students*. 51

5.1 Come si presenta l’interfaccia grafica del sito *helpdesk*. 65
5.2 Schema dell’evoluzione dell’architettura con l’aggiunta del modulo *helpdesk*. 69
5.3 Schema del flusso di dati che viene implementato dal modulo *helpdesk*. . 72
5.4 Architettura delle API `export` e della libreria `bebraslib`. 77
5.5 L’interfaccia di ricerca dei quesiti fornita da Bebras Question Explorer. . . 81
5.6 Come `bebraslib` mostra la valutazione del quesito svolto. 87

Elenco dei listati

5.1 Strutturazione degli URL delle risorse. 83
5.2 Esempio di risultato fornito da `list`. 84
5.3 Esempio di risultato fornito da `questions`. 85

Capitolo 1

Introduzione

Nel mondo di oggi l'informatica sta assumendo un ruolo sempre più importante, perché la maggioranza delle azioni che un individuo compie ogni giorno vengono silenziosamente supportate da qualche tipo di sistema informatico. In conseguenza della spinta proveniente da tutti gli strati della società, si sta assistendo all'ingresso dell'informatica nella scuola italiana, intesa però principalmente come uso di dispositivi e applicativi. Una parte degli esperti del settore preme invece affinché l'informatica venga considerata e insegnata come una disciplina scientifica propria, con un valore formativo al pari delle altre.

Tra le molte iniziative nate per diffondere quest'idea, c'è il "Bebras dell'Informatica", una gara non competitiva di informatica rivolta a bambini e ragazzi. Il Bebras intende stimolare l'interesse verso l'informatica come disciplina scientifica e presentare concetti complessi per mezzo del divertimento e della sfida, senza però richiedere una conoscenza specifica pregressa della materia, proponendo una serie di quesiti di vario genere, ognuno dei quali basato su un problema concreto di natura informatica.

Il Bebras è nato in Lituania e si è presto diffuso a livello internazionale, dando vita ad una comunità di paesi che ne cura le edizioni locali. L'edizione italiana della competizione è organizzata dal Laboratorio ALaDDIn del Dipartimento di Informatica dell'Università degli Studi di Milano e vede l'uso di diverse applicazioni *software*, sviluppate appositamente, per supportare la preparazione, lo svolgimento e il monitoraggio della competizione. Il sistema di gara è risultato particolarmente efficace nella gestione delle gare nelle scuole italiane, che notoriamente vertono in pessime condizioni per quanto riguarda la strumentazione informatica in dotazione.

Con il passare degli anni, il succedersi delle gare e il crescere del numero dei partecipanti coinvolti, il sistema di gara ha immagazzinato un elevato numero di informazioni interessanti: a partire dai quesiti inseriti per ogni edizione annuale della competizione, fino ai dati raccolti durante lo svolgimento delle gare. Sarebbe interessante poter estrarre questi dati e riutilizzarli anche al di fuori del contesto delle gare. I quesiti Bebras, ad esempio, risultano molto utili in ambito scolastico, poiché hanno un notevole valore formativo che attualmente non viene sfruttato. Inoltre l'analisi statistica dei dati relativi alle risposte date dai partecipanti potrebbe fornire elementi utili a migliorare la qualità dei quesiti per le competizioni successive, oppure a esaminare le difficoltà incontrate dagli studenti e il modo in cui le hanno affrontate.

Purtroppo il *software* di gara non è mai stato progettato in maniera organica, ma si è evoluto in risposta alle esigenze del momento, quasi sempre in tempi ridotti e a ridosso dei periodi di gara. Prima di questa tesi, intervenire sul sistema era poco praticabile: il *software* è stato realizzato ed è stato sempre gestito da un singolo sviluppatore esterno al gruppo di ALaDDIn. Non è mai stato impiegato nessun programma di controllo della versione, non esisteva documentazione implicita o esplicita e le funzionalità erano note allo *staff* di ALaDDIn solo dal punto di vista dell'utente. Per queste ragioni, benché il sistema si fosse dimostrato negli anni robusto e adatto allo scopo originario per cui era nato, non era scontata la possibilità di poterlo adattare a nuove funzionalità.

Il primo obiettivo della tesi è stato dunque quello di valutare se e come fosse possibile estendere il *software* preservandone il medesimo utilizzo durante lo svolgimento delle gare e senza stravolgerlo, in modo da non perdere la preziosa collaborazione del suo autore e unico gestore. A tal fine, si è in primo luogo condotta l'analisi del *software* di gara a diversi livelli di profondità, producendo una documentazione della sua architettura, delle tecnologie impiegate e dei dettagli tecnici che lo caratterizzano. Si è quindi valutata la qualità dell'applicazione, mettendo in risalto i vincoli architetturali dati dalla parte di sistema già in uso, la validità delle scelte implementative attuate durante gli anni di sviluppo e le criticità poste dal contesto di utilizzo. Una volta prodotta una chiara immagine del sistema in tutte le sue sfaccettature, è stato possibile progettarne miglioramenti e aggiunte.

In particolare, con l'obiettivo di facilitare l'attività di assistenza allo svolgimento delle gare, è stata creata la nuova componente *helpdesk* revisionando, consolidando e rendendo più efficienti alcune funzionalità già esistenti, in modo da poter monitorare e gestire efficacemente un grande numero di gare simultanee.

Inoltre, ci si è occupato di come favorire l'utilizzo dei quesiti Bebras anche al di fuori delle gare. A differenza di quanto detto per il supporto allo svolgimento delle gare, in questo caso il sistema non prevedeva nessun meccanismo di esportazione e anzi presentava una struttura che rendeva difficile la realizzazione di un meccanismo di questo tipo. Abbiamo affrontato il problema tramite l'individuazione e la messa a punto di un interfaccia e di un formato formato *standard* per i quesiti, assieme alla progettazione di nuove componenti *software* basate su questo formato.

La tesi è organizzata come segue: nel capitolo 2 "Il Bebras dell'Informatica in Italia", si presenta il"Bebras dell'Informatica", la comunità internazionale che lo organizza e i suoi obiettivi, soffermandosi in particolare sulle caratteristiche della gara italiana e sulla struttura dei quesiti che la compongono. Nel capitolo 3 "Vincoli progettuali imposti dal sistema esistente", si illustrano le criticità che il sistema di gara deve affrontare e si descrivono le modalità con cui è avvenuto il suo sviluppo. Viene valutato il sistema dal punto di vista della manutenibilità e si propone un possibile metodo operativo per inserire nuove funzionalità, prendendo ad esempio l'utilizzo dei quesiti al di fuori di una gara. L'analisi tecnica del sistema di gara è riportata al capitolo 4 "Analisi del sistema di gara", in cui si descrivono la sua architettura e le tecnologie che sono state impiegate nella sua implementazione e ci si sofferma su alcuni dettagli tecnici relativi ai principali tipi di dato trattati dal sistema. Il capitolo 5 "Revisione e implementazione" è dedicato al lavoro di revisione del sistema e di implementazione di nuove componenti. La sezione 5.1 "Il modulo *helpdesk*" è dedicata al nuovo sito di supporto, di cui si descrive l'architettura e si dettagliano le tecnologie impiegate, le scelte implementative e le differenze con le funzionalità preesistenti. Nella sezione 5.2 "Utilizzo dei quesiti anche al di fuori delle gare" vengono descritte le principali caratteristiche degli strumenti sviluppati *ex novo* per consentire l'esportazione dei quesiti Bebras in un formato *standard* e viene mostrato un esempio di applicazione che li utilizza. Nell'ultimo capitolo si riepilogano il lavoro svolto e i risultati raggiunti, presentando inoltre alcuni possibili sviluppi futuri.

Capitolo 2

Il Bebras dell'Informatica in Italia

In questo capitolo, vengono presentate le gare "Bebras dell'Informatica" (d'ora in poi *Bebras*) in Italia.

Nella sezione 2.1 "Caratteristiche della gara", si fa una descrizione dei diversi momenti di una competizione dettagliando le varie fasi della gara, si specificano quali sono gli strumenti a supporto del suo svolgimento e si definiscono i diversi soggetti coinvolti. Il sistema di gara è illustrato nella sezione 2.2, facendo riferimento soprattutto alla sua evoluzione nel tempo ed evidenziando le criticità che è in grado di gestire. La sezione 2.3 "Soggetti coinvolti" tratta le tre tipologie di attori che partecipano al Bebras, mettendo in risalto il loro ruolo nelle varie fasi della competizione ed elencando le modalità di interazione col sistema di gara. Le gare sono composte da una serie di quesiti, le cui caratteristiche sono discusse nella sezione 2.4.

2.1 Caratteristiche della gara

Il Bebras[1] è un'iniziativa internazionale nata in Lituania che ha lo scopo di far conoscere e promuovere gli aspetti scientifici dell'informatica e il pensiero computazionale nelle scuole, agli studenti di tutte le età, attraverso dei giochi informatici organizzati contemporaneamente in molti paesi del mondo. L'iniziativa Bebras coinvolge a livello internazionale una rete di organizzazioni, definita *comunità* Bebras, che si occupa di realizzare le gare su scala nazionale. La caratteristica fondamentale dei giochi Bebras è quella di non richiedere alcuna conoscenza pregressa o alcun tipo di

[1]Maggiori informazioni sull'iniziativa Bebras internazionale si possono trovare sul sito ufficiale in [2].

addestramento ai fini della partecipazione, pur riuscendo a proporre dei quesiti ispirati a situazioni reali, con problemi di natura informatica. I quesiti presentati si propongono di sviluppare le competenze informatiche di base, legate, ad esempio, alla rappresentazione delle informazioni, alla manipolazione di strutture discrete, all'elaborazione di dati e al ragionamento algoritmico. Considerando i dati aggiornati all'anno 2016, alla comunità Bebras hanno aderito più di 60 nazioni, tra cui l'Italia.

A livello internazionale, l'iniziativa Bebras è incentrata sullo svolgimento di due eventi annuali: il primo è una sessione di pensiero collettivo in cui esperti del settore provenienti da ogni nazione si ritrovano per decidere gli argomenti e creare i quesiti che saranno proposti all'edizione dei giochi dell'anno in corso; questo evento è rivolto esclusivamente agli organizzatori dei giochi; il secondo è la settimana riservata alle gare annuali ed è, al contrario, dedicato principalmente agli studenti. Ogni nazione partecipante organizza una competizione locale, nella quale viene data agli studenti la possibilità di confrontarsi con i quesiti precedentemente elaborati. La settimana di gara è fissata e condivisa tra tutti i paesi partecipanti, ma le gare vengono svolte in autonomia da ogni nazione. Ogni paese partecipante sceglie la propria serie di quesiti all'interno dell'insieme concordato collettivamente.

A partire dal 2015, esiste una declinazione italiana del Bebras[2] che è gestita dal Laboratorio ALaDDIn[3]. Precedentemente all'impegno con il Bebras italiano, ALaDDIn era l'organizzatore dei Kangourou dell'Informatica[4], un'iniziativa simile, in quanto anch'essa tesa ad organizzare dei giochi informatici, anche se con una strutturazione della competizione leggermente differente.

In Italia, i Bebras dell'Informatica prevedono tre utenti coinvolti nello svolgimento:

- squadre di *studenti*;
- *docenti responsabili* delle squadre;
- *staff* Bebras, diviso in *organizzatori* e *personale di supporto*.

[2] È possibile consultare il sito ufficiale del Bebras italiano in [3].

[3] ALaDDIn è il Laboratorio di Didattica e Divulgazione dell'Informatica del Dipartimento di Informatica dell'Università degli Studi di Milano; è possibile consultare il loro sito Web in [4] per maggiori informazioni

[4] Il sito Web delle competizioni dei Kangourou dell'Informatica è disponibile all'indirizzo `http://kangourou.di.unimi.it/`.

Una descrizione approfondita di questi attori e del loro ruolo nel Bebras si trova alla sezione 2.3.

Le gare Bebras vengono svolte a distanza, via Web, all'interno degli edifici scolastici dove gli studenti normalmente frequentano le lezioni. Contrariamente a quanto succede per altri paesi del mondo, non sono affrontate da studenti singoli, bensì viene proposta una competizione a squadre. Le squadre sono composte da gruppi di quattro studenti che svolgono la competizione insieme, lavorando davanti allo stesso monitor. La partecipazione alla competizione è gratuita ed è possibile iscrivere le squadre via Web. Per ogni scuola partecipante deve esserci un insegnante responsabile che si occupa di gestire fisicamente le squadre provenienti da quell'istituto e sovrintendere allo svolgimento della gara. L'organizzazione della competizione è curata dello *staff* del Bebras italiano, diviso in due gruppi operativi. Un primo gruppo si occupa di preparare i quesiti che compongono le gare, un secondo di fornire assistenza durante la settimana di gara. La competizione si articola, quindi, in diverse fasi consecutive:

- lo *staff* partecipa all'evento internazionale in cui tutti i paesi concordano lo stesso insieme di quesiti da utilizzare durante l'edizione della gara relativa all'anno in corso; come si può immaginare, i quesiti vengono ideati e scambiati in lingua inglese;

- i quesiti, che attualmente seguono il modello illustrato alla sezione 2.4, devono essere tradotti in italiano e convertiti nel formato che il sistema di gara è in grado di capire; la traduzione ha un'importanza particolare, poiché, vista la natura dei quesiti, è più un'operazione di interpretazione allo scopo di rendere il senso generale della domanda;

- la gara viene quindi preparata, mentre i docenti vi si iscrivono via Web, specificando le squadre di cui sono responsabili; per ogni squadra creata, vengono generati nome utente e *password* da utilizzare per accedere alla competizione nella settimana di gara;

- si svolgono le gare, durante la settimana di gara; di solito il tempo per ogni squadra è di 45 minuti ed i quesiti possono essere svolti nell'ordine desiderato; in questo frangente l'insegnante responsabile vigila sul rispetto del regolamento di gara;

- durante la competizione, la parte designata dello *staff* Bebras si occupa dell'attività di supporto allo svolgimento delle gare, fornendo assistenza tecnica ai docenti responsabili e alle squadre, garantendo la buona riuscita della competizione;
- vengono analizzate le gare, al fine di calcolare i risultati di ogni squadra e di stilare le classifiche, "Most Significant Bytes", delle scuole che hanno presentato le squadre che si sono posizionate meglio;
- lo *staff* Bebras abilita la visualizzazione dei risultati, cosicché i docenti possano avere la possibilità di discutere le soluzioni insieme alle squadre; gli insegnanti consegnano alle squadre i certificati di partecipazione.

Le gare Bebras in Italia sono state capaci di attrarre un grande numero di studenti che hanno affrontato le competizioni annuali. Secondo i dati disponibili[5], una base di più di 3000 studenti ha partecipato all'ultima edizione 2014-2015 dei Kangourou dell'Informatica, mentre, nell'ultima edizione 2017-2018 delle gare Bebras, è stato possibile superare la soglia delle 10 000 squadre, che si traducono in circa 40 000 studenti. Una partecipazione così ampia alle competizioni da concentrare per giunta in un'unica settimana di gara, rende necessario permettere che molte squadre possano competere contemporaneamente. Inoltre, anche se la competizione è organizzata in maniera centralizzata dallo *staff* Bebras, gli studenti che vi prendono parte devono essere in grado di svolgere la gara rimanendo all'interno dell'istituto scolastico di appartenenza, sotto la supervisione dei docenti responsabili.

Considerate le premesse appena fatte, allo scopo di portare a termine con successo una competizione senza un eccessivo dispendio di risorse, fin dal periodo Kangourou è disponibile sistema *software* in grado di gestire delle gare a distanza, costruito in maniera tale da adattarsi alla particolare situazione presente nelle scuole italiane, che rappresentano l'ambiente dove si trova ad operare. Il sistema sfrutta la rete Internet e permette a molte squadre di riuscire a svolgere contemporaneamente la propria gara, virtualmente senza l'intervento di alcun membro dello *staff* Bebras.

[5] Il dati citati sono pubblicati sul sito del Bebras italiano in [3], in questa pagina `https://bebras.it/comitato.html`

2.2 Il sistema di gara

In questa sezione, vengono illustrati gli strumenti *software* che fanno parte del sistema di gara e che sono le sue componenti principali, evidenziandone il loro processo evolutivo. Tali *software* sono nati in momenti molto differenti, ma attualmente operano all'unisono per affiancare i processi di iscrizione, monitoraggio e, più in generale, svolgimento dell'edizione nazionale dei Bebras. Formalmente, il "sistema di gara" si può definire come l'insieme delle componenti di cui è composto unitamente alle interazioni tra di essi.

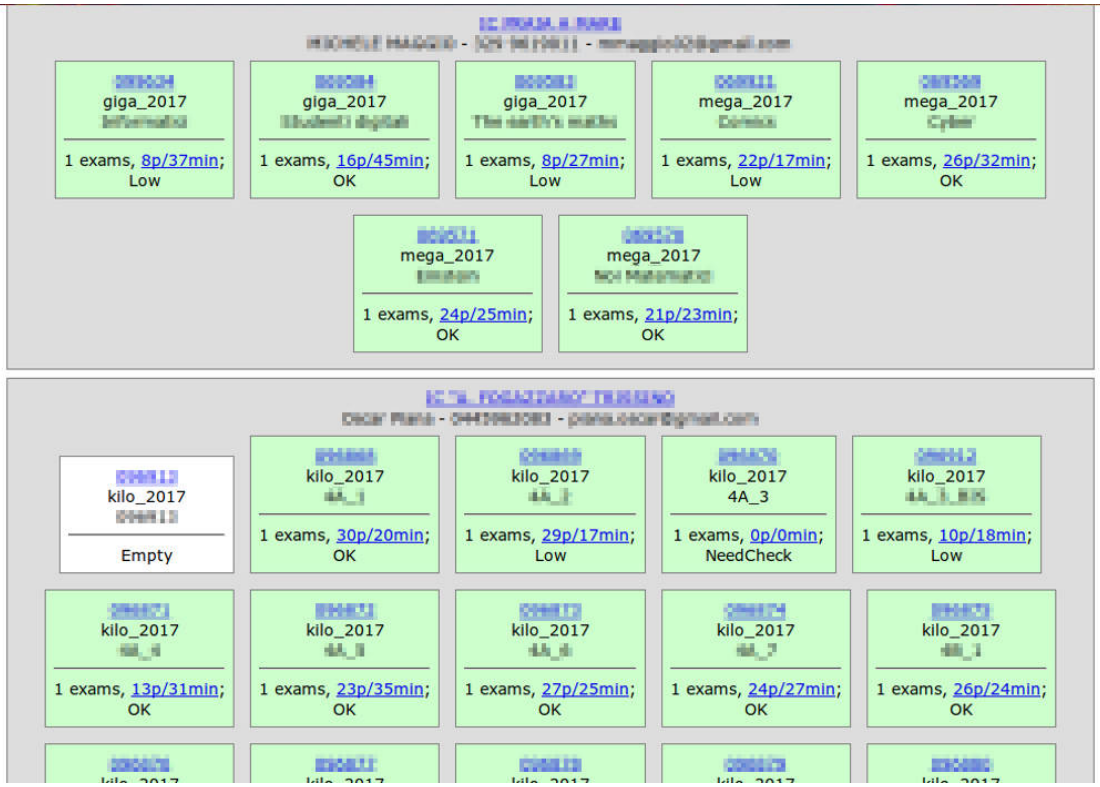

Figura 2.1: Interfaccia grafica del sito di monitoraggio che viene utilizzato dallo *staff* Bebras.

Se considerato nella versione disponibile al momento dell'analisi, dal punto di vista dell'utente finale, il sistema di gara si presenta sotto forma di applicazione Web. Le componenti a cui gli utenti possono accedere appaiono pertanto come dei siti Web[6]. Troviamo:

[6]I siti Web di cui si parla sono attualmente operativi e visionabili in [3].

- la *piattaforma di gara*[7], la componente che mostra l'interfaccia di gara; rende disponibile sia il *form* per l'accesso all'edizione corrente della gara tramite nome utente e *password*, sia un secondo accesso privo di credenziali, utile per l'allenamento con i quesiti delle edizioni passate; una volta entrati, vengono mostrati i vari *test* a disposizione e, scelto quello desiderato, vengono mostrati i quesiti relativi a quel *test*;

- il *sito docenti*[8], la componente che fornisce un'interfaccia di gestione delle squadre; da questo sito si possono effettuare una varietà di operazioni, a partire dalla creazione delle squadre, fino ad arrivare alla definizione dei membri di ogni squadra e alla stampa dei certificati di partecipazione; si può considerare come la componente tecnicamente meno complessa di tutto il sistema, poiché svolge un compito limitato e ben definito;

- il *sito di monitoraggio*[9], la componente che si occupa di fornire allo *staff* Bebras l'interfaccia che serve a gestire l'operazione di supporto allo svolgimento delle gare, visibile in figura 2.1; questo sito è riservato e accessibile, pertanto, solo dallo *staff* Bebras; tra le funzionalità è presente quella di calcolare e presentare alcune statistiche riguardanti i dati delle gare.

Esiste anche una base di dati che fornisce la memorizzazione a lungo termine delle informazioni nel sistema. Anche se non è visibile dall'utente finale, se ne fa cenno perché utile per descrivere l'evoluzione dello sviluppo del sistema. L'analisi tecnica delle tre componenti, e della base di dati, sarà effettuata al capitolo 4 "Analisi del sistema di gara".

Dei tre siti disponibili, ognuno è adibito ad una funzione ben precisa ed è evidente come ognuno sia legato ad uno specifico tipo di utente del sistema, tra quelli descritti alla sezione 2.3 "Soggetti coinvolti". Il fatto che esistano tante componenti quanti sono gli attori del sistema non è, però, frutto di una progettazione mirata, ma è una situazione che si è venuta a creare dopo che si è dovuto, più volte, aggiungere delle funzionalità per coprire le esigenze emerse in un determinato momento della vita del *software*. L'avanzamento tecnologico del sistema di gara come risposta a qualche tipo di bisogno sarà costante di tutto lo sviluppo del sistema.

[7] La piattaforma di gara è visibile all'indirizzo `https://bebras.it/students`.

[8] Il sito docenti è visibile all'indirizzo `https://bebras.it/teachers`, previa registrazione di un *account*.

[9] Il sito di monitoraggio è operativo all'indirizzo `https://bebras.it/backend`, ma è utilizzabile solo dallo *staff* Bebras durante il supporto alle gare.

Lo sviluppo dei tre siti è avvenuto, quindi, in tempi sostanzialmente diversi tra loro, nella maggioranza dei casi a ridosso dei periodi di gara e, soprattutto, aggiungendo uno "strato" di modifiche ad ogni revisione. Queste modalità caratterizzano così tanto il risultato finale, che la loro origine merita una descrizione più dettagliata.

Il *software* odierno affonda le radici in un sistema completamente diverso da quello che si vede oggi. Esso ha, infatti, subito due trasformazioni: è nato inizialmente per lo svolgimento delle sessioni d'esame ECDL[10], successivamente si è evoluto per supportare le gare dei Kangourou dell'Informatica e, infine, è diventato quello che si usa oggi per le gare Bebras.

All'inizio era destinato allo svolgimento degli esami ECDL. Per questo motivo non si parlava ancora di gare, ma di esami, ed esisteva solamente la piattaforma di gara, che, in quel caso, prendeva il nome di *piattaforma d'esame*. Ovviamente non erano presenti squadre, ma singoli studenti candidati all'esame. Non erano necessari ulteriori componenti *software*, poiché tutte le informazioni necessarie venivano inserite manualmente direttamente nel *database* e gli esami si svolgevano esclusivamente presso un centro d'esame con macchine appositamente configurate. Il funzionamento dell'esame era sostanzialmente differente da quello di una gara Kangourou o Bebras: l'intero sistema operativo veniva bloccato dall'interfaccia d'esame e per fare questo si impiegava una tecnologia diversa da quella che si usa oggi.

Successivamente è avvenuto il primo passaggio ai Kangourou dell'Informatica. Si è cominciato a parlare di gare e il funzionamento appena descritto, di bloccare completamente l'accesso al sistema operativo, si è evoluto verso una soluzione più permissiva. La piattaforma d'esame è stata ribattezzata in *piattaforma di gara*. Sebbene la competizione fosse già a squadre, ancora non era presente la loro gestione attraverso il sito docenti, in quanto le informazioni continuavano ad essere inserite manualmente. La tecnologia impiegata lato utente era però ancora la stessa del sistema precedente. Il problema con questa tecnologia era che si trattava di una soluzione proprietaria, non facilmente portabile a sistemi operativi diversi da Windows, richiedendo l'installazione di un *software* apposito nel calcolatore dello studente o della scuola: questo escludeva automaticamente i sistemi operativi basati su Linux e i dispositivi mobili. In questo momento è quindi avvenuta la riscrittura del *software* che presenta l'interfaccia di

[10]La patente europea per l'uso del computer, in inglese chiama, appunto, ECDL, è un attestato che certifica il possesso di una competenza informatica di base; per ottenerlo bisogna sostenere degli esami pratici al calcolatore.

gara all'utente utilizzando una nuova tecnologia capace di risolvere la maggioranza dei problemi appena descritti, senza stravolgere eccessivamente il funzionamento della gara. Nonostante la nuova forma, l'unico sito disponibile rimaneva la piattaforma di gara.

L'ultima trasformazione segna il passaggio dai Kangourou dell'Informatica alle gare Bebras. Il funzionamento della gara rimane principalmente lo stesso, ma, per semplificare l'accesso alla gara, si è deciso di spostare dallo *staff* Bebras agli insegnanti referenti l'onere dell'inserimento delle squadre nel sistema. Per consentire ciò, nasce il sito docenti, che li guida nella gestione delle squadre. Inoltre nel corso dello svolgimento della prima competizione Bebras, lo *staff* si è reso conto che non era sufficientemente pratico fornire supporto alle gare operando direttamente nel *database*. Con urgenza viene, quindi, sviluppato il sito di monitoraggio, la componente che fornisce un resoconto visuale dello stato delle gare in corso unito ad utili statistiche di gara. Contestualmente alla creazione di questo sito, nasce la strutturazione e la pianificazione delle attività di supporto allo svolgimento della competizione.

2.3 Soggetti coinvolti

Il sistema di gara, nella configurazione attuale, prevede tre attori. Si possono delineare:

- le squadre di studenti;
- i docenti responsabili delle squadre;
- lo *staff* Bebras.

A differenza di quanto avviene nella competizioni Bebras organizzate in altri paesi, in Italia il vero destinatario delle gare non è il singolo studente, ma la squadra. L'obiettivo principale delle gare Bebras si concentra nella fascia di età che va dagli 8 ai 18 anni, coincidente con l'età dell'istruzione primaria e secondaria, ma comunque rimanendo nelle classi pre-universitarie.

Al fine di poter comporre una gara che si adatti perfettamente, come difficoltà e come lunghezza, alla fascia d'età alla quale è destinata, la fascia 8-18 anni è stata divisa in cinque categorie, o *classi*, di partecipanti che sono state chiamate con i prefissi del Sistema internazionale di unità di misura, visto il loro ampio utilizzo in ambito informatico. Le categorie sono divise come segue:

KiloBebras
: studenti che frequentano gli ultimi due anni della scuola primaria, tra dagli 8 ai 10 anni circa.

MegaBebras
: studenti al primo e al secondo anno delle scuole secondarie di primo grado, dai 10 ai 12 anni circa.

GigaBebras
: studenti che frequentano le classi terze della scuola secondaria di primo grado, dai 12 ai 13 anni circa.

TeraBebras
: studenti al biennio della scuola secondaria di secondo grado, dai 13 ai 15 anni circa.

PetaBebras
: studenti al triennio della scuola secondaria di secondo grado, dai 16 ai 18 anni circa.

Come già precisato, le categorie appena presentate servono per differenziare le competizioni. Ciononostante, si possono trovare molti esempi in cui un quesito viene proposto sia ad una categoria, che a quella immediatamente successiva: in questo caso, cambiano le modalità di interazione del quesito, oppure variano i punteggi che vengono assegnati in fase di valutazione.

Procedendo nell'analisi, si trovano i docenti responsabili delle squadre. Gli insegnanti sono il punto di riferimento per le squadre che desiderano partecipare alla competizione e fungono da ponte tra lo *staff* Bebras e le squadre stesse. I loro compiti principali sono quello di intraprendere tutte le operazioni necessarie a rendere la scuola idonea alla partecipazione, iscrivendola attraverso il sistema di gara, e quello di introdurre i ragazzi alla competizione, spiegando le regole e chiarendo le incertezze che possono generarsi.

In qualità di responsabili, i docenti hanno dei compiti in ogni fase della competizione. Nel periodo prima della gara, formano le squadre per classi di età, curando il bilanciamento dei componenti; poi si preoccupano di registrare correttamente le squadre nel sistema di gara, specificandone la categoria di appartenenza e i nomi dei membri, che serviranno successivamente per la stampa dei certificati nominali di partecipazione. Durante lo svolgimento della competizione, devono supervisionare l'andamento della gara, garantendo soprattutto il rispetto del regolamento stabilito dallo *staff* Bebras. In

caso di problemi, possono contattare lo *staff* per il supporto tecnico. Al termine della competizione, devono stampare e consegnare gli attestati nominali di partecipazione ai componenti di ogni squadra. Una volta ottenute le soluzioni ai quesiti che lo *staff* Bebras mette a disposizione, gli insegnanti hanno la possibilità di discuterle assieme agli studenti, mostrandogli soprattutto la sezione "Anche questa è informatica".

