Progetto del corso di programmazione avanzata (6 CFU)

Facoltà di ingegneria informatica

Università di Pisa

Questo progetto è stato realizzato per l'esame di programmazione avanzata.

Tale corso è incentrato sull'apprendimento delle basi di java e javafx.

Il progetto si basa su l'applicazione di ciò che è stato spiegato durante il corso, per realizzare una applicazione di gestione di una biblioteca.

Il progetto presenta degli errori, che in sede di esame sono stati fatti notare.

Per tanto, si prega di considerare questo progetto come puramente didattico.

Si raccomanda di usare il codice con le accortezze dovute.

Tra i file presenti nella repository, è possibile trovare anche il registro compilato ogni sessione di lavoro sul progetto.

I requisiti rispettati dal progetto sono i seguenti:

Criteri di qualità del progetto 8 of 28

CRITERIO	DESCRIZIONE
(1) REGISTRO CHIARO E IN FASE	- Il registro è adoperato prevalentemente online, e le attività si susseguono nell'ordine: - **(1) analisi:** si descrive cosa fa l'applicazione con degli scenari tipici e ispirati ai casi d'uso **(2) progetto:** si disegna un diagramma delle classi del sistema, tramite le conoscenze derivanti dai laboratori e/o prototipazioni - **(3) prototipazione:** si provano frammenti di codice dei tutorial per ampliare le conoscenze di alcuni componenti, prendendo estratti ed eseguendoli in locale, ma non si modificano - **(4) sviluppo-integrazione:** si realizzano e integrano le componenti dell'applicazione - **(5) collaudo:** si testa l'applicazione sugli scenari stabiliti nell'analisi - **(*) documentazione** può essere abbinata alle fasi precedenti - condurre la fase (2) fino al massimo possibile con le conoscenze dei laboratori, poi passare alla fase (3) per formarsi su nuovi componenti, e infine completare la fase (2); è ammesso un solo passaggio indietro da 3→2 all'interno di un macro ciclo; - si può ripartire con un nuovo macrociclo, da (5) a (1) per la 2^A o 3^A iterazione, per ampliare/correggere il progetto, motivandolo nel registro, ma non si può tornare indietro tra le attività a meno del caso 3→2.
(2) ARCHI-TETTURA LEGGIBILE E MODULARE	- classi distinte tra front-end, middleware e back-end - classi e attributi nominati con nomi, metodi nominati con verbi (da dizionario italiano) - i nomi e i verbi corrispondono alle funzionalità realizzate - indentare il codice, commentare il codice come note in fondo al file - metodi non più lunghi di una videata - separare lunghe strutture dati (es. stringhe, numeri) da codice
(3) COPERTURA PROGRAMM A DEL CORSO	l'applicazione contiene le seguenti 5 component di I/O: (i) interfaccia grafica in JavaFX; (ii) file di configurazione locale in XML/XSD; (iii) cache locale degli input su file binario; (iv) base di dati in JDBC/MySQL; (v) server/file di log remoto in XML/XSD

CRITERIO	DESCRIZIONE
(4) COESIONE DELLE FUNZIONALITÀ	- le classi appartengono al medesimo servizio primario e interagiscono nel caso d'uso. evitare di aggregare servizi indipendenti come 'autenticazione', 'inserisci lista della spesa' e 'cerca negozio', ma specializzarsi su uno di essi.
(5) PORTABILITÀ	- l'applicazione deve essere eseguibile sui PC Windows dell'aula con il pacchetto all-in-one. Fornire script sql, zip di progetto netbeans, eventuali librerie o script .bat
(6) DOCUMENTAZIONE	- composta dal documento di analisi, documento di progetto (con un diagramma di classe), e dal documento di collaudo (manuale utente)
(7) TEST FUNZIONALE	- esecuzione dello scenario descritto nel documento di collaudo; assumere che l'utente sia disciplinato e non considerare scenari di robustezza diversi dal caso d'uso
(8) CODICE INEDITO	- rilevanti parti di codice molto simile tra due progetti, anche di appelli diversi, rendono il codice edito per il progetto che viene presentato dopo
(9) UTILITÀ	- l'applicazione deve essere di utilità per qualche utente del mondo reale, lo scenario pensato è realistico. l'utente per il quale si sviluppa l'applicazione non deve essere nè lo sviluppatore nè il docente.
(10) NON RIDONDANZA DEL CODICE	- il codice è ridondante se vi sono parti di codice di controllo o di dati che sono molto simili. ridurre al minimo il popolamento dei dati nel database.
(11) CAPACITÀ DI MANUTENZIONE	- lo studente è in grado di manutenere l'applicazione: localizzare il codice corrispondente alle richieste del docente ed effettuare modifiche minime in sede di esame, accedendo alle specifiche e/o al web se necessario