Esame di Ingegneria del Software – 10 CFU – 16 giugno 2023

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- **L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.**
- Tempo a disposizione: 3 ore.

Si vuole realizzare un software per supportare la gestione di Società Polisportive. Ogni società è caratterizzata da un nome, un indirizzo, un numero di telefono ed un proprietario, caratterizzato dalle proprie generalità (nome, cognome, data di nascita, codice fiscale, pec). Le società possono disporre di campi per diversi sport: calcio, tennis, basket, pallavolo, curling, etc. A loro volta i campi possono distinguersi in base al numero di giocatori per squadra e in base al tipo di superficie di gioco. Ad esempio, i campi da calcio possono essere fatti per squadre da 5, 8 o 11 giocatori, ed essere realizzati con superficie in erba sintetica, erba naturale, oppure in *gripper*. Il proprietario può inserire nel sistema nuovi campi sportivi, identificati da nome, sport, tipologia, e prezzo orario. Gli utenti del sistema possono registrarsi specificando nome utente, password, indirizzo e-mail e numero di carta di credito. Gli utenti registrati possono quindi prenotare l'utilizzo di un campo. In particolare, gli utenti registrati possono visualizzare i campi disponibili specificando lo sport desiderato e una data, e selezionando uno slot orario tra quelli disponibili. Le prenotazioni hanno tutte durata di un'ora. Prima di salvare la prenotazione, il sistema mostra all'utente una schermata di riepilogo con data, orari, campo selezionato, e prezzo per la prenotazione, e chiede ulteriore conferma all'utente. Se l'utente conferma, il sistema effettua l'addebito su carta di credito utilizzando le API del servizio esterno SmartPay. Se l'addebito va a buon fine, il sistema salva la prenotazione e mostra un messaggio di conferma all'utente. Se il pagamento fallisce, il sistema mostra un messaggio d'errore.

Esercizio 1

(a) Si modellino tutti i requisiti del sistema descritto sopra attraverso uno Use Case Diagram;
(b) Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di una prenotazione.
(c) Dettagliare il caso d'uso relativo alla funzionalità di inserimento di una prenotazione, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
(d) A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A

L'azienda per cui lavorate vuole addestrare un nuovo *Large Language Model (LLM)* Autoregressivo. Date le dimensioni del corpus di testo che sarà utilizzato per l'addestramento e l'architettura del modello stesso, si stima che l'addestramento richiederà all'incirca 15 giorni di calcolo per un cluster di 128 server con in totale 1024 GPU A100 Tensor Core Nvidia. Dopo la fase di addestramento, che è molto onerosa dal punto di vista delle risorse di calcolo, il modello potrà essere messo in uso su un semplice server con 8 GPU A100 Tensor Core.

Descrivere vantaggi e svantaggi (se ve ne sono) dell'utilizzo di servizi di public cloud per l'addestramento e la messa in opera del LLM di cui sopra.

Modulo A - Esercizio 3A

Il metodo `computeCosts` della classe `Billing` viene utilizzato per calcolare il prezzo dell'invocazione di una API per accedere a Large Language Model offerto con modello SaaS. Il metodo prende in input i seguenti parametri:

- `int tokens`: indica il numero (strettamente maggiore di 1 e minore di 512) di token (parole) inserite nel prompt dato come input;
- `boolean isPremium`: indica se il cliente ha richiesto il servizio premium, che permette di ottenere risposte più velocemente accedendo a risorse hardware dedicate.

Se i parametri non sono validi, il metodo solleva una `IllegalArgumentException`. In caso contrario, ritorna il prezzo dell'invocazione delle API da addebitare al cliente. Il prezzo dell'invocazione è calcolato moltiplicando il numero di token per il costo base per token, determinato in base al livello di priorità richiesto, come da tabella seguente. Inoltre, se i token sono più di 256, è prevista una ulteriore maggiorazione per prompt lunghi, fissata a 2€.

	STANDARD	Premium
Costo per token	0.01 €	0.05 €

a) Indicare, per ciascuno dei parametri del metodo `computeCosts`, le classi di equivalenza individuate.
b) Scrivere quattro test JUnit con strategia Black Box per il metodo `computeCosts`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
c) Quanti test sono necessari per testare il metodo con strategia WECT? Motivare la risposta.

Modulo B - Esercizio 3B

I prototipi si possono classificare secondo le seguenti caratteristiche: scopo, modalità d'uso, fedeltà, completezza funzionale e durata della loro vita. Per ogni caratterista, elencare le alternative possibili spiegandole brevemente.