Si conclude la descrizione con lo *staff* Bebras, l'ultimo tra gli attori attivi nel sistema di gara. All'interno dello *staff*, si possono distinguere due gruppi: uno che opera principalmente prima e dopo la competizione, mentre l'altro che svolge il proprio compito durante la competizione. Il primo gruppo si occupa dell'organizzazione della competizione, con attività quali la promozione della competizione nelle scuole, il rapporto con gli insegnanti, la creazione dei quesiti, e la loro vita all'interno del sistema di gara, e il risultato della competizione. Il secondo gruppo è principalmente impegnato nel fornire supporto tecnico agli insegnanti e alle squadre durante lo svolgimento della sessione di gara.

Com'è facile immaginare, anche nel caso dello *staff* Bebras, l'impegno è costante durante ogni fase della competizione. In primo luogo, lo *staff* si occupa di tradurre dall'inglese ed inserire nel sistema di gara i quesiti, selezionati tra quelli concordati durante il *workshop* annuale della comunità Bebras, com'è stato detto nella sezione 2.1 "Caratteristiche della gara". Per inserire i quesiti nel sistema di gara bisogna convertirli in un formato che si conformi a quello descritto più avanti alla sezione 4.3.1 "Quesiti e gruppi di quesiti". Ogni quesito presenta una difficoltà intrinseca, perciò bisogna decidere a quale categoria appartiene, tra quelle precedentemente descritte. Una volta creati i *test*, la gara è pronta e può subentrare il secondo gruppo. A questo punto, si è arrivati alla settimana di gara.

Questo gruppo si occupa, come già detto, di fornire supporto tecnico durante lo svolgimento delle gare. Si accerta che il maggior numero di gare vengano correttamente consegnate, monitorandole e contattando gli insegnanti nel caso sia necessario. Il loro compito è anche quello di rispondere a eventuali domande sull'uso della piattaforma di gara, segnalando allo sviluppatore del sistema eventuali problemi di natura tecnica. Terminata la settimana di gara, il controllo ritorna al primo gruppo.

Finita la competizione, lo *staff* stila le classifiche per decretare i vincitori della competizione. Successivamente vengono rese pubbliche le soluzioni dei quesiti con le relative spiegazioni. Per completare questa fase, lo *staff* Bebras fa un resoconto generale in cui si commenta l'andamento dell'organizzazione e si discute degli eventuali

miglioramenti da mettere in opera per l'edizione successiva della competizione. È spesso in questo momento che viene decisa la direzione in cui far evolvere il sistema di gara.

2.4 I quesiti Bebras

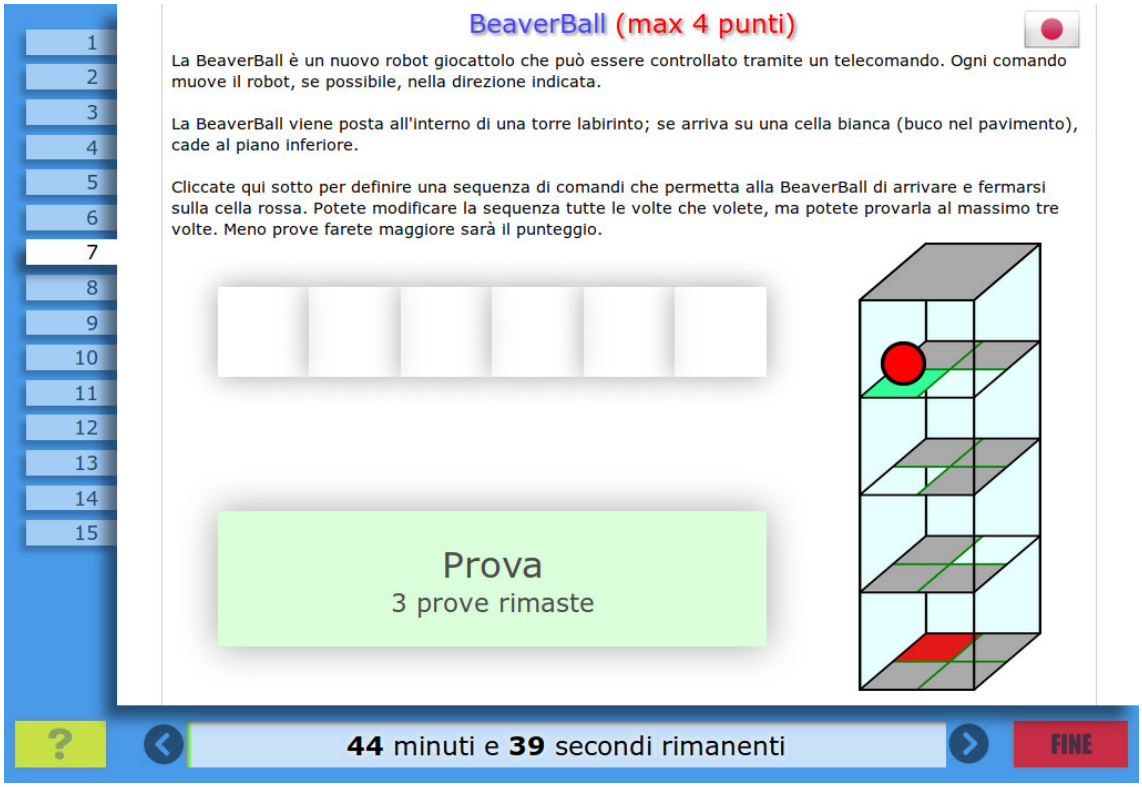

Figura 2.2: Esempio del quesito, come viene mostrato alla squadra durante la gara.

Nell'introduzione al capitolo, è stato fatto presente come venga attribuita una grande rilevanza ai quesiti Bebras. Ogni gara si compone di diversi *test*, ognuno destinato ad una specifica categoria di squadre. I *test* sono, a loro volta, formati da una serie di quesiti. I quesiti, o *task* in ambito internazionale, sono il principale mezzo con cui il Bebras cerca di far sviluppare capacità informatiche agli studenti, per questo motivo la descrizione di un quesito merita una trattazione a parte.

Un quesito nasce nel momento in cui un membro della comunità Bebras internazionale lo propone durante l'evento internazionale, chiamato "Bebras Workshop", che si svolge

verso il mese di maggio. Durante quest'occasione, le delegazioni nazionali del Bebras provenienti da tutto il mondo propongono i quesiti che hanno elaborato nei mesi precedenti. Il risultato di questo lavoro collettivo è la scelta di un insieme di quesiti, dal quale ogni realtà Bebras nazionale può scegliere quelli che desidera utilizzare nella preparazione della propria gara.

Il modello che viene usato per proporre un *task* a livello internazionale prevede la compilazione di diversi elementi:

- il titolo del *task*, insieme ad un codice che lo identifica univocamente tra tutti i quesiti proposti nei diversi anni;
- una tabella che assegna ad ogni fascia d'età, assimilabile grossomodo ad una tra le categorie viste nella sezione precedente, una difficoltà tra *easy*, *medium* e *hard*; nella stessa tabella vengono indicati anche gli argomenti toccati dal quesito e una lista di parole chiave che identificano specificamente l'argomento;
- il testo del quesito, corredato da immagini che ne aiutano la comprensione;
- la domanda vera e propria;
- la modalità di risposta consigliata, ad esempio "risposta aperta" o "risposta multipla";
- la risposta corretta, accompagnata da una spiegazione operativa;
- la sezione "*It's informatics!*", tradotta in italiano come "Anche questa è informatica", che spiega il legame tra il *task* e l'informatica e fornisce ulteriori spunti di approfondimento;
- un insieme di risorse, grafiche e testuali, da poter utilizzare per inserire il quesito nelle piattaforme nazionali.

I *task* vengono scambiati in lingua inglese. Per inserirli all'interno dei sistemi di gara nazionali, compreso quello italiano, è necessario compiere un'operazione di traduzione. Questo processo presenta tutte le caratteristiche di un'interpretazione, piuttosto che una semplice traduzione, poiché è più importante rendere comprensibile il senso di ciò che viene richiesto alla squadra, rispetto al significato letterale della domanda.

La preparazione del quesito per la gara non termina con la sua traduzione, ma è necessario effettuare una conversione di formato dal modello internazionale, ad un formato

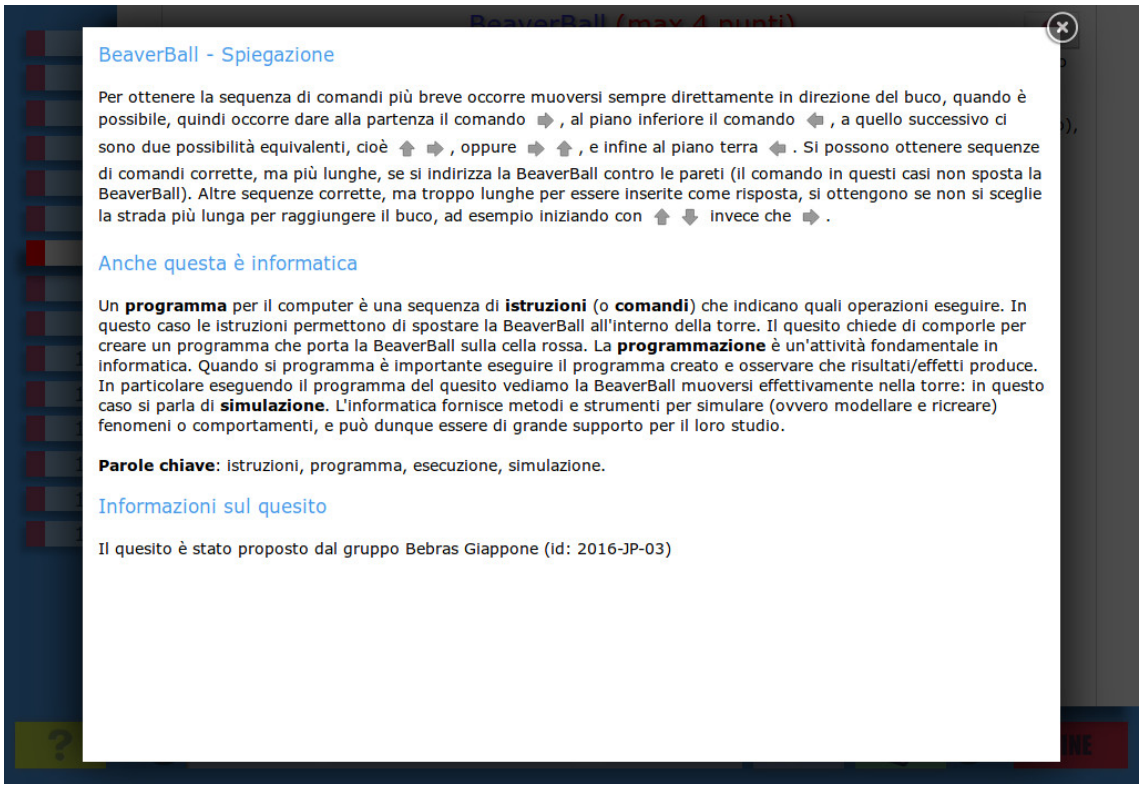

Figura 2.3: Esempio della spiegazione della risposta e della sezione "Anche questa è informatica" che viene mostrata alla squadra in fase di revisione della gara.

che il sistema di gara è in grado di interpretare. Questo formato di fatto è diventato un formato stabilito, tanto che gli è stata dedicata la sezione 4.3.1. Alla fine di questo processo si ottiene un risultato simile a quello visibile nella figura 2.2, per la parte interattiva, e 2.3, per la parte di correzione.

L'ultimo passo da compiere è la definizione dei gruppi di quesiti, i *test*. Ogni quesito viene assegnato ad una tra le categorie viste in precedenza, a seconda della difficoltà, oppure vengono create più versioni dello stesso quesito, con modalità di interazione o di valutazione differenti le une dalle altre, e vengono assegnate a categorie diverse.

Capitolo 3

Vincoli progettuali imposti dal sistema esistente

Il sistema che è attualmente in uso per lo svolgimento delle gare Bebras italiane, presenta numerose caratteristiche peculiari che meritano una trattazione più specifica.

Alla sezione 3.1, vengono descritte le principali criticità imposte dal contesto dove il sistema si trova ad operare. Una panoramica sulle modalità di sviluppo del sistema e le conseguenze che queste hanno avuto sul prodotto creato sono argomento della sezione 3.2. Alla sezione 3.3 è si spiega l'approccio usato per aggiungere nuove componenti *software* al sistema, preservando il modo con cui viene utilizzato durante lo svolgimento delle gare. Infine, nella sezione 3.4, si applica quanto appena detto sull'esempio dell'esportazione dei quesiti al di fuori del sistema di gara.

3.1 Contesto e criticità

Il *software* di gara opera principalmente nel contesto specifico delle scuole italiane. L'ambiente scolastico italiano introduce numerosi vincoli e punti critici, tanto che per affrontarli non è sufficiente né utilizzare un *software* con funzionalità *standard*, né adattarne uno già presente sul mercato, ma bisogna disporre di un sistema costruito appositamente.

Prendendo parte ad una sessione di supporto allo svolgimento delle gare oppure osservando il tipico svolgimento di una gara, si notano diverse criticità, che il sistema a disposizione deve necessariamente essere in grado, in qualche modo, di gestire:

- obsolescenza dei laboratori scolastici italiani: questi laboratori sono provvisti principalmente di macchine di vecchia generazione, i cui sistemi operativi non supportano tutte le tecnologie moderne, rendendo necessario sviluppare un'interfaccia di gara che gestisca specificamente le problematiche che si vengono a creare;

- uso frequente di dispositivi mobili per lo svolgimento della gara: proprio perché la strumentazione attuale della strumentazione delle scuole rende così critico lo svolgimento della gara, molte squadre affrontano la competizione usando diversi tipi di dispositivi mobili ed è, quindi, molto importante fornire un supporto di prim'ordine anche a questi ultimi;

- i quesiti mostrati devono essere renderizzati completamente in una singola schermata dell'applicazione, mantenendo consistenti le modalità di visualizzazione e di interazione con il quesito, anche cambiando il dispositivo utilizzato per affrontare la gara;

- inaffidabilità della connessione alla rete Internet nelle scuole: nei laboratori di numerose scuole d'Italia, soprattutto in particolari aree geografiche, non è possibile fare affidamento sul funzionamento stabile della Rete, costringendo il sistema ad essere capace di operare in assenza di connessione e ad usarla solo nei momenti in cui è assolutamente inevitabile; questo va di pari passo col supporto dei dispositivi mobili, che spesso sfruttano una connessione a pacchetto dati: se il sistema è in grado di funzionare in assenza di connessione, da questo punto di vista, funzionerà correttamente anche sui dispositivi mobili.

- elevato numero di gare simultanee: vedendo di anno in anno i numeri delle squadre partecipanti alla competizione crescere, ma avendo comunque a disposizione solamente una settimana per lo svolgimento delle gare, una delle criticità principali è sia garantire l'accesso alla piattaforma di gara a diverse squadre in contemporanea, sia gestire correttamente molte gare concorrenti; in una parola, il sistema deve essere perfettamente in grado di mantenere alta la soglia della disponibilità.

Il sistema gestisce quindi le criticità in questo modo:

- supporta sistemi operativi e versioni di Web *browser* ormai datate, al fine di consentire una visualizzazione corretta dell'interfaccia di gara;

- cerca di utilizzare la connessione alla rete Internet in maniera parsimoniosa: viene usata solo all'avvio e al termine di una gara; all'avvio viene scaricato tutto il pacchetto contenente ogni quesito ed ogni risorsa grafica che saranno mostrati in gara, cosicché non sia necessario usare la connessione durante la gara; al termine, la connessione viene impiegata solo per inviare le risposte al *server*, conservandole in caso di errore in una memoria locale;

- utilizza un'interfaccia costruita appositamente che si ridimensiona secondo le misure dello schermo del dispositivo usato, ridimensionando coerentemente, senza spostarli, gli elementi del quesito visualizzato;

- supporta i dispositivi mobili quasi nativamente, impiegando la connessione solo quando è necessario e riuscendo a gestire una grande varietà di risoluzioni di visualizzazione dell'interfaccia di gara diverse;

- mantiene alta la disponibilità, riuscendo a gestire un gran numero di connessioni simultanee e salvando lato *server* una copia di tutte le richieste effettuate dai *client* per la consegna delle risposte.

3.2 Modalità di sviluppo del sistema di gara

Le componenti *software* che fanno parte del sistema di gara sono state sviluppate senza una progettazione preliminare e spesso sulla base di urgenze emergenti. Inoltre, il sistema di gara è stato completamente ideato e implementato da un singolo individuo esterno ad ALaDDIn, il gruppo responsabile delle gare Bebras. Prima di questo lavoro, mancava del tutto una documentazione delle varie componenti, né erano presenti dei commenti, nella maggior parte delle righe di codice; non venivano nemmeno utilizzati strumenti che coadiuvassero lo sviluppo, quali un sistema di controllo versione del codice, un sistema di integrazione continua delle modifiche effettuate oppure dei test di unità. La mancanza di una documentazione e l'assenza di un sistema di controllo versione influivano negativamente sulla possibilità di un eventuale passaggio di consegne per quanto riguarda lo sviluppo del sistema.

Lo sviluppatore aveva pensato solo ad un metodo che garantisse un flusso di lavoro non distruttivo verso i moduli di produzione: le cosiddette versioni di sviluppo, versioni aggiuntive dei *file* nella base di codice con suffisso `_zz`. Queste versioni di sviluppo non

venivano usate esclusivamente come delle semplici copie delle versioni di produzione su cui apportare le modifiche per sperimentarle, ma rappresentavano soprattutto un abbozzo di meccanismo di *ridondanza* per garantire il funzionamento del sistema nel caso di modifiche durante i periodi di gara, quando un errore potrebbe essere fatale per il funzionamento della competizione.

Osservando direttamente il codice del sistema precedente alla realizzazione di questa tesi, si nota che la maggior parte di esso appare, qualitativamente parlando, discutibile: poco organizzato, difficilmente mantenibile e molto poco modulare. Molto probabilmente, la mancanza di organizzazione è dovuta ad una stesura frettolosa del codice, fatta, nella maggioranza dei casi, a ridosso dei periodi di gara.

Un esempio di quanto detto si può trovare nel fatto che non esiste alcun vincolo architetturalmente imposto ad accedere a qualunque contenuto nel *database*, anche se di fatto le componenti *software* sviluppate operino solo sulle informazioni che sono necessarie per l'espletamento del loro compito primario.

In aggiunta, la separazione dei compiti non viene rispettata nemmeno nella divisione delle componenti *software* del sistema. Valutando la storia evolutiva del sistema, appare chiaro che la divisione corrente tra le componenti non è frutto di una progettazione specifica basata sull'analisi delle esigenze di ogni utente coinvolto, ma si è venuta a creare progressivamente, versione dopo versione, per sopperire alle urgenze che sopraggiungevano in quel momento. Ogni componente, dopo essere stata costruita, è rimasta definitivamente all'interno della base di codice solo una volta appurato che coprisse sufficientemente il caso d'uso per cui era stata pensato. Riassumendo, è stata questa progressiva evoluzione che ha portato ad avere una componente per ogni attore del sistema e non viceversa.

Nonostante questo, il sistema di gara è riuscito ad affrontare ormai diverse edizioni della competizione e si è dimostrato soddisfacente per lo scopo e molto robusto nel corso del suo utilizzo. Valutando infatti la qualità delle idee che sono state messe in gioco per affrontare le criticità di cui si è già parlato, queste risultano ingegnose e per nulla scontate.

Dopo un'attenta analisi del codice del sistema di gara, è risultato molto coeso e poco affine ad una riorganizzazione, che avrebbe costituito un lavoro molto dispendioso a livello di risorse e che si sarebbe protratto per lungo tempo. Per affrontare questo lavoro, si è deciso di non predisporre il sistema all'aggiunta diretta di nuove componenti *software*, riorganizzandolo in qualche forma o riscrivendolo daccapo, ma di mantenerlo limitatamente al suo compito primario di sistema esclusivamente a supporto dello

svolgimento delle gare. In questo modo, è stato però necessario valutare delle nuove modalità di aggiunta di applicativi che non mettessero a rischio la collaborazione dell'autore originale del sistema, come descritto alla sezione successiva.

Ovviamente, dovendo mantenere il *software* di gara e l'esperienza utente associata tale e quale a quella esistente prima della messa in opera di questo lavoro di tesi, si è quantomeno cercato di mitigare i problemi noti. Ad esempio si è sopperito alla mancanza di documentazione con l'analisi minuziosa dell'architettura e delle tecnologie delle varie componenti del sistema e dei tipi di dato fondamentali gestiti. Inoltre, è stato affrontato anche il mancato utilizzo di un sistema di controllo della versione: esclusa la possibilità di imporne l'utilizzo allo sviluppatore del sistema, soluzione dimostratasi non viabile, si è pensato di creare un meccanismo che generasse una nuova versione del codice prodotto dallo sviluppatore ad ogni intervallo di tempo stabilito. Sebbene questo metodo non fornisca un commento alla storia evolutiva, permette comunque di mantenere in memoria ogni versione, consentendo un ripristino in caso di problemi durante lo sviluppo.

3.3 Realizzazione di nuove funzionalità

Uno degli obiettivi di questo lavoro è stato quello di creare le basi per aggiungere nuove componenti al sistema di gara attuale e fornire degli esempi funzionanti di modalità operativa. Ovviamente, l'aggiunta di componenti è vincolata al corretto compimento dello scopo principale dello *staff* Bebras, ossia le gare annuali. Il sistema di gara è gestito dal creatore del sistema stesso e lo staff Bebras non desidera perdere la sua collaborazione stabile. Quindi, l'obiettivo finale di questo lavoro è stato trovare un modo di coniugare il *software* originale con l'aggiunta di nuove componenti. Queste componenti forniranno nuove funzionalità, tra cui la più importante è sicuramente quella che consente l'esportazione di alcune informazioni che negli anni sono state raccolte all'interno del sistema.

Prima di questa tesi, non esisteva nessun metodo che consentisse di utilizzare in momenti esterni alle gare le informazioni immagazzinate nel sistema. Ciò è il risultato diretto della scarsa progettazione e delle modalità di sviluppo del sistema di gara descritte alla sezione precedente, che avevano portato ad avere un *software* privo di moduli direttamente riutilizzabili al di fuori del contesto dove sono stati creati.

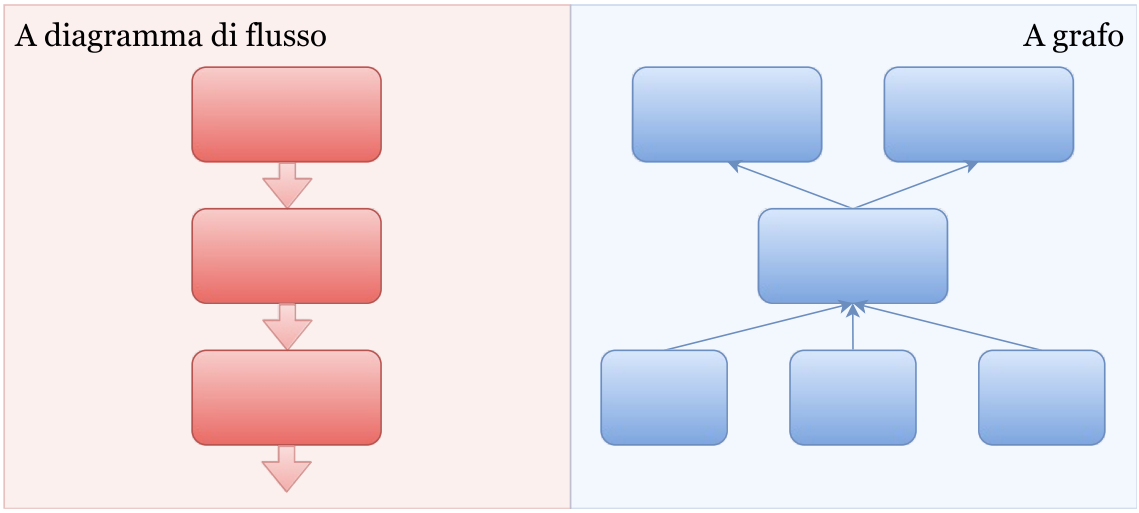

Figura 3.1: I due tipi di decomposizione di un sistema in moduli illustrati da Parnas in [1].

Riferendosi a quanto illustrato da Parnas in [1], si delineano due modi di concepire e portare avanti lo sviluppo di un sistema. Parnas evidenzia sostanzialmente due criteri per decomporre i sistemi al fine di velocizzarne lo sviluppo: uno che si definirà “a diagramma di flusso” e l’altro che sarà chiamato “a grafo”, entrambi esemplificati in figura 3.1.

Il primo criterio consiste nell’ideare i diversi moduli del sistema, a livello di codice, come una sequenza di funzioni, ognuna dipendente dalla precedente, in una conformazione a diagramma di flusso. Un esempio è schematizzato nella parte sinistra della figura 3.1. Secondo Parnas, questo modo di procedere garantisce che a lungo andare non sia possibile per esempio isolare due tra i moduli presenti per inserirne uno nuovo che dipende da questi.

Il secondo metodo presentato da Parnas prevede invece di sviluppare dei moduli che siano il più possibile autonomi: l’unica dipendenza che rimane è quella derivata da interfacce. Questo genera, come si può notare nella parte destra di figura 3.1, una conformazione dei moduli a grafo. Le interfacce sono dei contratti ben definiti che consentono di delineare delle pre-condizioni e delle post-condizioni per ogni modulo, in maniera tale che ognuno di essi sia in grado di funzionare dipendendo solo dalle prime e generando in uscita le seconde.

È il secondo metodo che Parnas definisce come quello più lungimirante per quanto riguarda la manutenibilità della base di codice. Oltre a consentire una documentazione rapida dei diversi moduli, ne permette la sostituzione immediata, operando essi, sia in entrata che in uscita, solo per mezzo di interfacce.

Tornando a parlare del sistema di gara e rapportandolo ad uno dei due modelli proposti, si può dire che, nella configurazione corrente, è possibile vederlo come un esempio di quella a diagramma di flusso. Ogni funzione presente a livello di codice è fortemente dipendente dall'ecosistema che gli sta intorno: in certi casi le funzioni sviluppate non sono caratterizzate da parametri specifici, ma vanno addirittura a modificare uno stato globale. Questo ne rende molto difficile la transizione verso un modello a grafo e quindi mina la possibilità di aggiungere altre componenti al sistema.

I metodi risultati viabili per poter aggiungere delle nuove funzionalità al sistema sono pertanto due:

- cercare di capire il funzionamento interno del sistema al fine di comprendere com'era possibile progettare del *software ad hoc* che si adattasse perfettamente alla conformazione a diagramma di flusso;
- sfruttare qualche tipo di interfaccia all'interno del sistema e basare il nuovo *software* su questa interfaccia.

Il primo metodo è stato intrapreso nello sviluppo del sito di supporto, spiegato alla sezione 5.1 "Il modulo *helpdesk*" del capitolo 5, mentre il secondo è la modalità con la quale si è progettato e sviluppato il sistema di esportazione dei quesiti, discusso a livello tecnico alla sezione 5.2, sempre al capitolo 5.

3.4 Esportazione dei quesiti

Un esempio pratico di decomposizione di *software* declinato nel caso specifico del sistema di gara è dato dalla possibilità di utilizzo dei quesiti Bebras al di fuori dei momenti di gara. L'argomento è messo in risalto in questa sezione, poiché è uno dei risultati più importanti ottenuti in questo lavoro ed è un esempio pratico di adattamento di un sistema sviluppato secondo il modello a diagramma di flusso verso quello a grafo.

Per proporre i quesiti durante una lezione scolastica e quindi sfruttare il loro potenziale formativo, occorre sia essere in grado di sceglierli a seconda del bisogno tra tutti quelli disponibili, sia avere un metodo comodo per prenderne visione. Potrebbe essere necessario utilizzare quesiti destinati esclusivamente ad una determinata fascia d'età oppure quesiti che trattano tutti lo stesso argomento.

Appare quindi chiara l'esigenza di nuovi strumenti che permettano lo sfruttamento dei quesiti Bebras e l'estrapolazione dei relativi metadati, in maniera indipendente rispetto alla piattaforma di gara. In una parola, occorre un sistema di *esportazione* dei quesiti accessibile, documentato e utilizzabile in maniera semplice da chiunque, soprattutto esterno allo *staff* Bebras.

Considerando il sistema di gara al lordo di tutte le sue componenti e prima di questa tesi, non era previsto un meccanismo efficiente che consentisse la ricerca di un quesito particolare all'interno di quelli utilizzati in molti anni di competizioni. Il procedimento che finora si era impiegato è stato quello di ricercare il quesito direttamente tra i *test* proposti nelle varie edizioni della competizione. Il metodo era però tutt'altro che efficace: presupponeva che il docente intenzionato ad usare il *task* Bebras in classe conoscesse a priori quello che gli interessa oppure fosse al corrente dell'anno in cui era stato proposto e della categoria per cui era previsto. Inoltre, non era possibile in nessuna maniera la ricerca per argomento. Un altra possibilità, limitata allo *staff* Bebras, era quella di ricercare un quesito direttamente nella base di dati oppure nel *file system* del *server*. Anche questa modalità operativa risultava poco pratica, poiché non consentiva di visualizzare la forma interattiva del quesito durante la ricerca, costringendo a ricordarne il nome o qualche sua caratteristica.

Un altro problema si poneva nel momento in cui il docente, o la persona interessata, doveva interagire con il quesito oppure mostrarlo ad un pubblico. L'unico metodo a disposizione era quello di individuare l'edizione e la categoria del quesito desiderato, come nel paragrafo precedente, e avviare una sessione di allenamento, simulando quella specifica configurazione di gara. A questo punto bisognava selezionare manualmente il quesito desiderato e interagirci. Per ottenerne la correzione e la spiegazione della risposta esatta, andava infine confermata l'intera sessione di gara. Questo lungo processo ripetuto per più quesiti, diventava oggettivamente insostenibile per una lezione scolastica.

Ideare uno strumento di questo tipo non ha coperto solo i casi d'uso descritti nei paragrafi precedenti, ma ha consentito l'apertura alla costruzione di diversi tipi di sistemi. Alcuni esempi possono esssere:

- una piattaforma di gara totalmente personalizzata, costruita sulla base di esigenze specifiche, anche molto differenti da quelle del Bebras; ad esempio, una scuola potrebbe ideare una competizione interna all'istituto sfruttando i quesiti già visti nelle edizioni precedenti del Bebras;

- un sistema di ricerca dei quesiti, basato non solo sull'anno di comparizione e sulla categoria Bebras, ma direttamente sulla difficoltà propria di ogni quesito e sull'argomento trattato;
- un sistema di allenamento mirato, totalmente differente da quello esistente e che permetta di addestrarsi su quesiti che trattano tutti lo stesso argomento o abbiano la stessa difficoltà.

Il sistema, ricalcando il modello di decomposizione in moduli a diagramma di flusso, non accoglieva facilmente una modifica del genere e questo era visibile in due fatti:

- non esisteva un formato *standard* per i quesiti: sebbene in questo lavoro sia stato descritto estensivamente il formato che un quesito doveva avere per essere correttamente inserito nel sistema di gara, esso non aveva mai subito un processo di standardizzazione; non si poteva nemmeno fare affidamento sulla stabilità del formato, poiché non ne erano mai stati documentati formalmente gli elementi che lo componevano;
- i blocchi e le risorse grafiche di un quesito erano in gran parte dipendenti dalla piattaforma di gara: per interpretare i blocchi di spazio che compongono la descrizione di un quesito e le risorse grafiche associate ad ogni blocco, venivano sistematicamente fatte delle assunzioni a livello del codice *client* della piattaforma di gara; questo rendeva complicato renderizzare un quesito senza utilizzare il codice interno alla piattaforma.

I problemi appena elencati sono stati superati mettendo in atto la situazione schematizzata nella figura 3.2. Il sistema di gara è rappresentato sulla sinistra secondo la configurazione già introdotta a diagramma di flusso. La linea rossa è il formato *standard* per l'esportazione dei quesiti, un formato fissato e stabile che permette alle applicazioni che vi dipendono di farci affidamento, sfruttando una dipendenza da accoppiamento debole. La parte destra della figura identifica i nuovi strumenti sviluppati, che dipendono dall'interfaccia creata da questo formato *standard*.

Evitando di perderne la preziosa collaborazione, lo sviluppo di questo formato è stato concordato insieme al creatore originale del sistema che ha lavorato per rendere più coerente quello che di fatto era diventato il formato di inserimento dei quesiti. Se il formato precedente, descritto alla sezione 4.3.1 "Quesiti e gruppi di quesiti", prevedeva

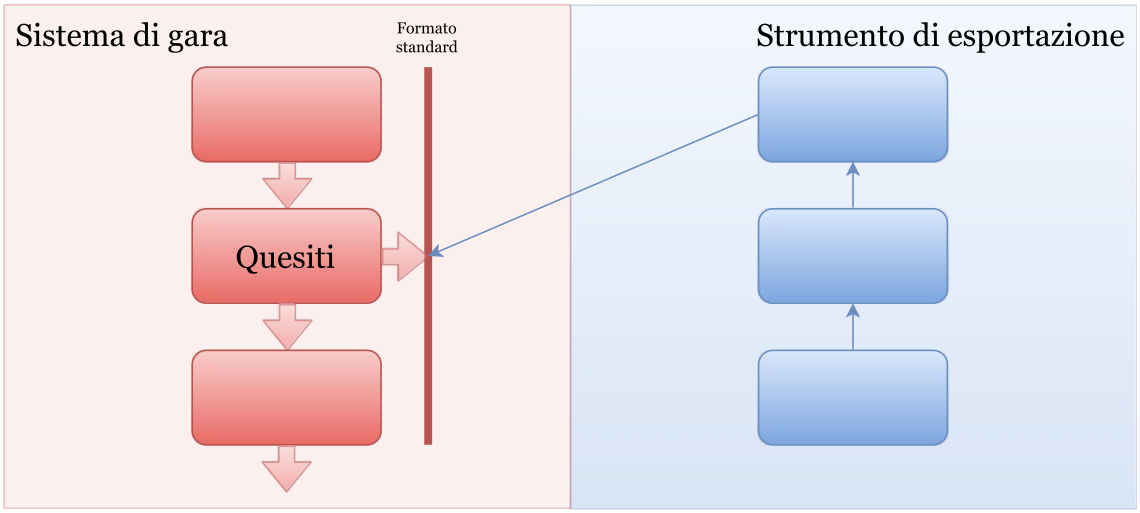

Figura 3.2: Il *formato standard* per l'esportazione dei quesiti utilizzato come interfaccia per i nuovi strumenti.

l'impiego di diversi file dislocati all'interno di una specifica cartella del *file system*, quello corrente utilizza un singolo *file* JSON ben definito, che ha dato avvio al processo di standardizzazione del formato, rendendolo stabile e permettendo la costruzione di *software* che lo utilizza direttamente.

In questa maniera, il formato *standard* è stato posto come post-condizione al sistema di gara originale e come pre-condizione ai nuovi strumenti creati per l'esportazione dei quesiti. Dalla parte del sistema di gara è quindi stato possibile utilizzare il principio dell'incapsulamento di cui parla Parnas, esponendo all'esterno un formato ben definito, ma nascondendone i dettagli tecnici.

A partire da questo formato, sono stati implementati gli strumenti descritti alla sezione 5.2, al capitolo 5.

Capitolo 4

Analisi del sistema di gara

Riassumendo in una frase quanto detto nel capitolo precedente, il sistema di gara attualmente in uso nelle gare Bebras in Italia è un insieme creato *ad hoc* di componenti *software* che permettono la gestione, grazie alla rete Internet, di gare a distanza e di più squadre in contemporanea. Nello stesso tempo, è in grado di fare fronte a diverse criticità proprie nel contesto delle scuole italiane, in cui si trova a funzionare.

Questo capitolo fornisce una descrizione tecnica del funzionamento interno delle diverse componenti del sistema di gara. Nella sezione 4.1 “Architettura del sistema”, viene presentata globalmente l’architettura corrente del sistema. Nella sezione 4.2 “Tecnologie impiegate” si descrivono le tecnologie che sono state impiegate per la sua implementazione, facendo riferimento alla storia evolutiva presentata alla sezione 2.2. Infine, alla sezione 4.3 “Dettagli implementativi”, sono mostrati alcuni dettagli interessanti che riguardano da vicino l’implementazione e le scelte tecniche compiute. Nello specifico, viene fatta una panoramica sulla struttura della base di dati utilizzata e sui tipi di dato fondamentali, quali: i quesiti, alla sezione 4.3.1, le squadre e i docenti, alla sezione 4.3.2, e le gare, alla sezione 4.3.3. Tutti i risultati che sono descritti nelle sezioni che seguono riguardano la versione del sistema che attualmente funziona come piattaforma di gara del Bebras italiano.

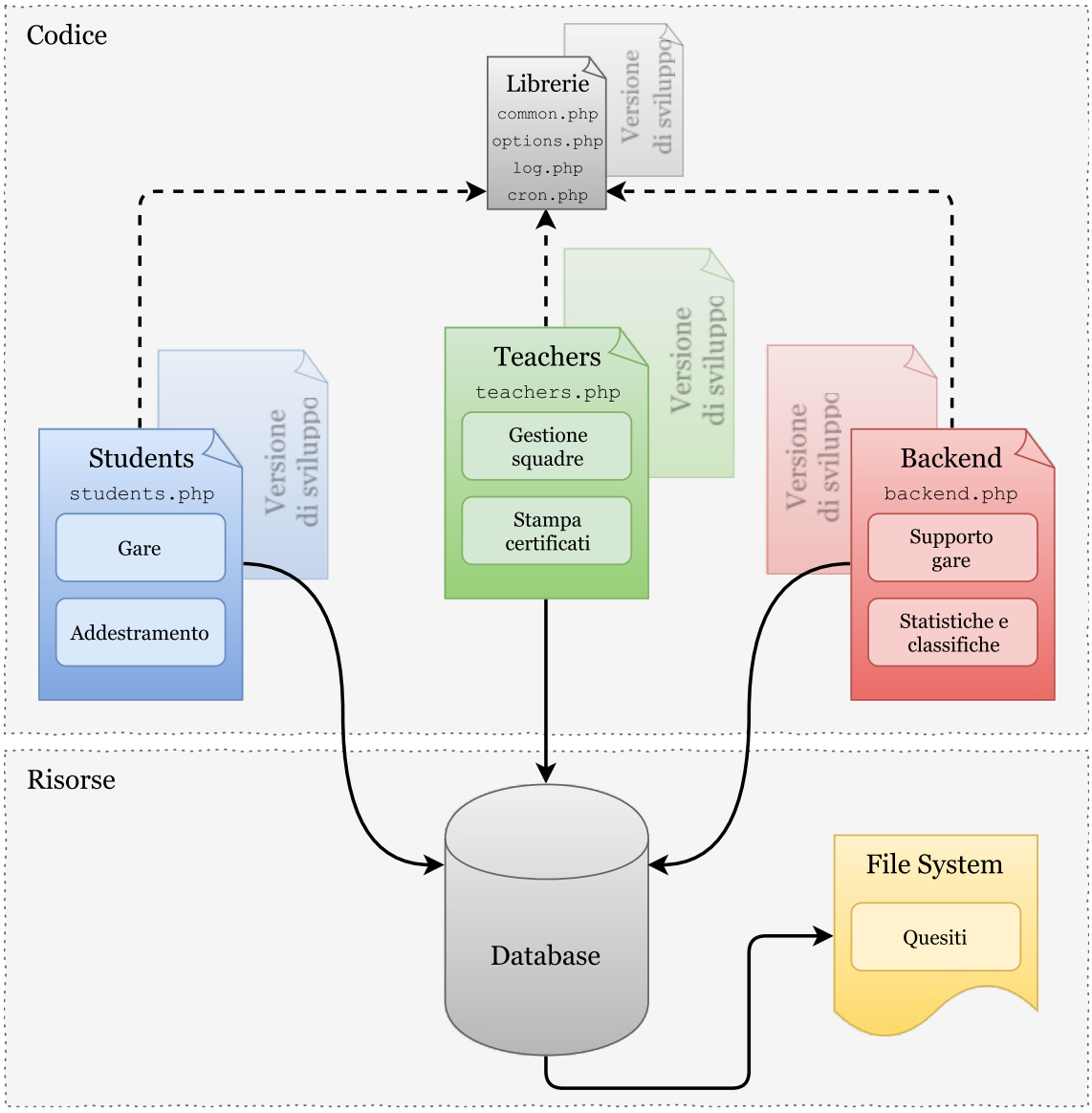

Figura 4.1: Schema generale dell'architettura del sistema e delle interazioni tra le sue componenti.

4.1 Architettura del sistema

La figura 4.1[1] riporta lo schema dell'architettura generale del sistema di gara, nel quale le frecce descrivono le interazioni tra le diverse componenti. Nei prossimi saranno descritti i dettagli di tale architettura, seguendo passo passo lo schema raffigurato nell'immagine.

[1]Le diverse componenti nella figura sono ispirate a quanto detto in [5].

Il sistema di gara nasce nei primi anni Duemila come piattaforma per lo svolgimento degli esami ECDL. L'architettura era basata sul paradigma *client-server*. La parte *server* era del tutto simile a quella di un'applicazione Web, ma veniva fatto uso di un *client* creato *ad hoc*, che era necessario installare sulle macchine da utilizzare durante la gara. A seguito delle diverse trasformazioni e in risposta alla crescita del numero delle squadre coinvolte e alla necessità di un *software* multi-piattaforma, la soluzione *client ad hoc* impiegata non era più adatta e si è avviata una riscrittura di questa parte, tesa a trasformare il sistema in una vera e propria applicazione per il Web. Da questo punto in avanti, l'architettura non ha subito grosse modifiche nel corso delle revisioni, fatta eccezione per l'aggiunta di altre componenti *software*.

Siccome l'applicazione viene servita tramite la rete Internet, il codice *server* è pensato per operare su una macchina diversa rispetto a quella sulla quale girerà il codice *client* e le due componenti comunicano tramite il protocollo HTTP.

La parte *server* di questo sistema utilizza anche una base di dati, al fine di mantenere ben organizzati i dati riguardanti gli utenti del sistema, come insegnanti e squadre, i quesiti e le varie fasi della gara, incluso lo storico dei risultati. I dati che modellano le due tipologie di utenti e le gare nei due stati, in corso di svolgimento e già svolte, nel caso di questo sistema hanno una caratteristica che li accomuna: uno schema poco variabile che permette di salvarli agevolmente in una tabella con dei campi fissi. La modalità di memorizzazione che viene usata correntemente è infatti una base di dati a modello relazionale. Per quanto riguarda, invece, le informazioni sui quesiti proposti nelle competizioni, queste risiedono nel *file system*. All'interno della base di dati rimane solamente un riferimento alla cartella del *file system* dov'è memorizzato ogni quesito. Se ne parlerà in maniera più approfondita nella sezione 4.3.3 "Gare". La scelta di un *database* relazionale come supporto per la memorizzazione a lungo termine dei dati su insegnanti, squadre e gare ha contribuito a realizzare un'applicazione che rispondesse all'esigenza di gestire migliaia di gare contemporaneamente, permettendo di memorizzare una mole relativamente grande di dati, che rimanesse prontamente disponibile in maniera indicizzata, indipendentemente dalla tecnologia che ne fa uso.

Per avere una vista di insieme, quello che è più importante definire è la divisione interna tra le componenti del sistema di gara. La maggior parte del codice e delle altre risorse del sistema, che non stanno nella base di dati, sono raccolte nel *file system*.

All'interno della cartella radice nel *server*, si trova un buon numero di *file* e cartelle che ricoprono ruoli diversi:

- la cartella `questions` raggruppa tutte le cartelle dei quesiti, il cui contenuto è analizzato alla sezione 4.3.1 "Quesiti e gruppi di quesiti";
- i *moduli* del sistema sono le unità di base del *software*; ricordando quanto detto nella sezione 2.2 "Il sistema di gara", si tratta della piattaforma di gara, `students.php`, d'ora in poi chiamato *students*, del sito docenti, `teachers.php`, d'ora in poi chiamato *teachers* e, infine,del sito di monitoraggio, `backend.php`, d'ora in poi chiamato *backend*;
- le *librerie* sono dei *file* che racchiudono tutto il codice utilizzato da più componenti, al fine di non dover riscrivere le funzioni di utilità per ogni modulo; si tratta di `common.php`, `config.php`, `log.php` e `cron.php`;
- le *versioni di sviluppo* dei moduli sono sostanzialmente delle versioni speculari dei moduli appena presentati, ma comprendono delle funzionalità ancora in fase di sperimentazione o comunque non abbastanza rifinite per essere portate in produzione; data la mancanza di un sistema di controllo delle versioni del *software*, come spiegato alla sezione 2.2 "Il sistema di gara", questi file si differenziano dai moduli di produzione grazie al suffisso "`_zz`".

Considerando la situazione attuale del sistema, ognuno dei moduli appena presentati costituisce un punto di accesso al sistema per un determinato tipo di utente, tra quelli descritti alla sezione 2.3 "Soggetti coinvolti":

- Il modulo *students* fornisce la piattaforma di gara ed è l'interfaccia di accesso al sistema per le squadre che affrontano la competizione o per gli studenti che desiderano allenarsi utilizzando i quesiti proposti nelle edizioni passate;
- *teachers* fornisce il sito docenti ed è il modulo attraverso il quale si permette agli insegnanti l'iscrizione e la gestione delle squadre di cui sono responsabili, insieme alla stampa dei certificati di partecipazione per ogni studente;
- *backend* fornisce il sito di monitoraggio ed è il modulo nato affinché lo *staff* potesse controllare le gare durante il loro svolgimento; in una versione successiva è

arrivato ad occuparsi anche di calcolare le statistiche per ogni edizione annuale delle competizioni e funzionare da interfaccia di controllo del sistema per lo sviluppatore.

Come è lecito immaginare, i moduli sono certamente in grado di scambiarsi informazioni tra di loro, ma, osservando l'architettura in figura 4.1, si vede che non comunicano direttamente, ad esempio attraverso lo scambio di file o messaggi, ma utilizzano esclusivamente la base di dati sottostante.

Per quanto riguarda le versioni di sviluppo dei moduli, queste svolgono principalmente due funzioni: una primaria di contenere le modifiche ai moduli di produzione e l'altra di funzionare come meccanismo di ridondanza in caso di errori. Un esempio chiarisce meglio questo concetto di ridondanza: si considerino i moduli di produzione `backend.php` e `students.php` e le due versioni di `common.php`, quella di produzione e quella di sviluppo, ossia `common_zz.php`. `common.php` è una libreria dove vengono raccolte tutte le funzioni di utilità. Osservando le inclusioni delle librerie, si può notare che `backend.php` include `common_zz.php`, mentre `students.php` include `common.php`. Questo perché se occorre eseguire una modifica, questa sarà fatta prima su `common_zz.php` e solo successivamente portata a `common.php`, permettendo a `students.php` di continuare a funzionare anche nel caso `common_zz.php` contenga un difetto fatale. La funzionalità è particolarmente utile nel periodo delle gare, quando un singolo errore potrebbe portare a conseguenze catastrofiche per la loro riuscita. Sebbene la soluzione non sia completamente risolutiva e preveda che lo sviluppatore sia al corrente di questo incrocio di flussi, finora è riuscita comunque a garantire l'integrità di sistema durante le fasi critiche.

Parlando della relazione che intercorre tra la base di dati e i moduli *software*, si può dire che l'accesso alle tabelle in essa contenute avviene solo limitatamente alle informazioni necessarie. Ad esempio, il modulo *teachers*, che come già detto si occupa di fornire un'interfaccia ai docenti per aggiungere, modificare e, in generale, interagire con la composizione delle squadre, di fatto agisce in lettura e in scrittura solo sulla tabella che riguarda le squadre e, al più, sulla riga della tabella che riguarda il docente responsabile. Anche il modulo *backend* ha bisogno, come *teachers*, dei dati sui docenti e sulle squadre per mostrare i dati di gara, ma si limita a leggere, e non scrive, i dati nelle relative tabelle. Osservando lo schema della base di dati, disponibile in figura 4.3 a pagina 37, appare chiaro che il modello della base di dati è concepito per rendere il più naturale possibile questa separazione dei compiti.

4.2 Tecnologie impiegate

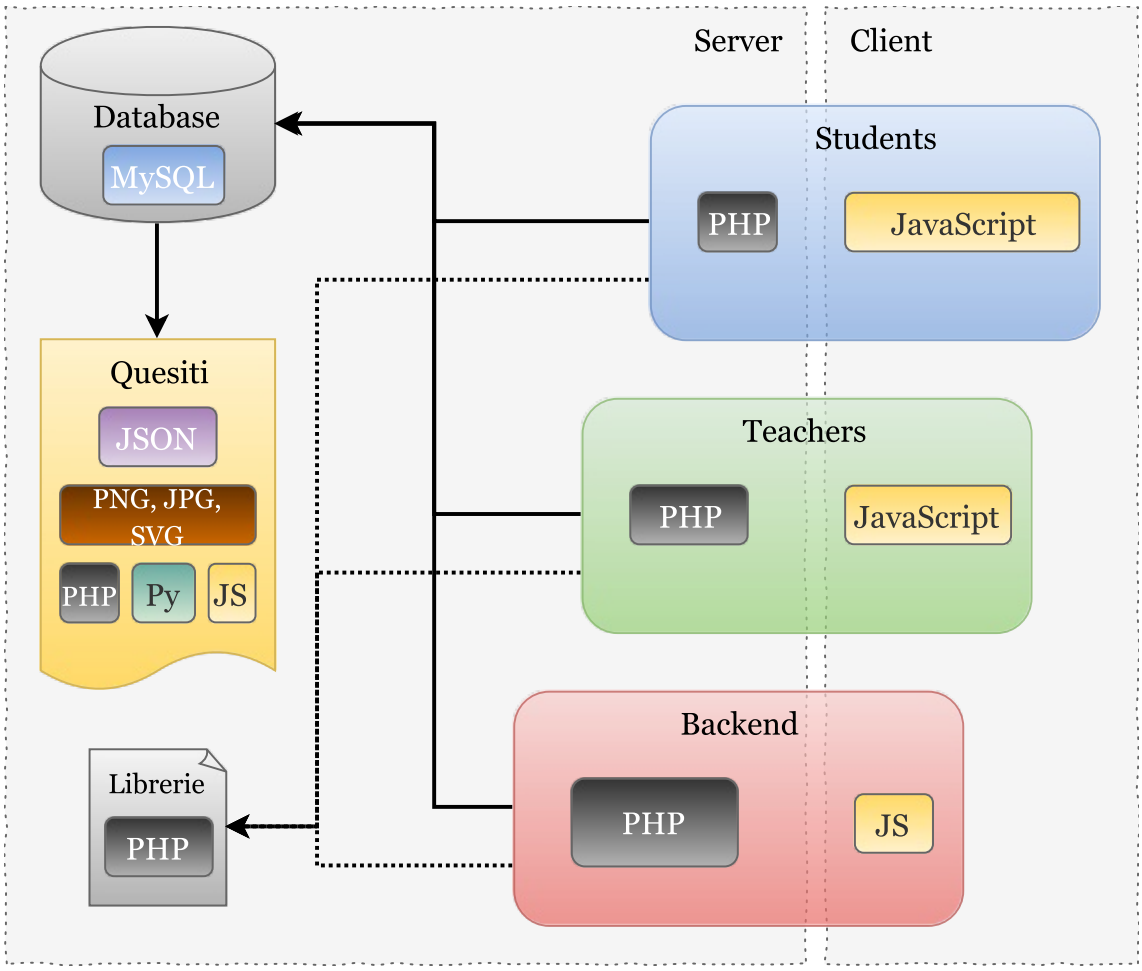

Figura 4.2: Schema generale delle tecnologie impiegate in ogni componente del sistema.

In questa sezione verranno elencate le tecnologie utilizzate per implementare l'architettura descritta nella sezione precedente.

In figura 4.2 è riportato uno schema che associa alle componenti del sistema, le stesse mostrate in figura 4.1, le rispettive tecnologie impiegate.

Come già detto, il *software* analizzato non è stato costruito appositamente per lo svolgimento delle gare Bebras, ma ha subito ben tre profonde trasformazioni nel corso degli anni. È nato originariamente, nei primi anni Duemila, come piattaforma per gli esami

ECDL e quindi criteri di progettazione e la selezione delle tecnologie utilizzate fanno capo al modello tipicamente in uso in quegli anni.

Il sistema di gara è arrivato ad avere la forma di applicazione Web. Conseguentemente sono incluse tre parti: due componenti *software*, quella *server* e quella *client*, e una componente di memorizzazione, la base di dati relazionale.

Nella struttura di un'applicazione Web che deve essere fruita tramite la rete Internet, c'è la necessità di avere un *server* Web che serva i contenuti richiesti tramite il protocollo HTTP. L'utente solitamente richiede questi contenuti al *server* attraverso un Web *browser*. Nel sistema di gara, la situazione non è differente. Il *server* HTTP scelto è Nginx[2], una delle applicazioni di questo tipo più diffuse e più performanti a disposizione. Avere usato in particolare Nginx, rispetto ad un altro Web *server*, non ha vincolato particolarmente il funzionamento del resto del sistema, infatti allo stato attuale sarebbe ancora possibile scegliere un'altra applicazione *server*, senza essere costretti a riprogettare nessuna parte del sistema.

Il linguaggio di programmazione *server-side* è PHP[3]. L'utilizzo di questo linguaggio trova due semplici motivazioni: innanzitutto bisogna considerare la vasta diffusione della macchina virtuale PHP sulle installazioni dei *server* di quel periodo; in secondo luogo, era il linguaggio standard per questo tipo di applicazioni, soprattutto se si considerano le sue funzionalità specifiche per il Web.

PHP è il linguaggio con cui questa componente *software* del sistema è stata originariamente scritta. Nel corso delle trasformazioni, almeno per quanto riguarda il linguaggio di programmazione della parte *server*, non si sono visti cambiamenti. È quindi stato possibile riutilizzare, nella versione corrente, la maggioranza del codice già scritto per le versioni precedenti.

Per quanto riguarda la componente *client*, occorre eseguire un'analisi più complessa. Originariamente l'interfaccia grafica destinata all'utente finale richiedeva un client *ad hoc*

[2]Nginx è un *server* web considerato leggero, ma ad alte prestazioni; riesce a servire i *file* statici molto più rapidamente di altri *server* e utilizza un meccanismo asincrono ad eventi, piuttosto che un paradigma *multi-thread*. Altre informazioni si possono trovare sul sito (`https://nginx.org/`).

[3]Maggiori informazioni sul linguaggio PHP si possono trovare sul suo sito web (`http://www.php.net/`). È un linguaggio sviluppato a partire dal 1995, al fine di facilitare la gestione delle pagine web personali in uso in quel periodo.

era scritta con la tecnologia Adobe Flash[4]. Questa tecnologia consentiva di realizzare in maniera facile un'interfaccia grafica interattiva, al costo di non essere particolarmente portabile sui sistemi mobili. Il cambiamento è avvenuto a partire dalla seconda revisione del *software*, risalente al periodo Kangourou. Vista la crescente richiesta di applicazioni multi-piattaforma e considerata l'ampia diffusione dei dispositivi mobili, in grado di accedere al Web ma incapaci di utilizzare la tecnologia *Flash*, si è avvertita l'esigenza di riscrivere la parte di interfaccia grafica verso l'utente, utilizzando il nuovo standard per le applicazioni Web: le tecnologie HTML5, CSS3 e JavaScript[5].

Da questa riscrittura, ne hanno sicuramente guadagnato le possibilità di uso del sistema. Un esempio apprezzabile lo si può ottenere prendendo in esame il pessimo stato delle strutture e degli strumenti delle scuole italiane: sono provviste nella maggior parte dei casi di poche macchine, spesso appartenenti alla vecchia generazione, e di una connessione alla rete Internet molto lenta o affetta da malfunzionamenti continui. In questi casi, l'opportunità di poter svolgere le gare con dispositivi mobili propri o *computer* non collegati direttamente alla rete del laboratorio è stata ben accolta dagli insegnanti e dalle squadre; sono anche diminuiti i problemi direttamente legati alle difficoltà di connessione o ai malfunzionamenti delle macchine dei laboratori.

Dal punto di vista del codice, questa trasformazione ha lasciato alcune tracce apprezzabili nel progetto analizzato. Per esempio, alcuni quesiti sono rimasti ancora codificati nel formato precedente, quello che comprendeva *Flash*, e non sono mai stati tradotti in quello nuovo. Questo ha reso obbligatorio modificare lo schema della base di dati, aggiungendo dei campi appositi per distinguere tra i due tipi di quesiti. Il problema sarà approfondito nella sezione 4.3.1 "Quesiti e gruppi di quesiti".

Passando alla base di dati, nella sezione 4.1 "Architettura del sistema" è già stato fatto cenno dell'utilizzo di una base di dati relazionale. Nello specifico, la soluzione *software*

[4]La piattaforma multimediale Adobe Flash, anche conosciuta come Macromedia Flash o Shockwave Flash, è un *software* per uso grafico, teso a facilitare la creazione di contenuti multimediali, come animazioni vettoriali o riproduttori audio e video, per le pagine web. Altre informazioni su *Flash* si trovano sul sito (`https://www.adobe.com/platform/`).

[5]HTML5 è lo standard più recente per quanto riguarda il linguaggio di marcatura per le pagine del *World Wide Web* e, corrispettivamente, CSS3 è quello più recente per il linguaggio che ne concerne la formattazione. JavaScript, inizialmente chiamato ECMAScript, è il linguaggio di programmazione comunemente utilizzato per la *logica di presentazione* delle pagine web. Maggiori informazioni sugli standard si possono ottenere sul sito del World Wide Web Consortium (`https://www.w3.org/standards/`).

utilizzata nel sistema è MySQL[6], uno dei più diffusi RDBMS. MySQL, insieme a Linux, Nginx e PHP, è un componente del cosiddetto "*Stack* LEMP". Avendo già utilizzato Nginx come *server* HTTP e PHP come linguaggio di programmazione del lato *server*, una componente *sofwtare* già ben testata unitamente agli altri due era, quindi, proprio MySQL.

L'ultima parte che rimane da analizzare, è quella che concerne le modalità di codifica e conservazione dei quesiti all'interno del sistema. L'argomento sarà trattato nel dettaglio alla sezione 4.3.1 "Quesiti e gruppi di quesiti", concentrando l'attenzione sul formato di codifica e su come i quesiti vengano interpretati dal sistema. In questo frangente è sufficiente sapere che i quesiti sono progettati per funzionare all'interno di un'applicazione orientata al Web e sono memorizzati nel *file system* del *server*, ognuno nella propria cartella. All'interno di ogni cartella è presente una serie di *file* in un formato stabilito, che il sistema è in grado di interpretare.

Le informazioni direttamente legate alla visualizzazione del quesito sono memorizzate in uno di questi *file*, nel formato JSON[7]. È stato scelto il formato JSON soprattutto per motivi pratici:

- è facilmente leggibile da un essere umano e, al contempo, si presta bene ad essere gestito da un punto di vista implementativo;
- è indipendente dal linguaggio di programmazione utilizzato per interpretarlo[8]; sono già disponibili, infatti, molte librerie capaci di leggerne e scriverne il contenuto, scritte in altrettanti linguaggi;
- la gestione dei *file JSON* in PHP e JavaScript è disponibile nativamente tra le librerie di base del linguaggio.

Per quanto riguarda gli altri *file* presenti nella cartella relativa ad un quesito, questi possono essere sostanzialmente di tre tipi: risorse grafiche, come immagini *raster*

[6]MySQL è un RDBMS, *Relational DataBase Management System*, composto da un *server* e *client* a riga di comando. Aderisce agli standard ANSI SQL e ODBC SQL ed è una tra le basi di dati più usate in ambito Web; Maggiori informazioni su MySQL sono disponibili sul sito (`https://www.mysql.com/`).

[7]JSON, *JavaScript Object Notation*, è un formato principalmente utile alla serializzazione di dati al fine dello scambio di essi tra le applicazioni; il suo impiego particolarmente semplice e immediato, sia dal punto di vista della sintassi, che da quello della lettura e scrittura da parte di un programma. Altre informazioni sono disponibili sul sito (`http://json.org/`).

[8]Sebbene esistano due *standard* diversi, RFC 8259 e ECMA-404, che definiscono il formato JSON e questo sia derivato da un sottoinsieme dello standard ECMA, è generalmente considerato un formato indipendente dal linguaggio utilizzato per interpretarlo.

o immagini vettoriali, risorse testuali, oppure dei semplici *script* di valutazione del punteggio della risposta data.

Precisando ulteriormente, si può dire che:

- se si tratta di immagini, possono essere in vari formati interpretabili da un Web *browser*, come PNG, JPEG, GIF, SVG, ecc.;
- se si parla di testo, sfrutta la codifica UTF-8 e, come linguaggio di marcatura, viene usato BBCode[9];
- per quanto riguarda gli *script* di valutazione delle risposte date, se ne trovano scritti in tre linguaggi di programmazione differenti: PHP, Python oppure JavaScript.

Nel capitolo 5 “Revisione e implementazione” viene illustrato un motore di *rendering* dei quesiti portatile e completamente disponibile lato *client*. In previsione della realizzazione di questa nuova componente *software*, nei quesiti più recenti è stato utilizzato prevalentemente il linguaggio JavaScript, l’unico interpretabile da un Web *browser* moderno.

4.3 Dettagli implementativi

In questa sezione viene descritta in maniera dettagliata la base di dati e vengono illustrati alcuni dettagli implementativi, significativi per il funzionamento del sistema, riguardanti tre tipi di dato: i quesiti e gruppi di quesiti vengono trattati alla sezione 4.3.1, la gestione delle squadre da parte dei docenti responsabili viene esposta alla sezione 4.3.2, mentre la sezione 4.3.3 descrive lo svolgimento delle gare osservando lo scambio di dati tra *server* e *client*. In particolare verranno illustrate alcune problematiche sistema e alcuni casi d’uso specifici di questo sistema di gara, evidenziando di volta in volta le risposte che è capace di dare. Le risposte meritano di essere commentate, poiché si tratta di soluzioni che non sempre sono di immediata comprensione, soprattutto se si valuta la mancanza della documentazione del *software*, ma che in generale risolvono in maniera soddisfacente i problemi che affrontano.

[9] BBCode, *Bulletin Board Code*, è un linguaggio di marcatura del testo, principalmente usato in molti forum; è traducibile direttamente in HTML, ma presenta una semplicazione dei *tag* usati.

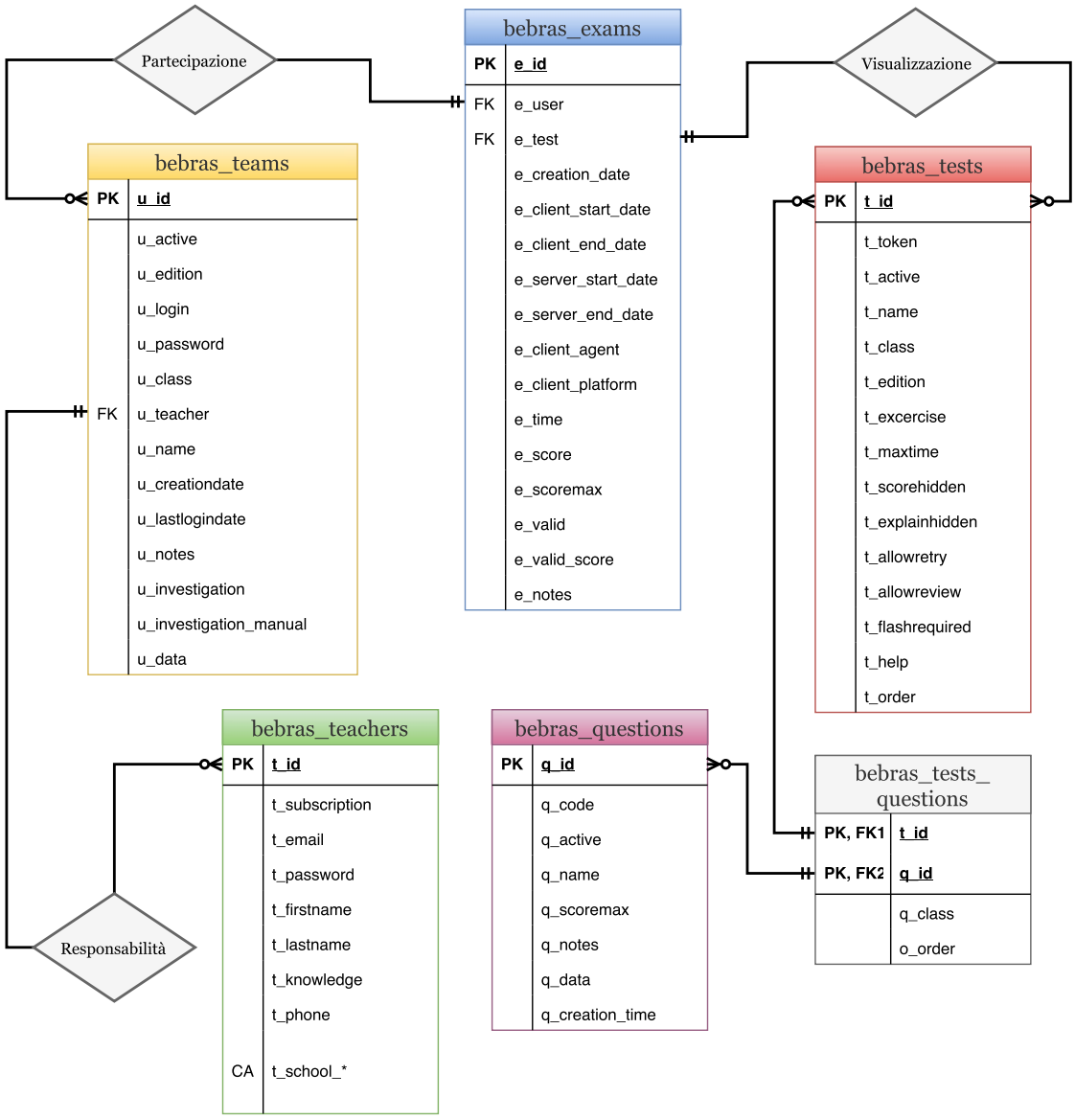

Figura 4.3: Schema delle tabelle e delle relazioni contenute nella base di dati.

Per cominciare l'analisi, bisogna identificare i principali tipi di dato trattati nel sistema: a questo scopo è conveniente partire dalla della base di dati. Se nel codice si notano inevitabilmente le tracce dei diversi processi di revisione, quasi privi di pianificazione e spesso completati a ridosso della settimana di gara, la struttura della base di dati sembra essere rimasta fissata o quantomeno modificata in maniera metodica. Facendo riferimento al *database*, è stato perciò possibile capire esattamente le caratteristiche dei dati trattati nel codice. Ciò ha permesso di ottenere una rappresentazione generale delle categorie di dati fondamentali e di capire le motivazioni delle scelte compiute durante lo sviluppo.

Nella figura 4.3, viene presentato lo schema delle tabelle e delle relazioni contenute nella base di dati del sistema di gara[10]. Nello schema appare chiaro che il sistema ruota intorno a cinque tipi di dato, che ricalcano gli utenti e le funzionalità, descritte nelle sezioni 2.2 "Il sistema di gara" e 2.3 "Soggetti coinvolti": squadre, docenti responsabili, quesiti, gruppi di quesiti, o *test*, e gare. Queste informazioni vengono modellate in sei tabelle:

- `bebras_teachers` contiene i dati sugli insegnanti responsabili delle squadre e, per estensione, anche i dati sugli istituti scolastici di provenienza;
- `bebras_teams` contiene i dati di ogni squadra;
- `bebras_questions` contiene i dati sui quesiti proposti nelle gare, che vengono raggruppati in gruppi di quesiti, contenuti in `bebras_tests`, grazie a `bebras_tests_questions`;
- `bebras_exams` contiene i dati sulle gare.

La maggioranza delle tabelle presenti possiede dei nomi espressivi, che alludono chiaramente all'informazione rappresentata. Lo stesso vale per i nomi degli attributi, i quali sono addirittura prefissi con una lettera che, nei casi di ambiguità, ne indica l'appartenenza ad una specifica tabella. Ad esempio, il prefisso "`e_`" indica univocamente che l'attributo appartiene alla tabella `bebras_exams`, mentre "`u_`" a `bebras_teams`. Qualche campo mantiene comunque un nome troppo generico, come i vari `e_data`, `u_data` e `q_data`, dei quali non si capisce, senza riferirsi al codice, che tipo di informazioni contengano. Risulta anche degna di nota la rete di relazioni tra le tabelle, che permette già in fase preliminare di intravedere dei legami tra alcuni tipi di dati. Quasi

[10] Per brevità e poiché, in questa situazione, uno schema di questo tipo rende molto più chiara la consultazione, non viene riportato il modello E-R della base di dati, ma solo lo schema delle tabelle e delle relazioni presenti.

tutte le relazioni sono del tipo *uno a molti*, mentre l'unica relazione *molti a molti* presente è stata reificata, creando quindi un'ulteriore tabella e le rispettive relazioni *uno a molti*. Al di fuori della rete di relazioni, e per brevità non riportate in figura, esistono anche delle tabelle che non sono legate in alcun modo ad altre:

- `bebras_edition` è una tabella completamente separata dalle altre che descrive le edizioni annuali della competizione;
- alcune tabelle che possiedono la desinenza `_archive` sono simili nella struttura a quelle principali, ma, in questo caso, si parla di tabelle archivio, in cui vengono trasportati i dati relativi alle edizioni precedenti della competizione, con lo scopo di alleggerire dai dati non strettamente necessari le tabelle dell'edizione corrente.

Nella figura mostrata sono state inoltre rimosse alcune tabelle definibili di utilità, che non sono legate strettamente al funzionamento del sistema.

Lo schema 4.3 si incentra sulla tabella `bebras_exams`, che racchiude le informazioni riguardanti le gare. Le gare vengono spesso definite *esami* all'interno del *software* a causa della sua storia, poiché in principio ci si riferiva agli esami ECDL. Ogni gara è contraddistinta da un identificativo `e_id`, che è anche la chiave primaria della tabella. Sono predisposti dei campi per salvare, per ogni gara, informazioni come il tempo impiegato, il punteggio, il tempo di inizio e fine, sia lato *client* che lato *server*, e la validità della gara stessa. Ogni gara contiene il riferimento ad altre due tabelle coinvolte: i *test*, in `e_test`, e le squadre, in `e_user`. Dallo schema si evince che ad ogni gara deve essere assegnato almeno un *test* e almeno un utente, cioè una squadra partecipante.

Dopo aver visto la loro struttura nel *database*, si può affermare che le gare ricalcano la definizione di tipo di dato fondamentale di cui si è accennato ad inizio sezione. Un discorso più ampio sulle gare è intrapreso perciò nella sezione 4.3.3.

Percorrendo la relazione di "Visualizzazione", si arriva a parlare dei *test*. I gruppi di quesiti, o *test*, sono insiemi di quesiti aggregati per edizione e per categoria, seguendo le categorie spiegate alla sezione 2.3 "Soggetti coinvolti". Ogni *test* può comprendere diversi attributi, tra i quali il tempo massimo di svolgimento oppure le azioni disponibili alla fine della competizione. Ogni gruppo di quesiti è collegato al corrispettivo insieme di quesiti, e viceversa, grazie alla relazione `bebras_tests_questions`, che è stata reificata in tabella. Per quanto riguarda i quesiti, sono contenuti nella tabella `bebras_questions`.

Sia i gruppi di quesiti che i quesiti stessi sono contrassegnati da un identificativo: in entrambi i casi, funziona da chiave primaria per la tabella.

Anche i quesiti e i *test* sono considerati un tipo di dato fondamentale. Sono trattati in maniera specifica alla sezione 4.3.1.

Percorrendo, invece, la relazione di "Partecipazione", si incontra la tabella `bebras_teams`, che contiene le squadre. Si può intravedere un'inconsistenza nella nomenclatura che il sistema prevede per le squadre, infatti talvolta ci si riferisce come *teams*, talvolta come *users*: questo perché in origine le gare erano svolte da utenti individuali, mentre oggi sono svolte da squadre. La tabella delle squadre prevede campi che contengono il nome, la parola chiave di accesso alla piattaforma di gara e lo stato delle gare in corso o di quelle già svolte. Ogni squadra è creata da un insegnante che ne è responsabile e questo viene esplicitato dalla relazione di "Responsabilità", che rende l'attributo `u_teacher` una chiave esterna riferita alla tabella `bebras_teachers`. `bebras_teachers` è la tabella relativa agli insegnanti responsabili. Ad ogni squadra è assegnato uno e un solo insegnante tutore. Per ogni docente viene predisposto un attributo composto, segnalato dal prefisso `t_school_`, che ne determina la scuola di provenienza.

Sia le squadre che gli insegnanti sono un tipo di dato fondamentale per il sistema di gara. La sezione 4.3.2 ne fa una trattazione dettagliata.

4.3.1 Quesiti e gruppi di quesiti

In questa sezione, vengono illustrate le modalità con cui il sistema di gara fornisce un metodo efficiente di inserimento dei quesiti nella piattaforma di gara, in merito a esempi così ricchi come sono i quesiti delle gare Bebras. Nel corso degli anni, il formato con cui i quesiti vengono inseriti nel sistema si è andato stabilizzando: di fatto, è stato definito un formato di inserimento che il sistema è capace di interpretare. Questo formato merita un interesse particolare, poiché cerca di bilanciare la facilità di inserimento di quesiti *standard*, con la possibilità di specificare, all'occorrenza, comportamenti completamente personalizzabili, creando così quesiti *non standard*. Nei prossimi paragrafi viene illustrato questo formato e vengono descritti i suoi punti di forza.

Riprendendo la descrizione del modello di un *task* internazionale illustrata nella sezione 2.4 "I quesiti Bebras", il formato che viene usato nel sistema di gara del Bebras italiano ricalca gli elementi visti in quel frangente. Brevemente, si ricorda che le parti di

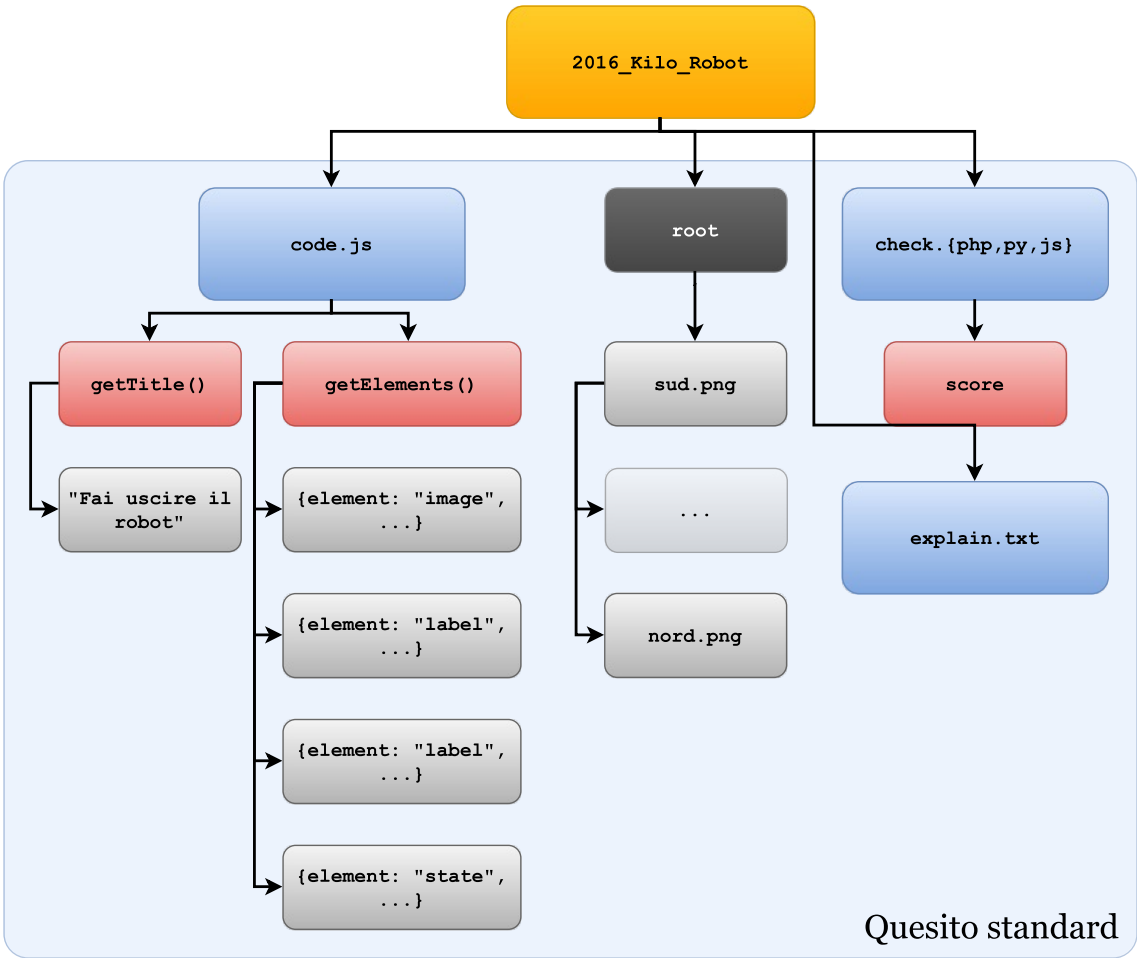

Figura 4.4: Diagramma delle risorse coinvolte nella modellazione di un quesito standard.

un *task* Bebras sono il titolo, la difficoltà, il testo del quesito, la domanda vera e propria, le risposte possibili, la correzione, la relativa spiegazione e la sezione “Anche questa è informatica”. Oltre a questi elementi, risultato di un’aggiunta specifica italiana, è presente uno *script* che valuta il punteggio ottenuto in fase di gara.

Dal punto di vista pratico, il formato elaborato per il sistema di gara italiano prevede che nel *file system*, all’interno della cartella `questions`, siano posizionate delle cartelle, contenenti dei *file*, che a loro volta presentano un formato particolare, utile per descrivere le diverse parti di un quesito. Questi *file* sono presenti in un numero variabile: mentre una buona parte deve essere necessariamente presente, alcuni sono facoltativi. Questo dipende dalla tipologia di quesito che si vuole rappresentare: se si tratta di un quesito *standard*, cioè contenente dei tipi di interazione già previsti dal sistema, sono necessarie solo le parti

obbligatorie, mentre se è uno *non standard*, è possibile aggiungere dei *file* facoltativi, atti a modellare un comportamento personalizzato.

Nei paragrafi seguenti, è presente un'analisi di entrambi i casi, con lo scopo di mostrare nello specifico com'è composto un quesito *standard* e uno *non standard*. Il risultato dell'interpretazione di un quesito all'interno del sistema di gara è quello ottenuto nella figura 2.2 a pagina 14, alla sezione 2.4 "I quesiti Bebras".

Al fine di fornire una visione di insieme di come venga rappresentato un quesito *standard* all'interno del sistema di gara, è stato scelto come esempio il quesito "Robot"[11], proprio perché sfrutta solo le interazioni già disponibili all'interno del sistema ed è un buon esempio di formato *standard*. Lo schema in figura 4.4 mostra quali sono le risorse coinvolte nella modellazione di un quesito di questo tipo.

Seguendo il diagramma, ad ogni quesito è associata una cartella all'interno del *file system*, in questo caso la cartella è `2016_Kilo_Robot`. La nomenclatura della cartella non è vincolata in alcun modo dal sistema, ma in generale si utilizza la consuetudine di inserire l'anno dell'edizione della gara in cui il quesito è comparso e la categoria di squadre per cui è stato utilizzato. Visitando questa cartella si possono trovare tutti i *file* che costituiscono le risorse relative al quesito; essi sono:

- `code.js` che contiene la descrizione dei *blocchi* visuali del quesito, di cui si parlerà più avanti;
- `explain.txt` che contiene il testo della spiegazione della risposta al quesito e la sezione "Anche questa è informatica";
- uno tra `check.php`, `check.py` o `check.js`, che sono gli *script* di valutazione della risposta data;
- infine, nella radice della cartella troviamo altri file, spesso immagini, che servono per arricchire la rappresentazione visuale del quesito.

Gli elementi riguardanti la parte grafica del quesito sono contenuti sia nel *file* `code.js`, dove si definiscono i blocchi visuali, sia nella radice della cartella, dove sono custodite le immagini e il testo. I contenuti testuali, come `explain.txt` sono semplici *file* di testo in *plaintext*, formattati tramite il linguaggio di marcatura BBCode e contenenti

[11] Il quesito "Robot" è stato tradotto e utilizzato durante le competizioni Bebras italiane del 2016, in questo caso, nella categoria *Kilo*.

i paragrafi “Spiegazione”, in cui viene spiegata la risposta esatta, e “Anche questa è informatica”, in cui viene chiarito il collegamento del quesito con l’informatica e vengono forniti ulteriori spunti di apprendimento. Il contenuto di quest’ultima parte è mostrato alle squadre come in figura 2.3 a pagina 16, nella sezione 2.4 “I quesiti Bebras”.

Le componenti grafiche della domanda presente nel quesito sono disposte sulla pagina Web facendo uso di *blocchi*, o *elementi*, che sono dei rettangoli che delimitano le aree di spazio utili per posizionare gli oggetti grafici. Le componenti grafiche possono essere sia statiche, come immagini o testo, oppure essere elementi interattivi che guidano l’utente nell’inserimento di una risposta. Un blocco interattivo si comporta in maniera simile agli oggetti che compongono un *form* di una moderna pagina Web. I blocchi sono utili non solo al posizionamento di tutte le risorse visive del *task*, ma servono soprattutto ad abilitare la condivisione delle risorse grafiche e dei modelli di interazione tra i quesiti. Siccome i *task* sono frequentemente strutturati in maniera simile, si è pensato di creare un insieme di *template* di elementi, già disponibili all’interno del codice della piattaforma, ma personalizzabili da quesito a quesito attraverso l’uso dei parametri specificati nel *file* `code.js`. Il vantaggio che se ne trae è quello di rendere molto più rapido l’inserimento delle domande di tipo *standard*. Sono presenti elementi per inserire immagini, etichette di testo formattato, domande a risposta aperta o a scelta multipla, associazioni di elementi, animazioni con cambiamento di stato oppure ordinamento di oggetti. `code.js` contiene un oggetto JavaScript che sovrascrive due funzioni: una, `getTitle()`, deve fornire il titolo del quesito, mentre l’altra, `getElements()`, deve restituire proprio la lista degli elementi che formano la domanda, parametrizzati a seconda del bisogno.

L’ultima fase della vita di un quesito è quella in cui viene valutato il punteggio ottenuto da una squadra durante la gara. Il compito di eseguire questa operazione è lasciato ad uno dei tre *script* che possono essere presenti nella cartella di un quesito: in ordine di priorità, `check.php`, `check.py` o `check.js`, a seconda che venga usato PHP, Python oppure JavaScript come linguaggio di *scripting*. Nel caso di quesiti *standard*, questi *script* sono dei brevi e semplici programmi capaci di valutare il punteggio della risposta ad un quesito. Le modalità con cui si inseriscono nel codice del sistema sono le seguenti:

- il sistema si preoccupa di trasmettere allo *script*, attraverso il flusso di ingresso `stdin`, una stringa di testo in formato JSON che contiene la serializzazione dello stato del quesito al momento della consegna delle risposte; il motivo della codifica

nel formato JSON è lo stesso spiegato alla sezione 4.2 “Tecnologie impiegate”: rimanere agnostico rispetto al particolare linguaggio di programmazione usato;

- successivamente lo *script* decide come assegnare il punteggio, lo *score*, attraverso alcuni controlli;
- una volta calcolato lo *score*, il punteggio viene stampato nel flusso di uscita `stdout`, dove il sistema è capace di leggerlo;
- successivamente il punteggio viene archiviato nel campo `e_score` della tabella `bebras_exams`.

Parlando, invece, di quesiti *non standard*, anche in questo caso è stato preso come riferimento un quesito, al fine di contestualizzare la descrizione. Si considera il quesito “PallaCastoro”[12], proprio perché è uno dei quesiti più ricchi, dal momento che utilizza le componenti facoltative, utili alla resa visiva e interattiva di domande più complesse o non *standard*. Lo schema delle risorse impiegate nel modello del quesito in oggetto è illustrato nella figura 4.5. Già apprezzandone la più vasta ramificazione rispetto a quello in figura 4.4, si può avere una prima dimostrazione di maggiore complessità.

Nei casi come questo, in cui il quesito richiede più interattività, è necessario gestire delle interazioni non previste tra i *template standard*. In queste situazioni, è necessario ricorrere ad un elemento “speciale” chiamato `custom`, da specificare sempre all’interno della funzione `getElements()` nel *file* `code.js`. Tale elemento identifica un oggetto JavaScript che si chiama come il parametro fornito e il sistema si aspetta di trovare l’implementazione di questo oggetto nel *file* `html.htm`. Per questa ragione si considera `html.htm` come un *file* opzionale. `html.htm` è, inoltre, diviso in due parti:

- nella prima parte è possibile specificare un foglio di stile nel linguaggio CSS, sia per gli elementi già presenti nel DOM che per quelli che saranno creati nella seconda parte;
- in questa seconda parte si può, invece, trovare l’implementazione dell’oggetto JavaScript specificato in `code.js`.

[12] Il quesito “PallaCastoro” è stato tradotto e utilizzato durante le competizioni Bebras italiane del 2016, in questo caso, nella categoria *Kilo*.

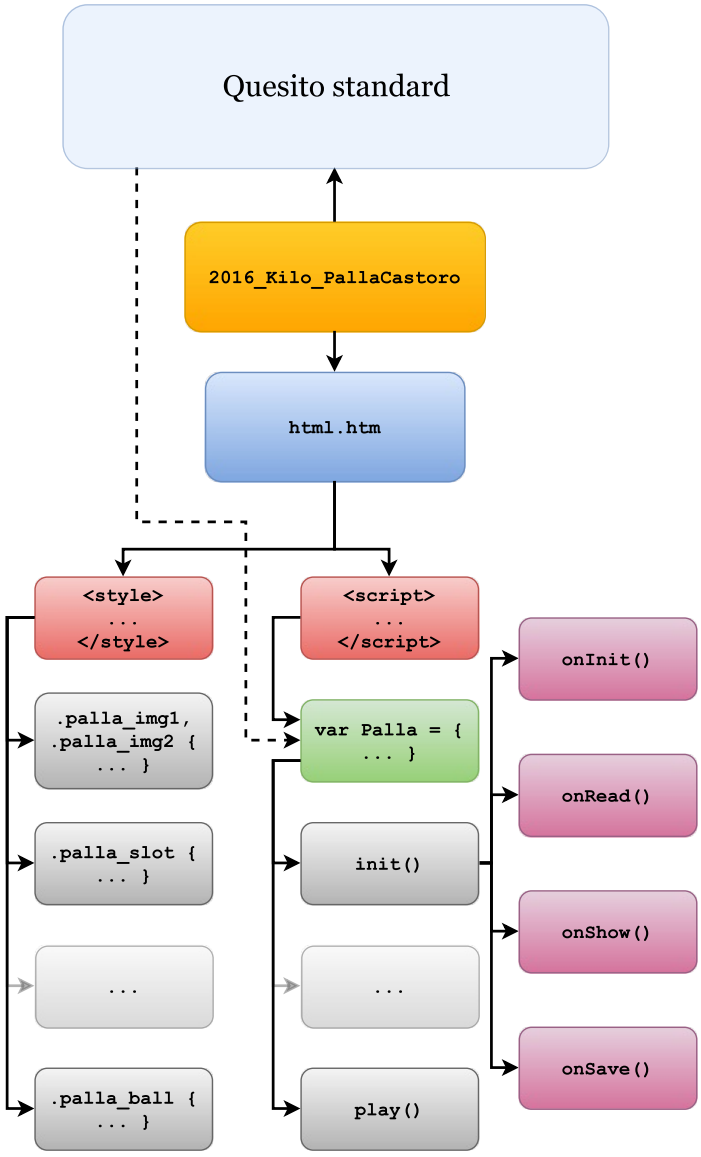

Figura 4.5: Diagramma delle risorse coinvolte nella modellazione di un quesito non standard.

L'oggetto che viene implementato definisce l'aspetto e il comportamento del nuovo elemento personalizzato. Il comportamento dell'elemento creato viene controllato ridefinendo i metodi `onInit()`, `onRead()`, `onShow()`, `onSave()`, all'interno del metodo `init()`. Questi metodi permettono all'oggetto creato di aggiungere azioni da compiere nelle diverse fasi in cui transita il quesito: in ordine cronologico, viene modificato il comportamento negli stati di inizializzazione, lettura della domanda, visualizzazione della domanda oppure salvataggio della risposta data.

Se ci si trova davanti ad un quesito *non standard*, si può riscontrare un incremento nella complessità degli *script* che si occupano di valutare la risposta data in fase di gara. Non si tratta più, quindi, di semplici controlli per capire la scelta di un utente. Tra gli *script* di correzione dei *task* che negli anni sono stati implementanti, se ne trova uno in cui è stato necessario implementare un semplice, ma completamente funzionante, interprete che interpretasse un limitato linguaggio di programmazione. Era necessario infatti valutare un'equivalenza semantica e per farlo correttamente è stato necessario realizzare uno *script* di valutazione così complesso.

A conclusione della doppia analisi, si può dire che questa particolare divisione tra quesiti *standard* e quesiti *non standard* è il modo in cui è stato affrontato il problema del bilanciamento tra il carico di lavoro per l'inserimento di quesiti semplici e la possibilità di realizzare quesiti che richiedessero un'interazione personalizzata.

Come spiegato alla sezione 4.1 "Architettura del sistema", i moduli *software* del sistema di gara non comunicano direttamente né tra di loro, né col *file system*, ma sono limitati all'uso della base di dati. Per questo motivo è stato previsto che ogni quesito debba avere quello che si può chiamare un riferimento all'interno della base di dati, nello specifico all'interno della tabella `bebras_questions`. La creazione di questo puntatore, allo stato attuale, viene effettuata direttamente dall'operatore che prepara le competizioni, agendo manualmente sulla base di dati. Nella stessa tabella vengono salvati anche dei metadati relativi al quesito, come il codice di riconoscimento, in `q_code`, il punteggio massimo ottenibile nel quesito, in `q_scoremax`, e ovviamente il nome del quesito, in `q_name`. Per come è stato pensato il meccanismo, nei casi in cui il quesito viene impiegato per più categorie di squadre o in diverse edizioni della competizione, è necessario creare molteplici puntatori nella base di dati.

Le gare Bebras sono, però, composte da un insieme di quesiti per ogni categoria di squadre e per ogni edizione annuale. È necessario un metodo che permetta di raggruppare facilmente, e anche con una certa dinamicità, i quesiti in quelli che sono stati definiti *test*. Al posto di raggruppare i quesiti direttamente a livello del *file system*, sono stati raggruppati all'interno del *database*. Questo però, come si nota dallo schema del *database* nella figura 4.3, significa introdurre una relazione *molti a molti* tra i quesiti e i *test*: un quesito può appartenere a diversi gruppi e un gruppo può contenere, e in generale contiene, molti quesiti. La relazione *molti a molti* è stata tradotta nella base di dati con un'ulteriore tabella, `bebras_tests_questions`, che associa ogni quesito ad un

gruppo e viceversa. Utilizzando quindi il codice identificativo di un *test*, contenuto del campo `e_id`, si ottengono tutti i quesiti coinvolti con una semplice operazione di *join*. Potenzialmente, questo crea un'associazione fortemente dinamica tra quesiti e gruppi, permettendo in futuro di sviluppare eventualmente delle gare personalizzate, oppure delle sessioni di allenamento mirate solo su un determinato gruppo di quesiti.

Si può notare come la tabella `bebras_tests` contenga molti attributi, questo poiché, anche in questo caso, vengono salvati i metadati relativi al *test*. Per ogni gruppo di quesiti vengono stabiliti il nome, in `t_name`, la categoria a cui è destinato, in `t_class`, l'edizione in cui viene presentato, in `t_edition`, il tempo massimo consentito per completare il *test* in gara, in `t_maxtime`, e se, al termine del test, la squadra può visualizzare la correzione, oppure ritentare lo svolgimento della batteria di quesiti. Ricordando quanto detto nella sezione 4.2 "Tecnologie impiegate", sono rimaste alcune tracce dell'evoluzione del sistema di gara: alcuni quesiti sono ancora in formato Flash ed è necessario che il sistema di gara ne sia al corrente; a questo scopo è stato creato l'attributo booleano `t_flashrequired`.

4.3.2 Scuole, docenti e squadre

Osservando lo schema della base di dati proposto in figura 4.3 a pagina 37, si può notare come docenti e squadre occupino un ruolo fondamentale nell'ecosistema Bebras. In questa sezione, si parla delle proprietà di questi due attori nel sistema di gara e di come queste siano state modellate all'interno della base di dati.

La gestione delle squadre di studenti che partecipano alle gare Bebras viene effettuata dai loro docenti responsabili e le relative modalità di accesso all'edizione annuale della gara sono già state spiegate nella sezione 2.1 "Caratteristiche della gara". Il modulo che si occupa di fornire il sito docenti, un'interfaccia alla gestione delle squadre, è *teachers*. Questo modulo ha una storia particolare: come spiegato alla sezione 2.2 "Il sistema di gara", è nato per sopperire ad un'esigenza unica e ben precisa come l'inserimento delle squadre nel sistema. A differenza di quanto avveniva nel periodo Kangourou in cui il processo era effettuato da un addetto dello *staff* operando manualmente sulla base di dati, con il Bebras le squadre vengono inserite nel sistema direttamente dagli insegnanti responsabili. In seguito, gli stessi insegnanti referenti hanno richiesto la possibilità di avere dei certificati di partecipazione nominali per ognuno dei membri di ogni squadra: la

posizione più naturale dove inserire questa funzionalità è sembrata, perciò, il sito docenti, *teachers*.

Per effettuare queste operazioni, era ovviamente necessario riservare una parte della base di dati alla memorizzazione dei profili dei docenti, contenente sia i dati utilizzati come criteri d'accesso al sito docenti, sia quelli delle squadre create, con tutte le informazioni necessarie per svolgere una gara. Seguendo lo schema della base di dati in figura 4.3, le tabelle coinvolte sono `bebras_teachers` e `bebras_teams`. Si può notare che tra le due intercorre una relazione *uno a molti* di "Responsabilità": com'è logico immaginare, un docente responsabile ha la capacità di gestire molte squadre.

È utile chiarire il criterio con il quale si concede l'accesso a `bebras_teachers` e `bebras_teams`. Il *database* non vincola in alcun modo il modulo *teachers* ad accedere esclusivamente a queste due tabelle, ma di fatto è quello che avviene a livello del codice: il modulo, durante il suo funzionamento, accede in lettura e scrittura solamente alle tabelle riservate a docenti e squadre. A conferma di questo, tramite le funzionalità di *teachers* non è assolutamente possibile operare sulle tabelle che contengono, ad esempio, dati di gara o dei *test* mostrati durante la competizione. Alla stessa maniera, qualsiasi altro modulo che abbia accesso alla base di dati sarebbe capace di alterare le informazioni presenti in `bebras_teachers` e `bebras_teams`. È il caso, ad esempio, dei moduli *backend* e *students*, che hanno bisogno di informazioni su docenti e squadre per fornire le statistiche sulle gare in corso ai supervisori o per servire le gare. La differenza con *teachers* è che *backend* e *students* fanno accesso ai dati esclusivamente in sola lettura, ma non li modificano in alcun modo, eccezion fatta per l'aggiornamento dei tempi di ultimo accesso.

Osservando i campi all'interno di `bebras_teachers` e `bebras_teams`, ce ne sono alcuni che rivelano diverse scelte implementative operate durante la modellazione della base di dati e che successivamente hanno avuto un peso sulle funzionalità del sistema.

Per quanto riguarda i dati sui docenti, si può notare l'attributo composito, caratterizzato dal prefisso `t_school_`. Nella base di dati questo si traduce come una serie di attributi del tipo `t_school_name`, `t_school_type`, `t_school_address`, ecc. Per ogni insegnante inserito nel *database*, bisogna, quindi, inserire i dati di appartenenza ad un certo complesso scolastico. I dati che si possono ricavare sulle scuole sono, perciò, confinati in ogni riga della tabella `bebras_teachers`, rendendone così difficile l'aggregazione. Se, per esempio, si volesse ricavare la lista di tutti i docenti che lavorano in una determinata

scuola, questo sarebbe complicato. Anche utilizzando un raggruppamento sul nome dell'istituto, basterebbe una difformità tra i nomi dello stesso istituto specificati in due righe differenti per falsare il risultato. Una modifica utile, potrebbe essere quella di inserire in una tabella apposita, ad esempio `bebras_schools`, contenente i dati di tutte le scuole italiane ricavati dal MIUR[13], e fornire al docente un meccanismo di scelta tra questi.

Le squadre, invece, presentano diversi attributi che è opportuno chiarire. L'attributo `u_edition` suggerisce che le squadre siano legate ad un'edizione specifica della competizione. È infatti vero che le squadre che ogni docente crea nel contesto della gara di quell'anno, alla fine della settimana di gara vengono invalidate automaticamente e non possono partecipare ad ulteriori competizioni. Per ogni squadra viene automaticamente generato un nome utente, salvato in `u_login`, e una *password*, archiviata in `u_password`, utili per l'accesso alla piattaforma di gara. Esiste anche un campo `u_data`, il cui nome andrebbe rivisto per renderlo più espressivo, ma che sostanzialmente contiene ulteriori dati sulla composizione delle squadre. Per stampare i certificati nominali, è necessario, infatti, avere a disposizione i dati anagrafici dei componenti della squadra. Questi vengono memorizzati in forma serializzata proprio nel campo `u_data`.

4.3.3 Gare

In questa sezione, viene fornita una descrizione dei moduli *software* coinvolti nello svolgimento di una sessione di gara, di come una gara viene modellata all'interno della base di dati e delle azioni che i moduli compiono sul di essa. Viene inoltre analizzato il flusso di dati che la parte *client* e la parte *server* si scambiano durante il corso di una competizione. In ultima battuta, si fa una digressione sul processo di supporto e sui meccanismi che lo regolano.

Il supporto *software* allo svolgimento delle gare è fornito dalla piattaforma di gara, implementata attraverso il modulo *students*. Si può affermare che questa sia la componente principale del sistema, poiché, oltre ad essere quella creata per prima come illustrato alla sezione 2.2 "Il sistema di gara", è anche quella più complessa e che presuppone il lavoro congiunto di tutto il sistema. Alla sezione 4.2 "Tecnologie impiegate", nello specifico nella figura 4.2 a pagina 32, si osserva uno sbilanciamento tra l'utilizzo di PHP

[13]Il MIUR è il Ministero dell'Istruzione, dell'Università e della Ricerca e fornisce, sul sito apposito (`http://dati.istruzione.it/opendata/`), i dati che riguardano tutte le scuole statali e non Italiane, provviste di codice meccanografico univoco.

e di JavaScript: prevale l'uso di JavaScript, perché è il linguaggio *client-side*. Infatti, per fornire un'interfaccia di gara, la piattaforma di gara deve occuparsi molto di più dell'interazione con l'utente, rispetto al lavoro *server-side*.

Per quanto riguarda la supervisione delle gare, il sito coinvolto è, invece, quello di monitoraggio, implementato dal modulo *backend*. La supervisione della competizione è necessaria per la riuscita della sessione di gara, a causa dei problemi tecnici che possono andare a crearsi nella durata dello svolgimento. Più avanti verranno spiegate le criticità che si sono dovute affrontare e le relative soluzioni che sono state trovate, che però spesso non forniscono una copertura totale del problema.

Anche la gestione delle gare ha bisogno del relativo spazio di memorizzazione nella base di dati. A questo scopo è dedicata la tabella `bebras_exams`, rappresentata nello schema in figura 4.3. Viene registrata ogni sessione di gara aperta, a cui viene assegnato in automatico un identificativo, posto in `e_id`. Vengono allegate alla gara anche informazioni ulteriori, come la squadra coinvolta, in `e_user`, e il gruppo di quesiti proposti, in `e_test`; entrambe forniscono una chiave esterna per la tabella `bebras_exams`.

È interessante parlare delle relazioni che intercorrono tra `bebras_exams`, `bebras_tests` e `bebras_teams`: nel primo caso si ha una relazione di "Visualizzazione", mentre nel secondo una relazione di "Partecipazione"; in entrambi casi la una relazione è di *uno a molti*. Analizzando la base di dati in isolamento, si può affermare che ogni squadra avrebbe la possibilità di svolgere più di una gara, mentre ogni *test* potrebbe avere relazioni con molte gare. Si può trovare una motivazione diversa a queste scelte, a seconda che si valuti il caso dei *test* o delle squadre. Nel caso dei *test*, è già stato confermato in precedenza che, qualora fosse necessario, un gruppo di quesiti può essere utilizzato in più d'una competizione e la relazione *uno a molti* ne è solo la conferma. Nel caso delle squadre, si è deciso in fase progettuale che una squadra avesse la facoltà di svolgere più competizioni all'interno della stessa edizione. Questo, al fine di avere un meccanismo di protezione, nell'evenienza di qualche errore che prevenga il regolare svolgimento della prima gara per quella squadra. I campi `e_valid` e `e_valid_score` nella tabella delle gare vengono impiegati a questo scopo, invalidando in un caso l'intera competizione e nell'altro solo il punteggio. Perciò, in conclusione, una squadra ha sì la potestà di poter sostenere più competizioni nella stessa edizione, ma solo previa autorizzazione e invalidamento delle gare precedenti da parte del supporto Bebras.

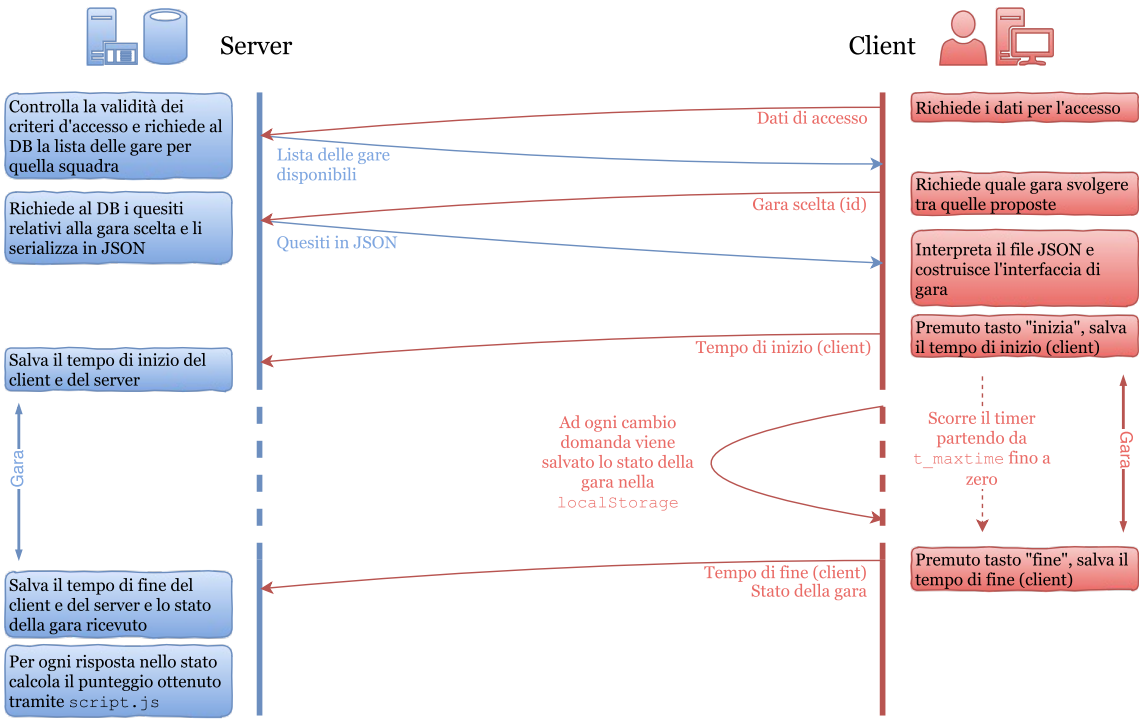

Figura 4.6: Diagramma dello scambio di dati tra la parte *server* e la parte *client* del modulo *students*.

Il modulo *students* è diviso in due parti, quella *server*, scritta nel linguaggio PHP, e quella *client*, scritta in JavaScript. Internamente al modulo *students*, i compiti sono suddivisi in questo modo: la parte *server* si occupa di comunicare con la base di dati, mentre la parte *client* si occupa di fornire un'interfaccia grafica che mostra i quesiti alle squadre, permettendogli di partecipare alla competizione. Nella figura 4.6, si trova un diagramma dello scambio di messaggi e di dati tra le due parti, in ordine cronologico. Se nella sezione 2.1 "Caratteristiche della gara" è stata fatta una descrizione della fase di gara dal punto di vista dell'utente, in questo frangente, tramite lo schema 4.6 è possibile andare nello specifico e descriverne in maniera particolareggiata lo svolgimento.

Per apprestarsi a cominciare una gara, la squadra dirige il proprio *client*, un Web *browser*, sul sito della piattaforma di gara. Viene proposta la scelta tra lo svolgimento della competizione e un'esercitazione. Nel primo caso serve essere nella settimana di gara e possedere dei criteri di accesso validi per l'edizione corrente, nel secondo si può accedere ai quesiti dell'anno precedente anche senza possedere un accesso valido. Per questa descrizione, si considera l'avvio di una gara. Scelta la gara, viene presentato alla

squadra un *form* da compilare con i dati di accesso, che una volta inseriti, vengono inviati al *server*. Questo li valuta, chiedendo al *database* se le credenziali sono valide, e in caso di esito positivo propone la lista di gare disponibili per la tipologia di squadra che ha effettuato l'accesso, valutando sia per edizione che per categoria. Per poter fornire questa lista, il *server* ha dovuto richiedere alla base di dati i *test* dell'edizione corrente tramite un'interrogazione.

In risposta, la squadra sceglie la gara che è interessata a svolgere e questo dà il via al *server* a richiedere al *database* i dati riguardanti quello specifico *test*. In questo processo, il *server* non si limita a richiedere informazioni alla base di dati, ma crea anche una nuovo *record* nella tabella `bebras_exams` con le informazioni riguardanti la squadra che ha richiesto la gara e il *test* scelto. In questo momento, il *server* conosce dati quali `t_maxtime`, che contiene il tempo massimo consentito per lo svolgimento della gara, `t_scorehidden`, `t_explainhidden`, `t_allowretry`, `t_allowreview` e `t_flashrequired`, che dicono al *client* in che modo inizializzare l'interfaccia di gara (mostrando il punteggio corrente, mostrando le spiegazioni, usando Flash o meno, ecc.). Ovviamente, tali dati vengono prontamente inviati al *client*, insieme ai quesiti relativi al *test* scelto. I quesiti sono inviati al *client*, serializzati nel formato JSON. Questo file risultante contiene tutto il necessario per visualizzare i *task*: a partire dalla descrizione dei blocchi, fino alle risorse come immagini e testo. Per evitare che le squadre possano accedere alle soluzioni del *test*, il *file* JSON non contiene la correzione dei quesiti, né la sezione "Anche questa è informatica", né tantomeno lo *script* che valuta il punteggio della risposta. A questo punto, il *client* possiede tutto ciò che serve per inizializzare la sessione di gara. Viene, perciò, mostrato un pulsante di "inizio" gara che, premuto, oltre ad avviare la competizione fa partire anche un contatore dal tempo specificato in `t_maxtime`, fino a zero, momento nel quale il tempo disponibile per completare la gara termina. Inoltre, quando la squadra preme il tasto di "inizio", il *client* si preoccupa di inviare al *server* il tempo corrente. In risposta, *server* salva anch'esso il tempo corrente. Sia il tempo di inizio ricevuto dal *client* che quello del *server* vengono archiviati negli appositi campi nella tabella `bebras_exams`. Questo viene fatto per avere un doppio riscontro sul calcolo del tempo di gara, che ovviamente deve produrre circa gli stessi risultati per entrambi i tempi. Infine, alla squadra viene mostrata l'interfaccia grafica di gara visibile nella figura 2.2 a pagina 14.

Durante lo svolgimento della gara, il *client* non sfrutta la connessione ad Internet e quindi non comunica nemmeno col *server*, ma salva i dati in locale nella *localStorage*. I

dati contenuti nella *localStorage* sono detti *stato della gara*. Questa sorta di *cache*, viene usata per evitare che qualche problema di connessione infici l'andamento della gara e la consegna dei risultati al *server*. Non potendo far affidamento sulla connessione ad Internet, e quindi su una comunicazione costante col *server*, si è optato per una *cache* locale. La mancanza di una connessione è lo stesso motivo per cui vengono salvati i tempi di inizio e di fine sia per quanto riguarda il *client* che per quanto riguarda il *server*.

Durante la gara, ogni volta che la squadra decide di cambiare il quesito visualizzato a schermo, la *localStorage* viene aggiornata con i dati del quesito appena svolto. La gara termina quando scade il contatore, e quindi finisce il tempo, oppure quando la squadra preme il tasto di "fine". In quel momento, lo stato della gara viene inviato al *server*, insieme al tempo di fine a livello *client*. Il *server* salva lo stato di gara, il tempo di fine *client* e quello *server*. Se per qualche motivo, come la momentanea assenza di una connessione ad Internet, il *client* non dovesse riuscire a comunicare lo stato di gara al *server*, i dati rimasti nella *localStorage* verrebbero inviati, senza alcun intervento da parte dell'utente, non appena la comunicazione venisse ripristinata.

Al termine della gara nella situazione in cui sia stato ricevuto tutto correttamente, il *server* deve calcolare il punteggio ottenuto dalla sqadra. Questo viene fatto in automatico per ogni quesito, attraverso l'esecuzione degli *script* presentati alla sezione 4.3.1 "Quesiti e gruppi di quesiti" (ad esempio `script.js`). Il punteggio totale è formato dalla somma di tutti i punteggi parziali risultanti dall'esito degli *script* e viene salvato nella tabella `bebras_exams` nel campo `e_score`. A seguito dell'invio del punteggio al *client*, vengono inviate anche le correzioni di ogni risposta e tutte le spiegazioni incluse al quesito. In questo modo, le squadre possono visualizzarle ed effettuare la revisione della gara.

Supporto allo svolgimento delle gare

Finora si è parlato dello svolgimento delle gare, che riguarda principalmente il modulo *students*, mentre si è solamente accennato al supporto alle gare che viene prestato dai membri dello *staff* del Bebras durante la settimana di gara.

Nell'architettura corrente del sistema rimane ancora presente un problema che può rivelarsi fatale per la buona riuscita di ogni gara. I dati sullo svolgimento della gara risiedono per un periodo, seppur breve, nella *localStorage* senza essere ancora stati inviati al *server*. Se in questo lasso di tempo, vengono persi o eliminati, non ci sarebbe più modo di

recuperare lo svolgimento della gara appena terminata, costringendo la squadra a ripetere la competizione.

Il problema rimane, tuttavia, senza una risoluzione definitiva: senza una connessione ad Internet, l'unico mezzo su cui si può fare affidamento è il *client*. Se anche il *client* presenta dei problemi nel funzionamento, non c'è più la sicurezza che lo stato di gara venga correttamente salvato, al sicuro, nella base di dati.

Un altro problema, che però fortunatamente ha una bassa incidenza statistica, è quello dato dal cattivo funzionamento degli orologi dei *client* sui quali vengono svolte le gare. Non ci si può sempre fidare del tempo *server*, poiché, come si è appena visto, la consegna dello stato di gara potrebbe avvenire in pesante differita rispetto al reale termine della gara. In quelle situazioni, si deve fare affidamento esclusivamente sul tempo *client*. Similmente al caso precedente, se non è possibile fidarsi che il *client* funzioni correttamente, il sistema non riesce a garantire, a sua volta, la coerenza dei dati temporali consegnati dal *client*. Ad esempio, nella storia delle competizioni sono stati riscontrati casi in cui il tempo di fine gara era minore del tempo di inizio.

Appare quindi evidente che, nell'impossibilità di prevenire alcuni problemi propri di questo tipo di competizioni, occorre agire nel momento in cui essi si presentano. È quindi necessaria la presenza di un gruppo di supporto all'interno dello *staff* del Bebras, che si preoccupi di monitorare le gare in fase di svolgimento, per identificare e prendere provvedimenti, riguardo a questo tipo di problemi.

Durante il tempo in cui è stata utilizzata questa piattaforma di gara nell'ambito delle competizioni Bebras, sono state isolate le principali cause delle problematiche appena descritte. Prendendo parte al supporto alle gare, i casi ricorrenti sono i seguenti:

- la gara viene terminata per errore dopo pochi minuti, senza che la squadra abbia potuto rispondere ad alcun quesito, poiché i membri del *team* hanno confuso il tasto che permette di terminare la competizione con un ipotetico tasto che permetta di terminare il quesito corrente; premendolo hanno, così, terminato la gara per errore, totalizzando un punteggio basso e un tempo di gara quasi nullo;
- allo stesso modo potrebbe capitare la medesima situazione descritta al caso precedente, ma dopo il completamento di qualche quesito; il problema verrebbe segnalato da un tempo tendente alla regolarità, ma da un punteggio decisamente basso;

- per problemi di connessione o di *client* non correttamente funzionanti, i risultati di una gara non vengono consegnati al *server*; potrebbe anche capitare nel caso la squadra non prema il pulsante di fine gara;
- la squadra avvia la stessa gara su due dispositivi diversi e per questo motivo, sul *server* risultano in corso due gare per la stessa squadra; questo è un problema, poiché lo svolgimento della competizione su due schermi separati è espressamente vietato dal regolamento;
- la squadra, per via di qualche errore, riesce ad avviare una gara che però si riferisce ai *test* dell'edizione passata;
- il *client* utilizzato dalla squadra ha qualche problema con la connessione ad Internet e non è riuscito a scaricare integralmente i quesiti, mostrando, perciò, una schermata di caricamento perenne.

Per riuscire efficientemente ad individuare questi problemi ed intervenire per correggerli, è stato creato il sito di monitoraggio, implmentato nel modulo *backend*. L'interfaccia fornita dal sito è quella che un membro del gruppo di supporto alle gare si trova davanti durante il suo lavoro ed è ritratta nell'immagine 2.1 a pagina 8, alla sezione 2.2 "Il sistema di gara". La schermata propone la lista di tutte le scuole che hanno delle squadre partecipanti all'edizione corrente; per estensione, questa è anche la lista dei relativi docenti responsabili. Supponendo che ogni squadra creata affronti una competizione, per ogni docente responsabile, sono elencate le diverse squadre che gli appartengono. Una trattazione più dettagliata delle funzionalità di *backend* si trova nel capitolo 5, alla sezione 5.1 "Il modulo *helpdesk*".

Al fine di fornire un allarme a livello *software* dei problemi citati in precedenza, sono stati creati nove stati, che il sistema cercherà di assegnare automaticamente ad ogni squadra. Ogni stato cerca di modellare uno dei problemi sopra descritti, o la mancanza di esso. Assegnare uno stato ad una gara significa popolare il campo `u_investigation`, presente nella tabella `bebras_teams` (confrontare lo schema presente nella figura 4.3). La descrizione degli stati è la seguente:

Empty

viene assegnato quando non c'è nessuna voce nella tabella `bebras_exams` con il campo `e_user` settato alla squadra in oggetto.

OK

serve per identificare una gara andata a buon fine, con un punteggio sufficientemente alto e un tempo di gara verosimile.

Running

identifica una squadra che ha cominciato ad affrontare la gara e questa si sta regolarmente svolgendo.

Low

viene assegnato alla squadra che ha terminato una gara con un punteggio inferiore ad otto punti (generalmente le gare prevedono sull'ordine della trentina di punti massimi); la gara che riceve lo stato di *Low*, nella maggioranza dei casi, è terminata dopo aver risposto a pochi quesiti e questo, spesso, si traduce nell'aver totalizzato pochi punti. Questo stato potrebbe venir assegnato anche ad una gara regolare, infatti, l'incidenza dei *falsi positivi* è piuttosto alta.

NeedCheck

viene assegnato alla squadra che ha terminato una gara con un punteggio inferiore ad otto punti e un tempo di gara minore di dieci minuti (quando la regolare durata è solitamente di quarantacinque minuti). Questo stato è simile a quello di *Low*, eccetto per il tempo di gara molto basso. Un tempo di gara minimo, oltre ad un punteggio basso, significa che la squadra non è riuscita a rispondere nemmeno a quei quesiti che avrebbero portato all'assegnamento di *Low*. In questo caso, l'incidenza dei falsi positivi è, però, minore, poiché è molto difficile che una gara così breve, sia stata terminata regolarmente.

MultiExams

nella tabella `bebras_exams` risultano più di due gare valide, ossia con `e_valid` a `true`, relazionate alla squadra in oggetto. Isolati i problemi tecnici, solitamente un *multi-esame* è dovuto all'avvio, da parte della squadra, di due gare parallele su due schermi differenti e quindi costituisce una violazione del regolamento.

RunningOut

avviene quando il tempo di fine gara, sia secondo il *server* che secondo il *client*, quindi i campi `e_server_end_date` e `e_client_end_date`, sono vuoti, ma la differenza tra l'orario corrente e il tempo di inizio a livello *server*, salvato in `e_server_start_date`, è maggiore del tempo massimo

consentito per lo svolgimento della gara, memorizzato in `t_maxtime` nella tabella `bebras_tests`, più una piccola tolleranza. Quando viene assegnato questo stato, significa che il *client* non è riuscito, per qualche ragione, a trasmettere lo stato della gara al *server*, perciò quest'ultimo non possiede né il tempo di fine, né le risposte date dalla squadra.

DateOut

segna che la gara che la squadra in oggetto ha cominciato è relativa ad un *test* destinato ad un'edizione passata della competizione, quindi il campo `t_edition` della tabella *bebras_tests* non coincide con l'anno corrente.

Loading

avviene quando il campo `e_creation_date` è settato, perciò la gara per quella squadra è stata creata, ma `e_server_start_date` non lo è, quindi il *client* non è mai stato in grado di comunicare al *server* di aver cominciato la gara.

La segnalazione che il sito di monitoraggio effettua al personale di supporto non è però sufficiente a risolvere la situazione. Il personale di supporto, una volta riscontrato il problema, provvede a contattare il docente responsabile, usando uno dei contatti specificati dal docente stesso in fase di registrazione e memorizzati all'interno della base di dati, nella tabella `bebras_teachers` nei campi `t_email` e `t_phone`. Una volta chiarita la causa del malfunzionamento con il docente, la risoluzione più frequente è che la squadra alla quale è capitato il problema proceda a rifare la gara. A questo scopo servono i campi già citati `e_valid` ed `e_valid_score`, che permettono di invalidare la gara corrente, permettendo alla squadra di effettuarne un'altra. Esiste anche il campo `u_investigation_manual`, che abilita la sovrascrittura manuale del campo `u_investigation` da parte del personale Bebras. Il campo si usa per correggere gli errori del *backend* nel caso questo sbagli a valutare la situazione, riferendo un diverso tipo di problema o un falso positivo.

Per realizzare il sito di monitoraggio è stato necessario affrontare numerose difficoltà tecniche. Bisogna mostrare una grande quantità di dati e devono essere aggiornati spesso. Sia rappresentare le informazioni in maniera efficiente, sia muoverne frequentemente una mole così ampia è un compito difficile e, attualmente, il sito risolve parzialmente il problema, ma non presenta un'interfaccia soddisfacente verso il personale di supporto. Un esempio su tutti riguarda la funzione di ricerca: considerando la presenza di un numero

di squadre nell'ordine delle decine di migliaia, l'assenza di un funzione di ricerca rende il supporto un'operazione tediosa e molto lenta. A questo scopo si è pensato di creare un sito apposito per il supporto, chiamato *helpdesk*, lasciando quello di monitoraggio solo per la visualizzazione delle statistiche e il controllo del sistema. Il sito di supporto, chiamato *helpdesk*, è presentato al capitolo 5.1 "Il modulo *helpdesk*".

Capitolo 5

Revisione e implementazione

In aggiunta all'analisi del *software* già presente all'interno del sistema di gara, nell'ambito di questo lavoro di tesi, sono state aggiunte altre componenti che mirano ad aumentare le funzionalità del sistema sotto diversi aspetti, oppure a sostituirne alcune tra quelle già presenti, al fine di migliorarne l'efficienza, l'interfaccia oppure il servizio proposto.

Una prima modifica al sistema esistente è l'aggiunta del modulo *helpdesk*. Tale modulo va a sostituire parte delle funzionalità già presenti in *backend* ed è pensato esclusivamente per coadiuvare l'attività di supporto alle gare. Punta a fornire un'interfaccia semplice, dinamica e reattiva ai membri dello *staff* Bebras che si occupano di dare aiuto tecnico durante la settimana di gara.

In questo capitolo, al modulo *helpdesk* è stata dedicata la sezione 5.1 "Il modulo *helpdesk*". All'interno di essa, alla sezione 5.1.1 "Stato dell'arte e possibili sviluppi", si trova una spiegazione delle urgenze che hanno portato allo sviluppo di questo nuovo modulo. Alla sezione 5.1.2 "Progetto e funzionalità", è presente una descrizione delle funzionalità di *helpdesk*, mentre alla sezione 5.1.3 "Implementazione e dettagli", è possibile trovare uno studio sull'architettura con cui è stato costruito il modulo e sulle tecnologie che sono state integrate nella sua implementazione.

Una seconda aggiunta al sistema è invece la creazione di *software* che dia la possibilità di utilizzare i quesiti Bebras al di fuori dei momenti di gara. Il lavoro sarà principalmente quello di definire un formato *standard* per l'esportazione dei quesiti e far sì che gli strumenti sviluppati sfruttino tale formato. Il risultato che si otterrà andrà di pari passo con l'impiego dei quesiti Bebras in ambiente didattico.

L’esportazione dei quesiti sarà trattata, all’interno di questo capitolo, nella sezione 5.2 “Utilizzo dei quesiti anche al di fuori delle gare”. Alla sezione 5.2.1 “Architettura”, si trova una descrizione dell’architettura dei *software* creati per lo scopo e delle loro funzionalità generali. Infine, alla sezione 5.2.2 “Dettagli tecnici”, l’analisi andrà più in profondità fornendo particolari tecnici che riguardano il funzionamento interno del *software* sviluppato.

5.1 Il modulo *helpdesk*

Nel corso delle edizioni delle gare Bebras che si sono alternate negli anni, lo *staff* si è reso conto che, col crescere dei numeri degli studenti e delle scuole coinvolte, l’attività di supporto allo svolgimento delle gare durante la settimana designata è diventata sempre più critica e impegnativa per la buona riuscita della competizione.

Uno strumento che affianca lo *staff* nell’attività di assistenza esiste già ed è all’interno del sito di monitoraggio. Per diverse ragioni, illustrate nella prossima sezione 5.1.1 “Stato dell’arte e possibili sviluppi”, l’interfaccia con la quale il sito di monitoraggio presenta i dati non è particolarmente funzionale allo scopo di fornire supporto tecnico.

Per questo motivo, è stato sviluppato un altro sito, chiamato sito di supporto[1], con caratteristiche e funzionalità specifiche ed esclusive per l’assistenza in tempo reale, come la semplicità di consultazione, la dinamicità dell’interfaccia e la velocità di ricerca. Queste peculiarità sono presentate nella sezione 5.1.2, mettendole a confronto, dove possibile, con quelle offerte dalla precedente interfaccia del sito di monitoraggio. Successivamente, alla sezione 5.1.3, viene effettuata un’analisi del *software* creato, sotto un profilo più tecnico: prima sarà illustrata la sua architettura in relazione a quella più generale del sistema di gara, mentre in seguito verrà fatta luce sulle tecnologie che sono state utilizzate per la sua implementazione.

5.1.1 Stato dell’arte e possibili sviluppi

Esiste già un sito, nella versione corrente del sistema di gara, che presenta un’interfaccia creata appositamente al fine di aiutare lo *staff* nel dare supporto ai docenti e alle squadre durante la settimana di gara: si tratta di sito di monitoraggio. Questo

[1] La piattaforma di supporto è visibile all’indirizzo `https://bebras.it/helpdesk`.

sito possiede sicuramente diverse funzionalità che possono essere utili allo svolgimento dell'attività di assistenza tecnica alle gare, ma, alla luce dell'esperienza maturata nelle edizioni della competizione già svolte, queste non sono risultate sufficientemente efficaci per lo scopo.

Per avere un'idea di come si presenta visivamente l'interfaccia grafica del sito di monitoraggio, si può fare riferimento all'immagine 2.1 a pagina 8, nonostante questa figura già mostri alcune migliorie applicate successivamente al momento in cui è stato preso in esame il sito. Il sito di monitoraggio contiene:

- un menu di navigazione tra le sezioni;
- una tabella con tutte le squadre registrate al sistema;
- lo stato delle gare per ogni squadra;
- una tabella che racchiude alcune statistiche sul sistema;
- dei bottoni che permettono di filtrare la lista delle gare secondo lo stato;
- una tabella che mostra i dati di tutti i docenti iscritti all'edizione corrente.

Il menu mostra varie voci che permettono di navigare tra le diverse sezioni disponibili: sono presenti soprattutto collegamenti che rimandano a funzionalità non documentate e, perciò, scarsamente utili allo *staff* di supporto. Infatti, allo scopo dello svolgimento dell'attività di assistenza alle gare, le sezioni rilevanti sono solamente quella principale, che mostra la lista delle gare, e quella che mostra la lista dei docenti iscritti al sistema.

Considerando la sezione iniziale, la parte principale mostra la tabella contenente le squadre iscritte all'edizione corrente della competizione. La lista comprende sia le squadre iscritte, ma senza gare all'attivo, sia quelle che hanno almeno una gara in corso o già svolta. Le squadre che hanno già completato la gara vengono colorate in verde, mentre quelle in attesa di avviarne una sono in bianco. Per ogni squadra, vengono mostrate anche alcune informazioni utili all'erogazione del servizio di assistenza, tra le quali il docente responsabile e i suoi dati di contatto. Al fine di rendere più veloce il monitoraggio delle competizioni che presentano dei problemi, il sistema assegna ad ogni competizione visualizzata uno stato che indica la situazione in cui si trova la gara. Gli stati e il loro funzionamento sono descritti nella parte finale della sezione 4.3.3 "Gare".

Per quanto riguarda le statistiche, la maggior parte concerne il numero di gare in un determinato stato, ma sono presenti anche informazioni di servizio, come quelle sull'indirizzo IP del *server* o sul carico corrente del processore della macchina.

Se si considera la pagina che contiene la lista dei docenti, questa è completa delle informazioni sulla scuola di provenienza e di contatto. Nella riga dedicata a un particolare insegnante, è possibile accedere direttamente alle squadre di cui è responsabile.

Per non incidere in maniera apprezzabile sul carico di lavoro della macchina adibita a *server*, che è già impegnata a garantire il corretto funzionamento della piattaforma di gara, il sito di monitoraggio impiega una *cache* per le informazioni che popolano la tabella delle squadre. La *cache* è necessaria perché il sistema deve calcolare lo stato corrente per le gare di ogni squadra e questo processo risulta particolarmente oneroso se i numeri in gioco sono nell'ordine della decina di migliaia di squadre. Non è possibile, pertanto, effettuare il calcolo ogni volta che viene servito il sito. Allo stesso modo, il processo deve comunque essere svolto periodicamente, perché lo *staff* di supporto necessita della versione più aggiornata dei dati disponibili.

La *cache*, in definitiva, funziona in questo modo: ogni richiesta di visualizzazione del sito di monitoraggio viene preceduta da un controllo mirato a valutare se è trascorso un certo quanto di tempo fissato, che identifica l'intervallo in cui i dati vengono ancora considerati freschi. Se l'esito del controllo è positivo, ossia se il tempo trascorso è superiore al lasso di tempo stabilito, i dati non sono aggiornati ed è necessario effettuare il ricalcolo degli stati. Il risultato della computazione viene memorizzato nella *cache* ed è utilizzato per servire tutte quelle richieste che rientrano nell'intervallo che va dal completamento del ricalcolo alla fine del quanto di tempo. Il ricalcolo dei dati in questo modo viene effettuato solo una volta ogni quanto di tempo e solo se un utente richiede il sito.

Parlando, invece, del processo di assistenza, è necessario precisare come viene svolta una sessione di supporto, al fine di contestualizzare meglio il *software* che è stato costruito.

La maggior parte del tempo è impiegato nel monitoraggio delle gare. Lo *staff* che si occupa del supporto controlla la tabella delle gare, cercando gare terminate in qualche stato di errore. È possibile farlo, filtrando la lista attraverso uno dei bottoni appositi oppure guardando le statistiche nella parte alta della pagina. Per trovare la riga interessata all'interno della tabella, si sfrutta la funzione "Cerca nella pagina" presente in molti *Web browser* moderni. Una volta trovata qualche gara che è terminata in maniera anomala,

si cerca di capirne la motivazione e si procede a contattare il docente responsabile della squadra. Successivamente al colloquio col docente, bisogna permettere alla squadra per la quale si è riscontrato un problema con la gara di ricominciare la competizione. Per fare questo, la soluzione più frequente è quella di annullare la gara già cominciata per permetterne l'avvio di un'altra, con un tempo disponibile commisurato all'entità del problema che è stato riscontrato. Per esempio, se è stato premuto per errore il pulsante di fine gara dopo pochi minuti, il tempo concesso sarà quello regolamentare, di solito 45 minuti; se, al contrario, i risultati della gara non sono semplicemente stati inviati al *server*, si concede solamente il tempo strettamente necessario a inserire nuovamente le risposte nel sistema.

Sulla base dell'esperienza maturata nelle precedenti edizioni della competizione, si è visto che alcune delle caratteristiche del sito di monitoraggio sono risultate funzionali al supporto ed è necessario riprodurle nella stessa maniera, oppure svilupparle ulteriormente nella stessa direzione. Per le rimanenti, invece, è, spesso, necessario modificare l'idea di fondo, affinché consentano un'assistenza migliore e più rapida. Di seguito, sono descritti i punti critici o di inefficienza contrapposti a quelli di forza. Per i primi, sarà in seguito evidenziata una strategia alternativa, mentre per i secondi, verranno mantenuti all'interno del sito di supporto.

- Il sito di monitoraggio è concepito come una *single-page application*, o SPA: questo rende molto più semplice la gestione dei dati a livello del codice e limita le richieste al *server*, con la conseguenza che l'applicazione risultante si comporta come un'applicazione nativa.

- Il menu che permette di navigare tra le diverse sezioni della pagina mostra solo due voci rilevanti per il supporto delle gare, il resto non è documentato, oppure serve allo sviluppatore del sistema per effettuare dei controlli sul sistema stesso. È necessario rimuovere le funzioni non dedicate al supporto, per semplificare l'utilizzo generale del sito.

- La tabella con i dati sulle squadre, che in genere contiene un numero di righe dell'ordine della decina di migliaia, è renderizzata interamente su una singola pagina: questo ne rende difficile la consultazione. Manca, inoltre, una funzione che permetta la ricerca globale all'interno della tabella stessa.

- Renderizzare una grande mole di dati su una pagina richiede un tempo non trascurabile. Per questo motivo, la richiesta di dati freschi può risultare lenta e non adatta ai tempi ridotti che prevede l'attività di supporto. In aggiunta, i dati non vengono ricaricati automaticamente, lasciando l'onere all'utente: così facendo, l'operatore dello *staff* rischia di lavorare su dati non aggiornati.
- Per procedere all'invalidamento di una gara allo scopo di sbloccarne l'avvio di un'altra, bisogna abbandonare la vista delle squadre, costringendo l'utente, alla fine dell'operazione, a ricaricare la tabella delle squadre, aggiungendo così altro tempo alla gestione del supporto.
- Dovendo fornire supporto a un gran numero di squadre, tenere traccia di tutti gli interventi diventa una criticità non indifferente. Anche nella versione corrente del sito di monitoraggio manca totalmente un meccanismo che permetta il tracciamento degli interventi aperti, differenziandoli da quelli ancora in attesa di essere gestiti.

Evidenziati i fattori critici, si è, quindi, sentita la necessità di ampliare il sistema aggiungendo del *software* che rispondesse esattamente a queste esigenze. Tale *software* deve contenere solamente le funzioni essenziali al supporto, essere fortemente dinamico, presentare dati i più aggiornati possibile, avere un sistema di ricerca globale delle squadre, implementare un meccanismo di tracciamento degli interventi aperti e riservare una sezione per la documentazione sulle procedure di supporto.

È tenendo presente questi bisogni che nasce il sito di supporto.

5.1.2 Progetto e funzionalità

Un estratto della pagina principale della nuova piattaforma per il supporto delle gare è visibile nell'immagine nell'immagine 5.1. Dal punto di vista dell'utente finale, appare sotto forma di sito Web al fine conformarsi alle altre, allo stesso modo della piattaforma di gara e del sito docenti.

Riferendosi all'immagine, è possibile notare che la struttura generale del sito non presenta forti differenze rispetto a quella del sito di monitoraggio; nonostante questo, ci sono sostanziali divergenze sotto il punto di vista dell'usabilità, che vengono spiegate di seguito.

In primo luogo, l'applicazione è progettata anch'essa sotto forma di *single-page application*, per sfruttare gli stessi vantaggi che sussistono per il sito di monitoraggio.

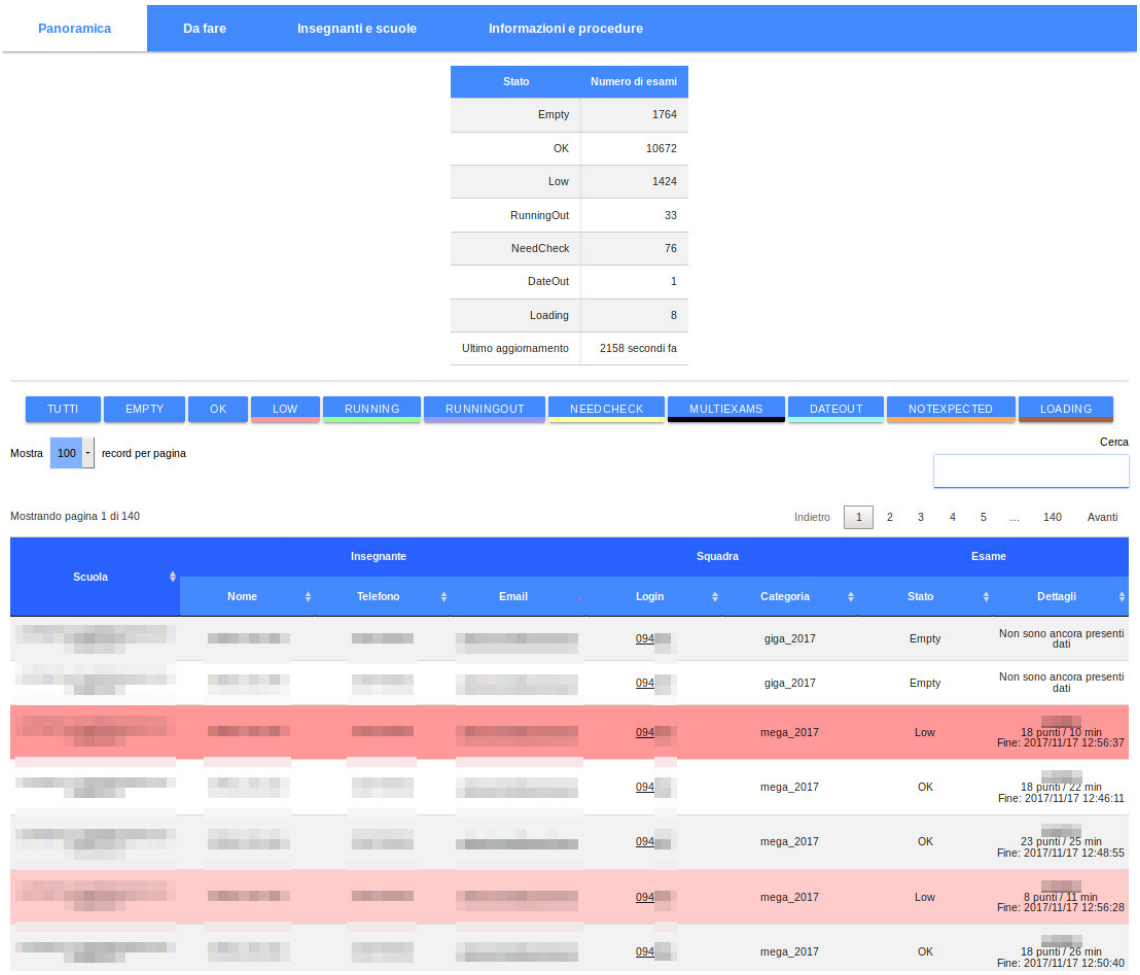

Figura 5.1: Come si presenta l'interfaccia grafica del sito *helpdesk*.

Scandendo l'immagine dall'alto al basso, viene evidenziata una maggiore rifinitura dell'interfaccia grafica, soprattutto per quanto riguarda il menu di navigazione, che in questo caso presenta esclusivamente le voci necessarie:

- "Panoramica", quella visibile direttamente nell'immagine, mostra la lista delle squadre iscritte all'edizione corrente della competizione;
- "Da fare" porta al sistema di tracciamento degli interventi aperti;
- "Insegnanti e scuole" mostra la lista di tutti gli insegnanti e le scuole;
- "Informazioni e procedure" presenta una guida sulle funzionalità diel sito di supporto e sulle procedure da impiegare nel suo utilizzo.

Per quanto riguarda la sezione “Panoramica”, la disposizione degli elementi non si discosta molto da quella visibile nel sito di monitoraggio. Quello che, sì, varia è la granularità col quale vengono mostrate le informazioni: la politica adottata in tutto il sito è quella di presentare solamente le informazioni rilevanti per il supporto alle gare.

In alto, si trova la tabella delle statistiche: in questo caso sono state rimosse tutte le informazioni che concernono il *server*, mentre sono state lasciate quelle sul numero di squadre in un certo stato, utili per avere un riassunto generale dello stato di avanzamento della competizione.

Procedendo verso il basso, sono stati posizionati i bottoni che consentono il filtraggio della tabella; premendone uno, il risultato è che la tabella delle squadre mostrerà esclusivamente le squadre che hanno all’attivo una gara nello stato scelto. Ci sono, però, due novità rispetto alla controparte nel sito di monitoraggio. In primo luogo, il filtro applicato da ogni bottone funziona al livello *server* e viene imposto direttamente nella query inviata al *database*, diminuendo il flusso generale di dati in uscita. Questo diverge dall’interfaccia precedente, dove i filtri, funzionando a livello *client*, venivano applicati dopo aver ricevuto i dati dal *server*. In secondo luogo, si è pensato di assegnare un colore ad ogni stato, di errore o meno, colore che viene utilizzato anche per colorare le righe della tabella delle squadre che presentano una gara in quello stato, rendendo molto più facile il riconoscimento visivo delle gare in errore.

Infine, nel resto della pagina è presente la componente principale, ossia la tabella delle squadre. Presenta, essenzialmente, le stesse informazioni disponibili nel sito precedente, ma le righe della tabella sono divise in pagine di lunghezza variabile, in maniera personalizzabile da ogni utente. I dati che popolano la tabella vengono aggiornati in automatico dal sistema, senza ulteriore intervento dell’utente. Premendo sull’intestazione di ogni colonna, le righe saranno ordinate alfabeticamente, sfruttando come chiave i dati contenuti nel relativo campo; questo risulta particolarmente utile, poiché consente di raggruppare le squadre secondo la loro categoria oppure lo stato della loro gara e contribuisce ad una rapida individuazione della squadra cercata. È presente, inoltre, un sistema di ricerca globale che filtra progressivamente i risultati dell’intera tabella, cercando la stringa inserita nel campo di ricerca in ogni cella di ogni riga. Ad esempio, se si inserisce nel campo di ricerca la stringa “`k`”, vengono mostrate solo le righe, in cui almeno una cella contiene il carattere “`k`”; se si continua a digitare e la stringa inserita diventa “`ki`”, vengono tolte dalla vista in tempo reale le righe che non contengono questa stringa. Inoltre, premendo sull’identificativo della gara associata alla squadra, disponibile

nell'ultima cella di ogni riga, viene aperta una finestra modale con i dati della gara e la possibilità di invalidare la competizione. La finestra modale consente di compiere quest'ultima operazione, senza abbandonare la vista delle squadre.

La sezione successiva contiene il sistema di tracciamento degli interventi in corso. Il sistema si basa sul concetto di intervento: un intervento viene aperto quando un membro dello *staff* nota qualche tipo di irregolarità oppure viene ricevuta una richiesta specifica da un docente responsabile; l'intervento può essere di carattere generale, oppure è possibile referenziarlo ad un determinato docente, ad una data squadra o ad entrambi. Di ogni intervento, vengono ovviamente mostrati i dati essenziali, come quelli dell'eventuale docente o della squadra al quale sono collegati, gli viene assegnato un titolo e ne vengono memorizzate la data e l'ora di creazione.

Nell'intero sistema è presente un meccanismo di ricerca globale del tutto simile a quello visto per la panoramica.

L'assistenza alle gare è fornita da diversi membri dello *staff* Bebras che si alternano durante la settimana di gara. Nella successione delle diverse edizioni della competizione, si è notato che, con il crescere dei numeri in gioco, diventava impossibile tenere traccia degli interventi in corso dopo i cambi di turno. Il sistema creato tiene, quindi, traccia degli interventi aperti, richiedendo all'utente di specificare uno stato per ognuno di essi. Lo stato rappresenta la fase temporale in cui l'intervento si trova. Come già fatto presente, solitamente un intervento viene aperto in seguito all'aver notato qualche irregolarità nel sistema di gara o è perché è arrivata una richiesta via posta elettronica. Nel primo caso, bisogna procedere a contattare il docente responsabile, prima per via telefonica e, nel caso in cui questa non vada a buon fine, via posta elettronica; differentemente, nel secondo caso, si procede direttamente via posta elettronica. In seguito, si rimane in attesa di una risposta da parte del docente. Per ognuna di queste azioni "Richiesta via email", "Chiamata telefonica", "Invio email" e "Attesa risposta" è stato creato un omonimo stato e gli è stato associato un colore, lo stesso riportato sui bottoni già descritti. Siccome non tutti gli interventi seguono questo iter, si è previsto anche uno stato generico. Nel sistema di tracciamento, ogni membro dello *staff* che si trova a dover gestire uno degli interventi aperti ha l'obbligo di aggiungere un commento per specificare le azioni che intraprende. Quando il problema è risolto, l'intervento relativo può essere chiuso. Chiudere un intervento equivale a portarlo in fondo alla lista, disabilitandone ulteriori modifiche, almeno finché non si decide, eventualmente, di riaprirlo.

Se l'ultima sezione presenta la documentazione utile per affrontare l'attività di supporto, la penultima contiene la tabella con l'elenco di tutti i docenti e, per estensione, le scuole. Essa è regolata dagli stessi meccanismi che regolano la panoramica. Ogni riga rappresenta un docente iscritto al sistema di gara e ogni colonna un attributo del docente. I risultati sono divisi in diverse e pagine e le colonne sono ordinabili. Anche in questo caso, è disponibile il medesimo sistema di ricerca globale. I dati di ogni insegnante sono arricchiti di un collegamento che, tramite una nuova finestra modale, riporta tutte le gare di ogni squadra che ha vede quel docente come responsabile.

Il sistema appena descritto è stato progettato e realizzato in tempo per essere utilizzato durante l'ultima edizione della competizione[2]. L'evento è stato visto come un'occasione per poter mettere sotto sforzo il sistema nell'ambiente dove sarà utilizzato regolarmente. In un primo periodo, il *software* è stato fornito in prova allo *staff* ed è, quindi, stato possibile raccogliere dei commenti e delle proposte di miglioramento da parte dello *staff* che si trovava ad usarlo, oltre che risolvere direttamente i problemi insorti durante l'uso.

Il sito di supporto è stato, pertanto, utilizzato con successo durante tutto l'arco temporale della settimana di gara. Ha riscosso una valutazione positiva dai membri dello *staff* che lo hanno impiegato durante questo periodo e, grazie al suo utilizzo, si sono ridotte le situazioni in cui era necessario ricorrere all'intervento dello sviluppatore del sistema per risolvere alcuni casi. Nonostante non sia stato effettuato alcun tipo di verifica formale e pertanto non sia possibile dimostrarne la robustezza, l'applicazione si è comportata secondo le attese quando messa sotto sforzo.

5.1.3 Implementazione e dettagli

In questa sezione, vengono descritte le scelte implementative compiute durante la realizzazione del sito di supporto. La costruzione del modulo è stata preceduta da una fase di pianificazione, nella quale sono state evidenziate le caratteristiche che questo *software* avrebbe dovuto necessariamente presentare. Per questo motivo, ad ogni scelta visibile nel codice corrisponde un processo decisionale compiuto a stretto contatto con lo *staff* Bebras, tenendo presente le necessità analizzate nel corso della sezione precedente. Al contrario di quanto è accaduto per lo sviluppo del resto dei moduli del sistema, in cui il risultato finale è frutto della risposta rapida ad una necessità incombente, con il sito di supporto ci si è sforzati di preventivare accuratamente ogni scelta implementativa.

[2] L'ultima edizione, nel momento in cui questo lavoro viene redatto, è quella 2017-2018.

Volendo effettuare un'analisi del prodotto derivato da questo lavoro, è possibile soffermarsi su due aspetti: il primo, l'architettura interna del sito e come questa si metta in relazione con le altre parti del sistema, e il secondo, le tecnologie impiegate per fornire tutte le funzionalità descritte alla sezione precedente. Il primo punto è coperto dalla sezione 5.1.3 "Architettura", mentre per il secondo bisogna riferirsi alla sezione 5.1.3 "Tecnologie".

Architettura

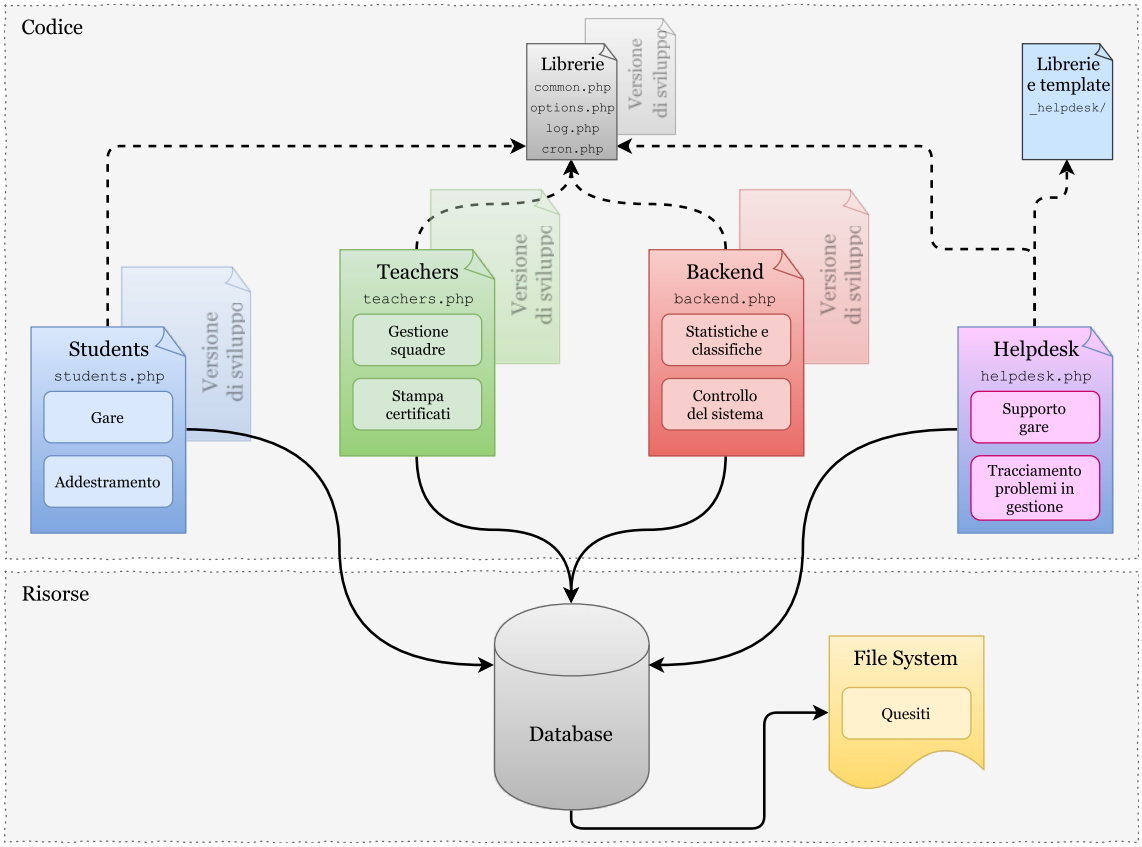

Figura 5.2: Schema dell'evoluzione dell'architettura con l'aggiunta del modulo *helpdesk*.

Per meglio descrivere l'architettura della componente creata, è utile contestualizzarla nella situazione in cui viene inserita all'interno del sistema di gara. La figura 5.2 mostra una rappresentazione schematica dell'architettura del sistema aggiornata col sito di supporto. In alcuni punti dell'analisi, sarà utile comparare questa rappresentazione con

quella presente nella figura 4.1 a pagina 28, già presentata nella sezione 4.1 “Architettura del sistema”.

Il sito di supporto è fornito, all’interno della base di codice da `helpdesk.php`, o semplicemente `heldesk`, che concorda con la definizione di modulo data nella sezione 4.1. Si può, infatti, attribuire ad *helpdesk* il pieno status di modulo, poiché è costruito seguendo gli stessi principi utilizzati per gli altri tre, *students*, *teachers* e *backend*. È totalmente indipendente dal sistema: risulta possibile, pertanto, disabilitare completamente questo modulo senza inficiare in alcun modo le prestazioni del resto del sistema. Questo fatto è supportato anche dalle caratteristiche dell’attività di supporto: trattandosi maggiormente di monitoraggio e non di modifica delle informazioni, è risultato più semplice costruire un *software* la cui assenza non avrebbe recato danno al funzionamento generale del sistema. In aggiunta, il modulo *helpdesk* è costruito anch’esso come un’applicazione Web che prevede una parte *server* ed una *client*, altra peculiarità che lo assimila al *software* già presente.

Il modulo si pone allo stesso livello degli altri anche per quanto riguarda i permessi di accesso, ad esempio, alla base di dati. Ciò si traduce nella seguente metodologia operativa: gli viene concessa la disponibilità completa della base di dati, ma di fatto accede esclusivamente alle tabelle che interessano direttamente l’attività di supporto. Addirittura, nel caso di *helpdesk*, si cerca non solo di utilizzare esclusivamente le tabelle strettamente necessarie, ma, anche nel caso di queste ultime, si limita la scrittura al minimo indispensabile. Facendo riferimento all’analisi del *database* fatta alla sezione 4.3, il modulo accede a queste tabelle:

- `bebras_teams`, la tabella che contiene i dati sulle squadre, per popolare la lista delle squadre nella pagina principale;
- `bebras_teachers`, la tabella che contiene le informazioni sui docenti, per costruire la lista dei docenti e le loro informazioni di contatto;
- `bebras_exams`, la tabella che raggruppa le gare, per avere informazioni sulle gare in corso, ma soprattutto per leggere e scrivere il loro stato.

Quindi, in definitiva, *helpdesk* non accede, nel modo più assoluto, alle tabelle che contengono informazioni sui quesiti e sui test, oppure alle tabelle archivio, semplicemente perché non sono di alcuna utilità nella fase di supporto.

All'interno della cartella che contiene il codice e le risorse del sistema di gara, accanto a quanto già descritto alla sezione 4.1, vengono aggiunti:

- `helpdesk.php`, il *file* principale che contiene il codice del sito *helpdesk* e a cui ci si riferisce come modulo;
- una cartella `_helpdesk`, che, a sua volta, contiene
  - risorse come fogli di stile e *file* con il codice da eseguire dal lato *client*, nello specifico `helpdesk.css` ed `helpdesk.js`;
  - la cartella `templates`, che contiene i modelli delle pagine Web che verranno popolati durante l'esecuzione di `helpdesk.php`;
  - la cartella `lib`, che raggruppa i *file* di libreria: `states.php`, la descrizione degli stati delle gare, `template.php`, la libreria di gestione dei `template` e `overview_compute.php` e `todo_compute.php`, che costruiscono rispettivamente la *cache* della lista delle squadre e la lista degli interventi.

Il modulo `helpdesk.php` e le librerie citate fanno uso, al bisogno, anche delle librerie già viste negli altri moduli del sistema, quali `common.php`, `options.php` e `config.php`.

Nella figura 5.2, si può notare che, a differenza degli altri tre moduli *students*, *teachers* e *backend*, per *helpdesk* manca la rispettiva versione di sviluppo. Questo perché durante la costruzione del modulo è stato attivamente utilizzato un sistema di controllo versione del codice. Ciò ha reso completamente obsoleto il metodo impiegato negli altri moduli, dei file con suffisso `_zz`, almeno sotto il punto di vista del mantenimento delle modifiche nelle diverse versioni del *software*.

Passando ad analizzare i meccanismi che regolano il funzionamento interno del modulo, ci si può riferire alla figura 5.3. La figura mostra un diagramma, fortemente semplificato, dell'evoluzione del flusso di informazioni all'interno del modulo *helpdesk*, in risposta ad ogni richiesta inoltrata da un utente.

La figura è da osservare a partire dall'utente che effettua una richiesta verso il sito *helpdesk*; tale richiesta viene, quindi, indirizzata verso il modulo `helpdesk.php`, allo scopo di essere gestita.

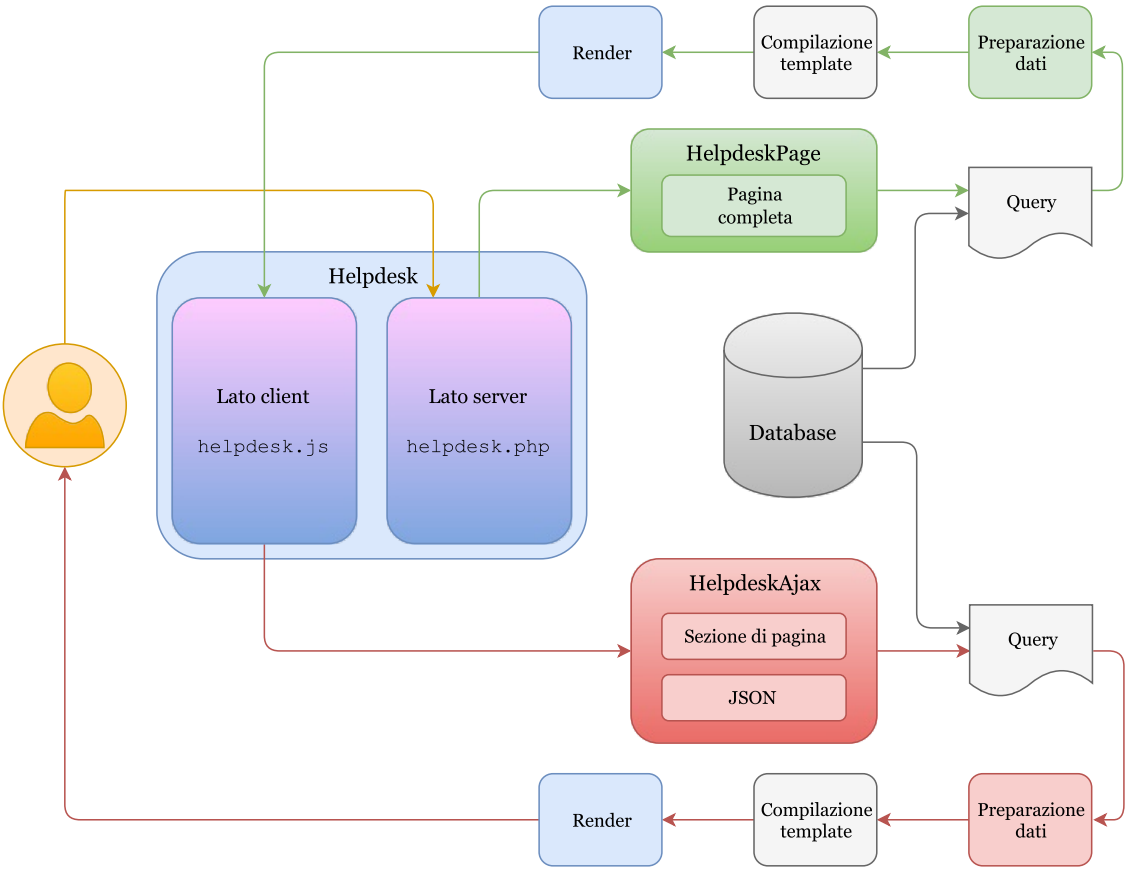

Figura 5.3: Schema del flusso di dati che viene implementato dal modulo *helpdesk*.

A questo punto, il sistema deve operare una scelta: vista la dinamicità intrinseca al processo di supporto allo svolgimento delle gare e la necessità di aggiornare costantemente i dati impiegati, si è pensato di dividere le richieste in due categorie

- richieste che danno luogo alla generazione di pagine Web complete;
- richieste che servono solamente a creare sezioni di pagina oppure serializzare dati nel formato JSON;

Il primo tipo di richieste mira ad ottenere l'ossatura della pagina, inizialmente priva di informazioni, seppur già provvista del codice che verrà eseguito lato *client*. Ed è proprio questo codice che si preoccupa di richiedere ulteriori dati al *server*, utili a popolare le

parti della pagina ancora vuote, usando delle richieste AJAX[3]. I due tipi di richieste danno luogo, nel diagramma del flusso dell'informazione, a due percorsi simili nella sostanza, ma differenziati a livello *software*, uno tracciato in verde e l'altro in rosso.

Seguendo lo schema proposto, una volta che, in `helpdesk.php`, si è capito se la richiesta dell'utente è per una pagina completa o parziale, viene seguito il rispettivo percorso, allocando l'oggetto corrispondente: `HelpdeskPage` per le richieste del primo tipo, oppure `HelpdeskAjax` per quelle del secondo.

Come già detto, l'idea alla base del processo svolto per arrivare a servire l'utente è circa il medesimo, per entrambe le tipologie di richieste. Descrivendo il flusso che scaturisce da `HelpdeskPage`, in verde nella figura, viene effettuata, per prima cosa, la connessione al *database* e, per mezzo di questa, viene eseguita la *query* verso la base di dati, per ottenere le informazioni a cui si è interessati. Successivamente, i dati ottenuti subiscono una conversione di formato, che ha lo scopo di predisporli per l'uso all'interno di un *template*. Il *template* è un modello di pagina che viene popolato con le informazioni che il *server* gli trasmette ed è letto scegliendolo tra i *file* nella cartella `templates`, citata in precedenza. Nel passo finale, questo *template* viene compilato e la pagina, una volta trasmessa alla parte *client*, viene renderizzata.

Quando il codice del lato *client* viene eseguito, questo si preoccupa di popolare le zone della pagina Web che sono rimaste ancora vuote. Vengono, quindi, lanciate diverse richieste AJAX, tese a questo scopo. Ad ognuna di esse, il *server* reagirà con l'avvio del flusso raffigurato di colore rosso in figura 5.3, relativo a una richiesta per una pagina parziale. L'iter che tale flusso segue è idealmente lo stesso delle richieste di pagine complete, descritto al paragrafo precedente.

Quando `helpdesk.php` ha esaurito i due flussi descritti, l'utente vedrà una pagina simile a quella in figura 5.1 a pagina 65.

Il comportamento appena descritto, ossia inviare all'utente prima una pagina vuota e successivamente popolarla attraverso richieste AJAX, ricalca quello tipico di una SPA. Nel caso del *software* descritto, questa modalità d'azione apporta principalmente due vantaggi:

- risulta molto più semplice implementare ulteriori pagine o sezioni di pagina; è infatti sufficiente aggiungere il nome del metodo che gestisce la richiesta desiderata nella lista dei metodi validi e provvedere alla relativa implementazione nel codice

[3] AJAX, *Asynchronous JavaScript And XML*, è una tecnica nata per lo sviluppo di applicazioni web interattive, basata sullo scambio di dati tra Web *browser* e *server*, al fine di aggiornare dinamicamente una pagina, senza renderizzarla daccapo.

dell'oggetto relativo, sia esso `HelpdeskPage` o `HelpdeskAjax`; allo stesso modo, tramite la lista dei metodi validi, diventa agevole, gestire eventuali errori legati alla richiesta di pagine o sezioni di pagina non consentite;

- se è il codice lato *client* a creare le richieste atte a popolare la pagina base inviata all'inizio, la gestione dell'aggiornamento dei dati nelle sezioni dinamiche diventa lineare: è sufficiente che lo stesso *client* lanci una nuova richiesta, simile alla precedente, ad ogni intervallo stabilito di tempo[4]; il risultato ottenuto è proprio quello di rinfrescare le sezioni dinamiche con i dati aggiornati provenienti dal *server*.

L'unica eccezione a questo comportamento è rappresentata dalla gestione della lista delle squadre. Come specificato nella sezione precedente, per questioni legate al carico del processore sulla macchina *server* e ai tempi di gestione del supporto, per questa lista viene impiegata una *cache*. L'eccezione rispetto a quanto descritto nei paragrafi precedenti, è che questa richiesta non sempre fa uso della base di dati. Riferendosi al meccanismo spiegato alla sezione 5.1.2, se la richiesta è inviata nel periodo di grazia della *cache*, non viene utilizzato il *database*, bensì i dati consegnati provengono direttamente dal *file* di *cache*.

Tecnologie

Per garantire un'implementazione semplice, ma robusta, dell'architettura del modulo, spiegata nella sezione precedente, durante lo sviluppo è stato fatto ricorso a diverse tecnologie prontamente disponibili.

In quanto il modulo è stato costruito sotto forma di applicazione Web, esiste una parte *server* e una parte *client*, riprendendo quanto detto alla sezione 4.1 "Schema generale dell'architettura del sistema e delle interazioni tra le sue componenti". Per conformarsi al resto dei moduli già presenti, il linguaggio utilizzato per implementare `helpdesk.php` è, appunto, PHP, mentre, dal lato *client*, è stata utilizzata la classica combinazione HTML5, CSS3 e JavaScript. Per gestire i *template* attraverso il linguaggio PHP, è stata impiegata un'apposita libreria chiama Twig[5]. Questa libreria stabilisce un linguaggio avanzato per i

[4] Per quanto detto al capitolo 3, non è stato possibile utilizzare tecnologie quali WebSocket o sistemi di *push* dei dati dal *server* al *client* in questo frangente.

[5] Twig è una libreria che implementa il linguaggio di modellazione Jinja2; questo fornisce un potente linguaggio per definire modelli di documenti di qualsiasi tipo. Maggiori informazioni sul sito ufficiale `https://twig.symfony.com/`.

template HTML e JSON, consentendo di creare *template* che rimangono espressivamente coincisi, pur essendo capaci di effettuare operazioni anche molto complesse.

Per quanto riguarda il *database*, ci si è adattati al *software* già in uso, dato che si è impiegata la stessa base di dati disponibile agli altri moduli del sistema; per questo motivo, nel codice, sono presenti numerose *query* in linguaggio SQL, specializzate per MySQL.

Visto lo specifico bisogno di un'interfaccia reattiva e dinamica, il focus principale delle tecnologie impiegate è fortemente sbilanciato verso il codice *client*. Come già introdotto alla sezione precedente, per gestire le sezioni dinamiche della pagina sono state utilizzate molte richieste AJAX, con una conseguente risposta nel formato JSON da parte del *server*.

Dal punto di vista dell'interfaccia grafica verso l'utente, si è scelto di usare Materialize[6], una libreria grafica che fornisce alcuni componenti base per costruire più agevolmente le pagine Web. Se si analizza l'aspetto dell'esperienza d'uso finale, la componente che maggiormente risalta è DataTables[7], una libreria che consente di creare tabelle dinamiche a partire da sorgenti di dati JSON. Grazie a quest'ultima componente *software*, è stato possibile realizzare facilmente la ricerca globale e la paginazione dei risultati, insieme all'aggiornamento automatico dei dati. Siccome la vista delle squadre è la sezione della pagina che viene consultata più spesso durante la fase di assistenza alle gare, si può affermare che questa libreria ha permesso di ridurre notevolmente il carico di lavoro dello *staff* di supporto.

5.2 Utilizzo dei quesiti anche al di fuori delle gare

Lo scopo principale del Bebras è quello di promuovere l'informatica come scienza vera e propria ed insegnare il pensiero computazionale nelle scuole, agli studenti di tutte le età. Come si è visto a partire dalla sezione 2.1 "Caratteristiche della gara", ciò viene fatto attraverso giochi informatici organizzati in molti paesi del mondo, basati su una serie di quesiti interattivi e coinvolgenti. Come evidenziato anche alla sezione 3.4 "Esportazione

[6] Materialize è una libreria composta da codice CSS e JavaScript che fornisce sia uno stile uniforme per tutti i *tag* HTML, sia un insieme coerente di *widget* grafici aggiuntivi per rendere più agevoli visualmente le pagine Web. Maggiori informazioni sul sito ufficiale `http://materializecss.com/`.

[7] DataTables è un *plugin* per la libreria jQuery che permette la creazione di tabelle HTML specificatamente pensate per l'uso da parte di un utente finale. Permette, tra le altre funzionalità di generare tabelle a partire da fonti JSON, cercare dati nella tabella, paginare i risultati oppure rendere interattiva la visualizzazione. Per avere una descrizione più ampia delle potezialità riferirsi al sito `https://datatables.net/`.

dei quesiti", viene perciò messa in evidenza la funzione formativa che i quesiti Bebras possiedono e si apre la possibilità di utilizzarli in ambito didattico.

Purtroppo, allo stato attuale, almeno per quanto riguarda il Bebras italiano, non si riesce sfruttare a pieno le potenzialità didattiche dei quesiti interattivi visti nelle gare, poiché mancano totalmente degli strumenti che supportino la possibilità di usarli al di fuori della piattaforma di gara. Questo è dovuto al fatto che il formato dei quesiti dipende fortemente dal *software* del sistema di gara, senza contare che non esiste una standardizzazione del formato utilizzato per codificarli.

Dopo che alla sezione 3.4 è stato spiegato come affrontare il problema dell'utilizzo dei quesiti al di fuori delle gare annuali, questa sezione è dedicata a spiegare quali sono gli strumenti creati, come hanno permesso di adempiere a quanto detto al capitolo 3 e come si è arrivati finalmente a definire un formato *standard* per la loro esportazione.

Per agevolare l'analisi compiuta, la sezione è divisa in due parti. Alla sezione 5.2.1 vengono introdotti gli strumenti *software* che consentono l'uso dei *task* indipendentemente dalla piattaforma di gara, evidenziando i vantaggi della loro architettura; ne viene mostrata una possibile implementazione, realizzando un'applicazione che copre un caso d'uso specifico, come *proof of concept*. I dettagli implementativi relativi alle due componenti appena presentate e alla *proof of concept* sono argomento della sezione 5.2.2.

5.2.1 Architettura

In questa sezione vengono introdotti alcuni strumenti *software* per esportare i quesiti Bebras al di fuori alla piattaforma di gara, allo scopo di utilizzarli in ambiente didattico. Si fa una panoramica generale che comprende le idee alla base della progettazione, il processo di costruzione e la descrizione delle interazioni tra le diverse parti. Alla fine della trattazione, come prova di funzionamento del *software* creato, verrà illustrata un'applicazione che integra i nuovi moduli implementati per offrire delle funzionalità che rendano la ricerca tra i quesiti più efficienti di quanto sia possibile al momento.

Nella figura 5.4 è riportata l'architettura delle due componenti sviluppate: le API[8] di esportazione dei quesiti, chiamate `export`, e una libreria capace di interpretarne una parte specifica, chiamata `bebraslib`.

[8]Le API, *Application Programming Interface*, sono un insieme di procedure, chiamate *endpoint* nello specifico, disponibili al programmatore per svolgere una determinata funzione oppure ottenere determinati dati all'interno di un programma.

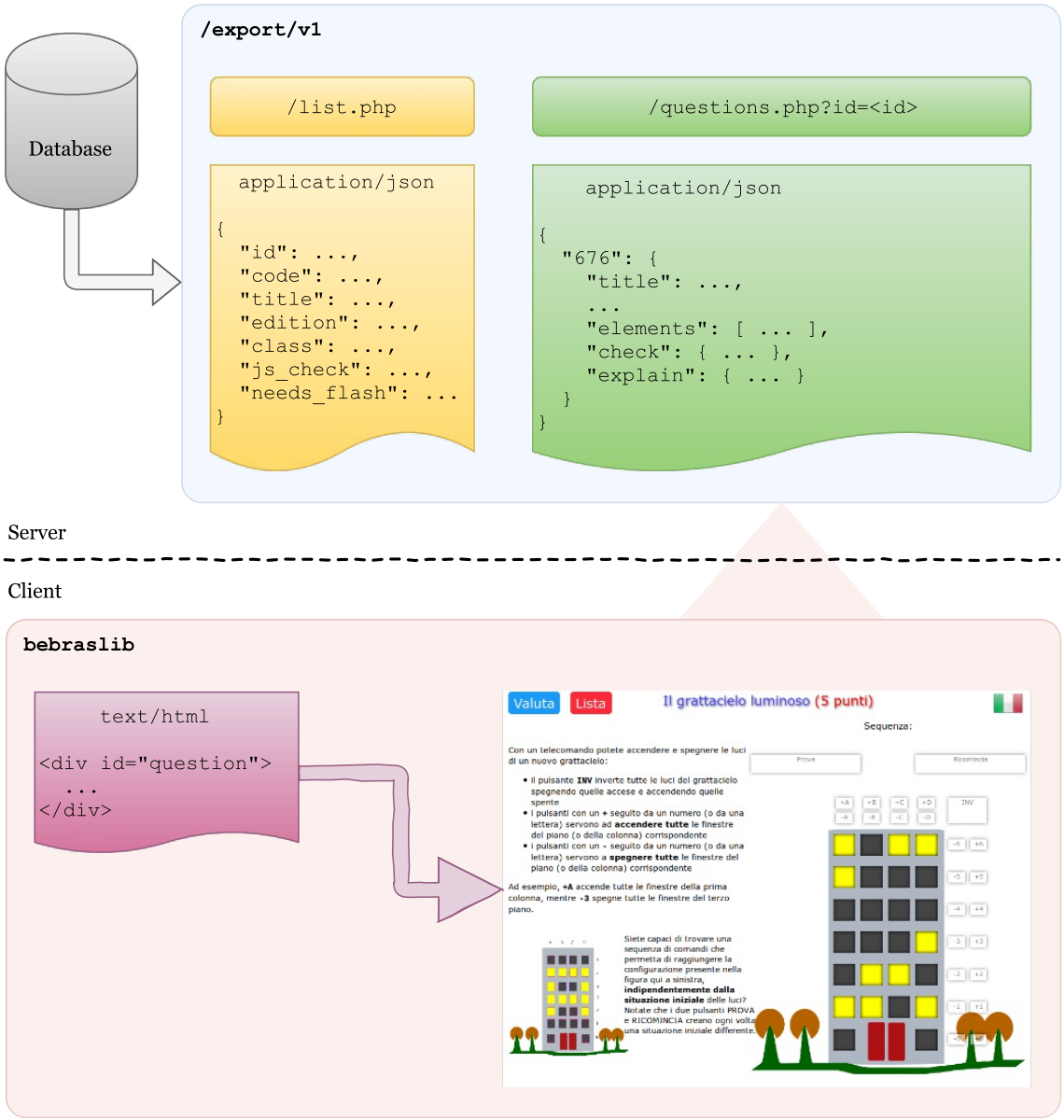

Figura 5.4: Architettura delle API `export` e della libreria `bebraslib`.

L'architettura generale del sistema di gara è quella di un'applicazione per il Web: concettualmente, è presente una parte *server* che ha l'accesso diretto alla base di dati e una parte *client* che si occupa della presentazione grafica delle informazioni. Lo stesso paradigma viene seguito dall'architettura del sistema di esportazione dei quesiti. Le API di esportazione[9], d'ora in poi definite API `export`, sono posizionate lato *server*

[9]Le risorse relative alle API sono disponibili utilizzando l'URL di base `https://bebras.it/`

e recuperano le informazioni che usano, direttamente dalla base di dati. Il *database* raffigurato nello schema 5.4 è quello generale del sistema, da cui anche i moduli *students*, *teachers*, *backend* e *helpdesk* prendono le informazioni. Nel lato *client* si trova invece `bebraslib`, una libreria che si occupa di renderizzare e mostrare all'utente il quesito, completo di tutte le parti interattive.

Le API `export` sono un insieme di risorse compatibili con l'architettura REST[10] e quindi rientrano nella definizione di API RESTful. Tipicamente REST serve per trasportare grandi moli di dati orientate all'utilizzo nelle applicazioni Web. REST espone delle *risorse* che consentono di ottenere le informazioni richieste e ogni risorsa è identificabile univocamente e in maniera uniforme per mezzo di un URL apposito. REST è basato su un'architettura *client-server* in cui ogni richiesta che intercorre tra i due è priva di stato. Dato l'ambito Web, il protocollo utilizzato è HTTP. Lo scambio di dati avviene utilizzando formati *standard* quali XML o JSON. Per la maggiore semplicità di utilizzo e per la comunque vasta compatibilità con la maggior parte dei linguaggi di programmazione più comuni, si è scelto di consentire l'esportazione dei quesiti utilizzando esclusivamente JSON.

Tralasciando i dettagli implementativi, che saranno presentati nella sezione 5.2.2, e riferendosi alla figura 5.4, sono state rese disponibli le seguenti risorse:

- `/list.php`, o semplicemente `list`, restituisce, nel formato JSON, un elenco dei quesiti disponibili, completo, per ogni quesito, di alcuni metadati particolarmente caratterizzanti; `list` è pensata per sviluppare interfacce di ricerca tra i quesiti;
- `/questions.php`, o `questions`, restituisce tutte le informazioni necessarie per la ricostruzione grafica del quesito richiesto nel parametro `id`; serve specificamente per ottenere la rappresentazione testuale, nel formato JSON, di un particolare quesito.

L'architettura REST si adatta particolarmente bene al tipo di problema che si sta affrontando, per diversi motivi:

`export/v1` e concatenando il nome della risorsa. Attualmente sono disponibili all'uso solo per lo *staff* Bebras.

[10] REST, *REpresentational State Transfer*, è un tipo di architettura software per sistemi distribuiti e consente lo scambio di dati strutturati, attraverso il protocollo HTTP. I sistemi REST hanno un'architettura *client-server* e non prevedono il concetto di sessione.

- la struttura di un quesito, o di una lista di quesiti, si adatta naturalmente ad essere una risorsa fornita dalle API `export`;
- la possibilità di utilizzare il formato JSON come formato di scambio dei dati è coerente con la premessa fatta alla sezione 3.4 “Esportazione dei quesiti”, di trarre beneficio dalla transizione a favore di questo formato avvenuta nel sistema di gara;
- l’architettura di tipo *client-server* è simile a quella già in uso negli altri moduli del sistema di gara;
- l’assenza di uno stato nella richiesta di una risorsa è un fattore che si adatta perfettamente al caso dei quesiti Bebras, i quali possono essere visualizzati a prescindere da qualsiasi stato.

Nella parte *client* della figura 5.4 viene mostrata invece la libreria `bebraslib`. Questa libreria serve per interpretare il formato *standard* definito per la rappresentazione di un quesito e per renderizzare graficamente lo stesso quesito su una pagina Web, in maniera totalmente indipendente dal sistema di gara.

`bebraslib` prende in considerazione esclusivamente la risorsa `/questions` delle API `export`, poiché il suo unico compito è, appunto, rappresentare visualmente il *task* richiesto. I principi alla base dello sviluppo della libreria sono la praticità di utilizzo e la versatilità, poiché è costruita per allontanare dall’utilizzatore la maggior parte dei compiti che renderizzare un quesito comporta: basta infatti creare un documento HTML con un contenitore a cui viene assegnato l’identificativo “`question`”, per far sì che il quesito venga renderizzato in quel contenitore, completo di ogni parte interattiva e giocabile come se fosse parte di una competizione. Una descrizione più dettagliata delle funzionalità della libreria verrà fatta alla sezione successiva 5.2.2.

Un’architettura come quella presentata in questa sezione presenta numerosi vantaggi implementativi, lasciando comunque ampio spazio ad un possibile cambiamento o rinnovamento di alcune sue parti. Le caratteristiche che hanno portato alla scelta di costruire un sistema di esportazione strutturato in questo modo sono:

- la modularità: le API `export` e `bebraslib` sono due componenti *software* totalmente indipendenti tra di loro; l’unico fattore che le accomuna è il formato *standard* definito per la rappresentazione di un quesito Bebras: in questo modo la

stringa JSON che descrive un quesito diventa una *post-condizione* che il *software* che genera le API deve rispettare nella gestione di una richiesta, mentre è una *pre-condizione* per `bebraslib`, che data la stabilità del formato, può farci affidamento;

- la separazione dei compiti: il lato *server* e quello *client* di questo sistema svolgono due mansioni completamente differenti; il primo fornisce una rappresentazione serializzata di un quesito (o di una lista di quesiti) che può servire a molti scopi, mentre il secondo usa questa rappresentazione per uno scopo molto particolare, come quello di rappresentare, in maniera visuale e interattiva, un quesito su una pagina Web;

- l'uso di una libreria per la rappresentazione grafica di un quesito: riprendendo anche quanto detto al punto precedente, `bebraslib` è solo un tipo di libreria ideabile che adempie allo specifico compito di rappresentare i quesiti su una pagina HTML, utilizzando quindi le tecnologie proprie dello sviluppo Web; è possibile costruire delle alternative che svolgano compiti differenti o presentino caratteristiche differenti:

  – potrebbe sorgere la necessità di avere una libreria del tutto simile a `bebraslib` nelle funzionalità, ma che sia implementata in un linguaggio alternativo;

  – si possono pensare altri casi d'uso che lo sviluppo di altre librerie risolverebbe; un esempio è la rappresentazione statica di un quesito, utile per la stampa cartacea;

- facilità di gestione di API RESTful: oltre ai vantaggi già illustrati in precedenza, una API RESTful consente l'aggiunta di risorse in maniera facile; per aggiungere una risorsa basta infatti scegliere un URL apposito e aggiungere del codice che la gestisca, senza essere costretti ad alterare lo stato di quello già presente;

- retrocompatibilità delle API: le API RESTful permettono un approccio facile alla compatibilità con le applicazioni che sfruttano una versione precedente delle API; basta infatti aggiungere un identificatore della versione della API all'URL di ogni risorsa: nel caso della configurazione corrente delle API `export`, l'URL è `/export/v1`, a segnalare che questa è la prima versione;

Ulteriori vantaggi di questa architettura saranno chiari una volta mostrati i dettagli implementativi delle due componenti *software*, presenti alla prossima sezione.

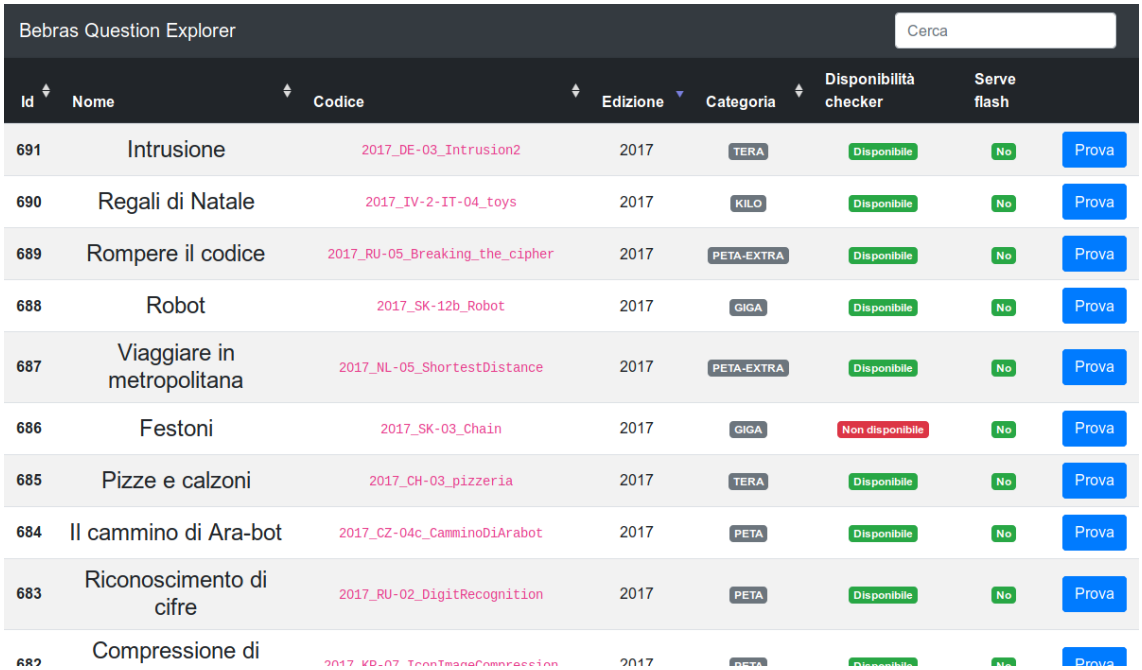

Figura 5.5: L'interfaccia di ricerca dei quesiti fornita da Bebras Question Explorer.

Per dimostrare che l'architettura e il formato scelti per il sistema di esportazione dei quesiti sono validi, efficaci e adatti allo scopo, è stata realizzata un'applicazione chiamata "Bebras Question Explorer"[11], o BQE in breve, che risponde proprio alle esigenze di ricerca illustrate alla sezione 3.4: permette di cercare efficientemente un quesito Bebras tra tutti quelli utilizzati nei diversi anni di competizioni.

Il funzionamento di questo nuovo *software* sarà dettagliato nella sezione seguente, per ora è sufficiente precisare che BQE è un'applicazione relativamente semplice sotto il punto di vista architetturale. Mostra una lista di quesiti e una volta scelto quello desiderato, permette di provarlo in maniera simile a quanto succederebbe in una competizione, di valutarlo e di leggerne la spiegazione e la sezione "Anche questa è informatica".

L'interfaccia grafica principale di BQE è presentata nella figura 5.5 e mostra la lista di tutti i quesiti Bebras disponibili sotto forma tabellare. Ogni colonna della tabella mostra un

[11] L'applicazione è visibile all'indirizzo `https://bebras.it/explorer`. Attualmente è disponibile all'uso solo per lo *staff* Bebras.

attributo specifico del quesito. La lista osservabile è la trasposizione grafica della risorsa `list` delle API `export` vista precedentemente. Dalla figura si può notare anche la presenza di un campo di ricerca: funziona allo stesso modo di quello implementato per il sito di supporto descritto nella sezione 5.1.3, cercando globalmente in tutti i campi della lista di quesiti. Gli attributi che caratterizzano ogni riga della tabella, quali il nome, il codice, la categoria e l'edizione di prima comparizione, contribuiscono a fornire validi filtri per identificare velocemente il quesito desiderato.

La parte più importante della schermata mostrata in figura 5.5 è il pulsante Prova. Questo bottone permette di accedere ad una nuova sezione dell'applicazione che avvia la renderizzazione del quesito scelto. Questa funzionalità è resa possibile integrando la libreria `bebraslib`. Il risultato è perciò simile a quello visibile in figura 5.4, nella parte *client*, e il quesito mostrato è del tutto indistinguibile da quello che si troverebbe davanti uno studente durante una sessione di gara, compresi gli elementi interattivi e la valutazione.

5.2.2 Dettagli tecnici

In questa sezione sono presentati i dettagli che riguardano l'implementazione dei tre strumenti introdotti finora: le API `export` per l'esportazione dei quesiti, la libreria `bebraslib` per la renderizzazione di un quesito su una pagina Web e Bebras Question Explorer, un applicativo che mostra l'utilizzo congiunto delle due componenti precedenti, coprendo un caso d'uso specifico.

API `export`

La prima caratteristica che si nota nelle API di esportazione è la relativa semplicità strutturale, dato che, nella configurazione corrente, si tratta solamente di due risorse, `list` e `questions`. Per implementarle si è deciso di procedere in questo modo: sia `list` che `questions` sono, lato *server*, due *file* PHP totalmente differenti, `list.php` e `questions.php`. Allo stesso modo, il percorso che si desidera far apparire nell'URL della risorsa, ad esempio `/export/v1`, è costituito da una serie di cartelle, una dentro l'altra; in questo caso, la cartella `v1` è all'interno della cartella `export` e contiene i due file `list.php` e `questions.php`. La situazione è esemplificata nel listato 5.1.

Ogni *script* PHP si occupa quindi di gestire una singola risorsa. L'algoritmo che viene seguito per recuperare le informazioni dalla base di dati e inviarle all'utente è sempre lo

```
export/
  v1/
    list.php
    questions.php
```

Listato 5.1: Strutturazione degli URL delle risorse.

stesso per ogni risorsa, fatta eccezione per qualche piccola differenza trascurabile. I passi sono:

- collegamento al *database*;
- lettura dei valori forniti dall'utente come parametri dell'URL;
- utilizzo dei parametri letti per comporre la *query* verso la base di dati ed esecuzione della *query*;
- filtraggio e manipolazione dei risultati ottenuti;
- invio della forma finale dei risultati all'utente, in formato JSON.

La strutturazione mostrata finora apporta dei vantaggi operativi. Innanzitutto è molto semplice alterare all'occorrenza l'URL di una risorsa, senza dover necessariamente modificare il codice: è sufficiente spostare gli *script* che gestiscono le risorse in una nuova cartella. Questa tecnica è stata utilizzata per dotare di una versione il sistema delle API: lo spazio per una nuova versione delle API si crea predisponendo una nuova cartella, per esempio `v2`, allo stesso livello di `v1` e copiando i programmi PHP che gestiscono le nuove risorse in quella cartella. Anche implementare una nuova risorsa è molto semplice con questa struttura dei *file*: si crea un nuovo *script* PHP che abbia il nome della risorsa che si vuole aggiungere e, all'interno di esso, si scrive il codice che la gestisce. È utile precisare che, sebbene l'algoritmo seguito dalle risorse esistenti sia di fatto quello appena illustrato, questo non è obbligatorio: è sufficiente che lo *script*, alla fine della sua esecuzione, fornisca i dati nel formato JSON, ma non viene messo alcun vincolo per quanto riguarda la sorgente delle informazioni o la loro manipolazione.

Parlando delle risorse già implementate e disponibili all'utente, si può fare riferimento al listato 5.2, che mostra un esempio del risultato restituito da `list`, e al listato 5.3, che mostra quello restituito da `questions`.

```
{
  // identificativo del quesito
  "id": 676,
  // nome della cartella in cui risiede il quesito
  "code": "2017_IT-06_Skyscraper",
  // nome del quesito
  "title": "Il grattacielo luminoso",
  // edizione in cui è stato usato il quesito
  "edition": 2017,
  // categoria del quesito
  "class": "peta",
  // se esiste o meno un checker in JavaScript
  "js_check": true,
  // se il quesito necessita flash
  "needs_flash": false
}
```

Listato 5.2: Esempio di risultato fornito da `list`.

Per quanto riguarda `list`, si possono fare delle considerazioni sui dati che questa risorsa produce. La risorsa fornisce una lista di tutti i quesiti disponibili, sotto forma di oggetti JSON, di cui quello rappresentato nel listato è un esempio. La sequenza restituita è classificata per anno e per categoria. Come si può osservare, vengono fornite delle informazioni di base riguardanti il quesito: un numero identificativo univoco, il titolo, il nome, l'edizione e la categoria della competizione in cui è comparso la prima volta. Riprendendo quanto già detto in precedenza, questa risorsa è orientata all'uso per interfacce di ricerca tra i quesiti; Le chiavi `js_check` e `needs_flash` servono quindi per filtrare i risultati e, ad esempio, mostrare solo quelli che non necessitano della tecnologia Adobe Flash e hanno uno *script* di valutazione della risposta scritto in JavaScript. Questa possibilità tornerà utile nel caso di `bebraslib`, che utilizzando tecnologie Web moderne non può usare Flash ed è limitata all'impiego di JavaScript.

Attualmente non sono disponibili ulteriori possibilità di filtraggio tramite i parametri inviati nell'URL di richiesta. In futuro si potranno aggiungere delle parti di codice che gestiscano questa situazione, consentendo di filtrare i risultati desiderati direttamente lato *server* e non esclusivamente lato *client* come avviene ora.

Parlando di `questions`, la situazione risulta un po' più complessa. La risorsa legge una lista di identificativi numerici dall'URL, per esempio `/export/v1/questions?id=676,675,674`. Il meccanismo è pensato in questa maniera: l'utente prima ottiene gli identificativi desiderati tramite `list` e successivamente li richiede a `questions`. `questions` restituisce tutte le informazioni dei quesiti contraddistinti dagli `id` specificati. Per comodità la risorsa accetta anche un identificativo singolo come parametro

```
{
  // id del primo quesito richiesto
  "676": {
    // nome del quesito
    "title": "Il grattacielo luminoso",
    // paese che ha ideato il quesito
    "flag": "it",
    // nome della cartella in cui risiede il quesito
    "id": "2017_IT-06_Skyscraper",
    // categoria del quesito
    "class": "peta",
    // cartella radice dove sono posizionate le risorse nel server
    "root": "\/platform",
    // edizione in cui è stato usato il quesito
    "edition": 2017,
    // massimo punteggio totalizzabile
    "scoreMax": 5
    // blocchi grafici del quesito
    "elements": [
      // blocco 1, label
      {
        "element": "label",
        ...
      },
      // altri blocchi
      { /* ... */ },
      // blocco custom (opzionale)
      {
        "element": "custom",
        "class_name": "Skyscraper",
        // il codice del blocco custom ofuscato in base64
        "code": "PHN0eWxlIHR5cGU9InRleHQv...",
        ...
      }
    ],
    // checker disponibile (viene inviato anche se non è in JavaScript)
    "check": {
      // linguaggio in cui è scritto
      "language": "javascript",
      // il codice del checker ofuscato in base64
      "body": "Cgp2YXIgc2NvcmVNYXggPSA..."
    },
    // spiegazione quesito / "Anche questa è informatica"
    "explain": {
      // corpo dell'explain
      "body": "...",
      // risposte corrette
      "answers": {
        // elenco delle risposte possibili
        ...
      }
    }
  },
  // id del secondo quesito richiesto
  "677": {
    // dati del secondo quesito
    ...
  }
}
```

Listato 5.3: Esempio di risultato fornito da `questions`.

e infatti è questa la forma più utilizzata, essendo sfruttata anche da `bebraslib`. Un esempio del JSON restituito da `question` si può osservare nel listato 5.3. Il formato visibile è quello che precedentemente è stato chiamato formato *standard* per l'esportazione dei quesiti e viene ricavato direttamente da quello di interscambio dei dati, interno al sistema di gara. In aggiunta alle informazioni che vengono prodotte anche da `list`, si trova:

- la lista dei blocchi che contengono gli elementi interattivi del quesito;
- lo script di controllo della risposta, anche chiamato *checker*, unitamente al punteggio massimo ottenibile, in `scoreMax`;
- la informazioni di correzione e spiegazione della risposta, insieme alla sezione "Anche questa è informatica";
- la lista delle risposte esatte.

La lista dei blocchi ricalca la configurazione descritta nella sezione 4.3.1 al capitolo 4. Ciò è intenzionale, poiché in questo modo è più facile per le librerie che implementano questo formato garantire il funzionamento dei quesiti in maniera identica a quanto accade nella piattaforma di gara. Oltretutto avere lo stesso formato dei blocchi, permette di riutilizzare gran parte del codice della piattaforma di gara che lo interpreta, come si vedrà in `bebraslib`.

Lo *script* di valutazione della risposta viene inviato anche nel caso il relativo codice di programmazione non sia scritto in JavaScript. Parlando di codice, in questo formato sia il codice relativo ai blocchi *custom* che quello relativo al *checker*, prima di essere inglobato nel JSON, viene codificato in Base64[12], per essere correttamente serializzato.

Sebbene il formato *standard* sia già completamente utilizzabile nella versione corrente del *software* di gara, rimangono ancora delle possibilità di miglioramento. Per esempio non esiste un modo globale di impedire che un quesito venga esportato, oppure una maniera per definire la difficoltà di un quesito indipendentemente dalle categorie della competizione. Queste proposte di avanzamento del formato *standard* potranno essere affrontate in una versione successiva del sistema di gara.

[12] Base64 è una codifica *standard* orientata alla conversione di dati binari in testo e di solito si usa, appunto, per inglobare un flusso binario in un file di testo; si tratta di una codifica totalmente reversibile, efficiente e si chiama Base64 perché usa un alfabeto di 64 simboli.

`bebraslib`

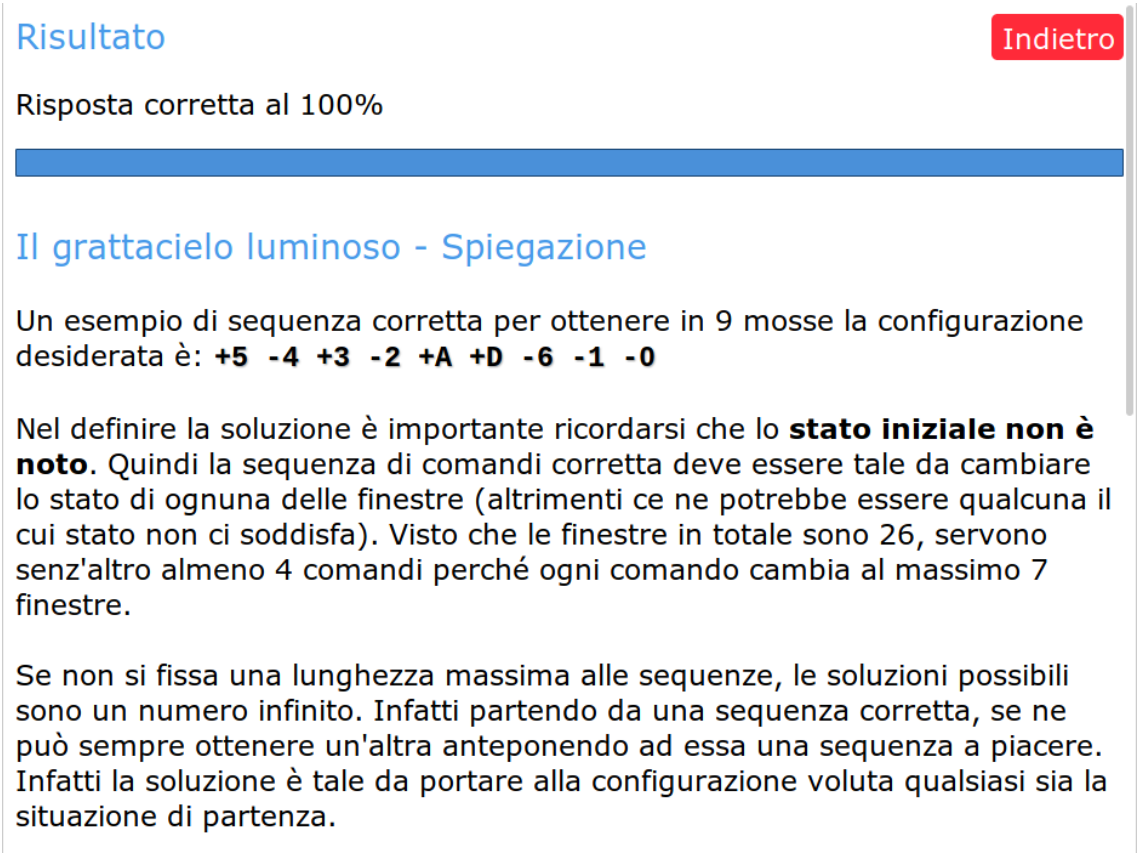

Figura 5.6: Come `bebraslib` mostra la valutazione del quesito svolto.

La libreria `bebraslib` sfrutta il formato *standard* restituito dalla risorsa `questions`, compresa nelle API `export`, per renderizzare un quesito Bebras su una pagina Web.

Come è già stato detto nei paragrafi precedenti, `bebraslib` cerca di risparmiare all'utente finale la maggior parte delle azioni necessarie per utilizzare un quesito Bebras. Sostanzialmente, le azioni da intraprendere sono tre. In *primis*, bisogna creare una pagina HTML5 standard, completa di dichiarazione `!doctype`, e inserirci un contenitore `<div>` che abbia l'attributo `id` impostato a `question`. Successivamente è necessario includere lo *script* JavaScript che contiene l'implementazione di `bebraslib` e inizializzare la libreria; in questo passaggio si passa al costruttore, sotto forma di stringa, il JSON restituito dalla risorsa `/questions.php`. Il quesito sarà infine renderizzato nel contenitore specificato, completo di tutte le parti interattive, e sarà simile a quello

osservabile nella parte *client* della figura 5.4. Ovviamente sarà presente anche la parte di valutazione, mostrata come nella figura 5.6.

Senza scendere in eccessivi dettagli, per arrivare alla renderizzazione di un quesito, `bebraslib` si presenta ad esso come se fosse la piattaforma di gara, seguendo un processo simile all'emulazione. Il codice che gestisce questa funzionalità è infatti in gran parte importato da quello della piattaforma di gara, rivisto, corretto e adattato a questo specifico caso d'uso, per garantire la massima compatibilità. A livello grafico, gli unici cambiamenti presenti, rispetto alla versione dello stesso quesito visibile nella piattaforma di gara, sono i pulsanti per la valutazione della risposta e le modalità di visualizzazione della spiegazione. Per rendere `bebraslib` quanto più versatile possibile, sia i pulsanti, sia le informazioni relative alla valutazione vengono mostrati nello stesso contenitore utilizzato per visualizzare il quesito, quello con identificativo `question`.

La libreria implementata utilizza tutte le tecnologie per lo sviluppo Web moderno, nella fattispecie HTML5, CSS3 e JavaScript. È progettata per fornire le proprie funzionalità esclusivamente lato *client*, senza richiedere alcun intervento esterno da parte del *server* del sistema di gara, eccezion fatta per le richieste alla API e le risorse grafiche come le immagini. Per questo motivo, la parte interattiva di un quesito è disponibile solo se non usa Adobe Flash e la valutazione solo se presenta un *checker* scritto in JavaScript.

Bebras Question Explorer

Bebras Question Explorer è la naturale prosecuzione di quanto detto finora. Sono state create le API `export` per produrre la lista dei quesiti disponili e per l'esportazione di un singolo quesito in un formato *standard* ed esiste una libreria capace di interpretare questo formato e di renderizzare il quesito corrispondente su una pagina Web. Manca uno strumento capace di interpretare `list`, la lista dei quesiti disponibili, permettendo di sceglierne uno, di estrarne l'identificativo numerico e di richiederlo in automatico a `questions`, affinché `bebraslib` possa usare il JSON restituito: BQE copre esattamente questo caso d'uso.

BQE è un'applicazione scritta in Node.js[13]. Il funzionamento della pagina principale di BQE, visibile nella figura 5.5, è già stato descritto alla sezione precedente; si può aggiungere che il compito del pulsante Prova è quello di portare l'utente su una pagina

[13]Node.js è un'interprete JavaScript costruito sul motore JavaScript del noto Web *browser* Google Chrome. È una tecnologia interessante poiché utilizza un modello ad eventi con flussi in entrata o uscita non bloccanti e quindi è orientato alle prestazioni. Maggiori informazioni sul sito ufficiale (`https://nodejs.org/`).

dove viene inizializzata `bebraslib` per renderizzare il quesito scelto. Da quel momento, l'utente è libero di interagire con il quesito come desidera. È possibile saltare l'interazione con la lista dei quesiti ed accedere direttamente al quesito utilizzando l'URL che viene fornito cliccando sul bottone Prova, ad esempio `/explorer/question/676`.

Si possono individuare altre caratteristiche di BQE degne di nota:

- l'applicazione è stata sviluppata cercando di renderla utilizzabile da qualsiasi dispositivo; può perciò essere usata senza perdita di funzionalità anche sui dispositivi mobili;
- se all'URL che porta direttamente alla prova del quesito, ad esempio `/explorer/question/676`, si sostituisce `/explorer/embed/676`, se ne ottiene una versione integrabile in una pagina Web attraverso un `iframe`; la funzionalità è particolarmente utile se si impiega il quesito in ambito didattico, poiché permette di raggruppare una serie di quesiti all'interno della stessa pagina Web con minimo sforzo.

Capitolo 6

Conclusione e sviluppi futuri

In questa tesi, abbiamo analizzato il sistema di gara utilizzato per supportare lo svolgimento delle gare Bebras, con l'obiettivo di valutare se potesse essere migliorato ed esteso con nuove funzionalità e con quali approcci.

Dopo aver presentato le caratteristiche della competizione Bebras in Italia, si sono evidenziati gli aspetti cruciali relativi al sistema di gara: la competizione vede la partecipazione simultanea di molte squadre che gareggiano da sedi scolastiche diverse, attrezzate in maniera non uniforme, e ad ogni squadra deve essere garantita la possibilità di accedere e visualizzare i quesiti nello stesso modo; il contesto di utilizzo nelle scuole presenta particolari criticità (connessione non affidabile, *client* obsoleti, supporto di dispositivi mobili); il *software* è stato sviluppato senza una progettazione organica, non dispone di documentazione, di sistemi d'integrazione continua o test di unità, né dello storico delle versioni.

A fronte di quest'analisi è stata prodotta una descrizione generale dell'architettura complessiva, con approfondimenti sulle varie componenti, le tecnologie utilizzate, i dettagli implementativi in particolare relativi all'organizzazione dei dati. Il sistema è apparso molto coeso internamente, ma poco adattabile ad una qualunque estensione, benché svolgesse i compiti previsti adottando soluzioni *ad hoc* che si sono dimostrate affidabili nel corso delle varie edizioni.

Siamo intervenuti sul sistema di gara seguendo due approcci. Da un lato è stato realizzato un sito di supporto migliorando e consolidando alcune funzionalità preesistenti, mantenendo l'impostazione e lo stile di sviluppo utilizzato dal creatore, cercando di mitigare quanto più possibile i difetti legati al processo di sviluppo e alla scarsa separazione degli ambiti all'interno del codice. Dall'altro lato si è individuato un formato

per l’esportazione dei quesiti, si è definita un’interfaccia e si sono sviluppati degli strumenti esterni basati su di essa, per l’esportazione e visualizzazione dei quesiti al di fuori del sistema di gara.

Quest’ultimo lavoro rende possibile la progettazione e realizzazione futura di nuove applicazioni centrate sui quesiti, tra le quali: nuove librerie che consentano la formattazione di un quesito per la stampa; la ricerca di quesiti basata sulla base di caratteristiche diverse, ad esempio il livello di difficoltà; lo sviluppo di una piattaforma per gli insegnanti in cui i quesiti sono presentati in relazione alle indicazioni curricolari e agli obiettivi formativi.

Inoltre, l’approccio può essere applicato anche agli altri tipi di dato immagazzinati nel sistema, come i dati relativi alle gare o alle risposte date ai quesiti. Definendo delle opportune interfacce si potrebbero realizzare sistemi di gara completamente personalizzati e rendere i dati fruibili ai ricercatori che si occupano di didattica dell’informatica.

Infine, l’analisi ha evidenziato l’opportunità di alcune modifiche sulla base di dati, ad esempio per quanto riguarda la difficoltà stimata di un quesito, perché al momento il dato non è esplicitamente indicato e quindi non può essere messo a disposizione per mezzo dell’interfaccia.

Bibliografia

[1] D. L. Parnas, “On the criteria to be used in decomposing systems into modules,” *Commun. ACM*, vol. 15, pp. 1053–1058, Dec. 1972.

[2] Bebras Organization, “Bebras.” `http://bebras.org/`, 2018. [Online; in data 26-February-2018].

[3] Bebras Italia, “Bebras dell’Informatica.” `https://bebras.it/`, 2018. [Online; in data 26-February-2018].

[4] ALaDDIn, “Aladdin - laboratorio di divulgazione e didattica dell’informatica.” `http://aladdin.unimi.it/`, 2018. [Online; in data 26-February-2018].

[5] C. Bellettini, F. Carimati, V. Lonati, R. Macoratti, D. Malchiodi, M. Monga, and A. Morpurgo, “A platform for the Italian Bebras,” in *Proceedings of the 10th international conference on computer supported education (CSEDU 2018) — Volume 1*, pp. 350–357, SCITEPRESS, 2018.