Relazione su scelte progettuali e di interfaccia

Scelte progettuali

- **Maven**
  Abbiamo deciso di utilizzare lo strumento Maven per questo progetto perché ci sembrava più flessibile ed eravamo già abituati ad usarlo, al contrario di Eclipse che avrebbe richiesto un grande sforzo di adattamento.

- **Interfaccia Pagina**
  Il menu di navigazione del sito permette di cambiare il contenuto della pagina html, accedendo a specifiche sezioni di informazioni (nel nostro caso solo Homepage e Dipartimenti, ma in un vero sito web ce ne sarebbero diverse).
  Per questo motivo abbiamo voluto creare una struttura che permetta facilmente di aggiungere nuovi pulsanti al menu di navigazione.
  Questo è stato fatto creando un'interfaccia Pagina che viene concretizzata nelle diverse pagine accessibili dal menu (un pulsante reindirizza a una pagina).
  Ogni pagina implementa il metodo aggiungiContenuto che rimuove i contenuti precedenti e aggiunge quelli della pagina selezionata.
  Anche il pulsante Accedi al portale apre una nuova pagina che implementa questo metodo.

- **Service, ServiceAsync, ServiceImpl**
  GWT fornisce diversi modi per comunicare con un server. Dal momento che eseguiamo Java sul back-end e abbiamo bisogno di un'interfaccia per la logica dell'applicativo server-side, abbiamo deciso di utilizzare GWT RPC (Remote Procedure Call). In questo modo sfruttiamo il protocollo HTTP per passare oggetti da e verso un server: basta effettuare una chiamata a un servizio del server e lasciamo che GWT si occupi dei dettagli di basso livello come la serializzazione degli oggetti.
  L'implementazione di un servizio GWT RPC è basata sull' architettura Java servlet. I servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web (Jetty nel nostro caso) permettendo la creazione di applicazioni web.
  All'interno del codice client, usiamo una classe proxy generata automaticamente per fare chiamate al servizio.
  Per definire l'interfaccia RPC, abbiamo scritto tre componenti:
  1. Un'interfaccia (es. UtenteService) per il servizio che estende RemoteService ed elenca tutti i metodi RPC.
  2. Una classe (es. UtenteServiceImpl) che estende RemoteServiceServlet e che implementa l'interfaccia creata sopra.
     L'implementazione del servizio non implementa la versione asincrona dell'interfaccia del servizio. Ogni implementazione di servizio è in definitiva una servlet, ma invece di estendere HttpServlet, estende RemoteServiceServlet. Quest'ultima gestisce automaticamente la serializzazione dei dati che vengono passati tra il client e il server e l'invocazione del metodo previsto nell'implementazione del servizio.

3. Un'interfaccia asincrona (es. UtenteServiceAsync) per il servizio da chiamare dal codice lato client.

Gli oggetti passati devono essere serializzabili e quindi possono essere:

- Tutti i tipi primitivi (int, char, boolean, ecc..) e i loro oggetti wrapper sono serializzabili per default.
- Un array di tipi serializzabili è serializzabile per estensione.
- Una classe è serializzabile se soddisfa questi tre requisiti:
    - Implementa l'interfaccia Java Serializable o GWT IsSerializable, direttamente o perché deriva da una superclasse che lo fa.
    - I campi delle istanze non finali e non transitori sono essi stessi serializzabili.
    - Ha un costruttore di default (zero argomenti) con qualsiasi modificatore di accesso (ad esempio private Foo(){} funzionerà)

- **Interfaccia Form**
  Abbiamo deciso di implementare un'interfaccia Form che viene concretizzata in tutte quelle classi che dovranno fornire un form (le classi di inserimento e modifica).
  Tutte le classi che implementeranno l'interfaccia Form, in questo modo, avranno in comune il metodo getForm (implementato diversamente a seconda dell'utilizzo) che potrà essere chiamato nei vari portali.

- **Classe astratta Portale**
  Abbiamo voluto implementare il portale come se fosse un'applicazione web separata dal vero e proprio sito che è accessibile a chiunque.
  Per questo motivo abbiamo creato una classe astratta Portale che viene concretizzata nei quattro portali specifici per ogni tipo di utente (studente, docente, segreteria e admin).
  Abbiamo deciso di utilizzare una classe astratta perché la struttura grafica del portale è la stessa per tutti gli utenti, mentre sono i contenuti ad essere diversi (ad esempio i pulsanti nella barra laterale).
  Questa gerarchia ci permette dunque di ereditare il metodo caricaPortale e di sovrascrivere i metodi caricaMenu, per aggiungere i pulsanti alla barra laterale e caricaDefault, per mostrare il contenuto di default quando si accede.

- **Strutturazione del database**
  Per salvare i dati nel database in modo chiaro e leggibile abbiamo preferito cercare di ricreare un database di tipo relazionale con MapDB.
  Per fare ciò, abbiamo creato una classe per ogni "tabella" con gli attributi che rispecchiano le colonne della tabella.
  Le istanze delle classi vengono salvate all'interno del rispettivo database che corrisponde a un file .db.

Scelte di interfaccia

- **Header e footer**
  Per la struttura dell'applicazione abbiamo deciso di ispirarci ai classici siti web creando un header e un footer che sono sempre presenti in qualunque schermata.
  Il footer è complementamente statico, infatti contiene solo alcune voci fittizie del sito e i contatti.
  L'header contiene il logo, il nome dell'università (il quale è un pulsante che reindirizza sempre alla homepage, anche dopo aver fatto il login) e un pulsante che può essere di login o logout.

- **Menu di navigazione (sito)**
  Sempre nell'ottica di riprodurre la struttura dei siti web moderni abbiamo deciso di aggiungere un menu di navigazione orizzontale subito sotto l'header (presente solo nel sito e non nel portale).
  Questo menu permette di navigare con facilità nelle diverse pagine del sito.

- **Barra laterale (portale)**
  Una volta che si accede al portale dell'università la barra di navigazione orizzontale sparisce e viene sostituita da una barra laterale arancione.
  Questa barra contiene i pulsanti rappresentano le azioni principali che un utente può svolgere a seconda della propria categoria.
  Abbiamo scelto questo metodo di navigazione perché ci sembrava il più intuitivo e user-friendly.

- **Tabelle**
  In molte sezioni del portale sono presenti delle tabelle (Grid), ad esempio quelle per visualizzare la lista dei corsi disponibili o per visualizzare la lista di studenti e docenti.
  Abbiamo deciso di utilizzare delle tabelle perché permettono di mostrare tante informazioni in maniera chiara e schematica, ad esempio quando si vogliono visualizzare i corsi a cui iscriversi tutte le informazioni rilevanti sono già mostrate a fianco del nome del corso.
  Allo stesso modo, anche tutte le azioni possibili per una voce della tabella sono facilmente accessibili.

- **Form**
  I form sono presenti in molte sezioni del portale (generalmente in quelle di inserimento e modifica).
  I form permettono di compilare i campi con le informazioni desiderate e di poterle, poi, inviare al server per l'inserimento nel database.
  Nella maggior parte dei form sono presenti dei campi obbligatori e se al momento dell'invio dei dati questi non sono compilati, verrà visualizzato un messaggio di errore in cui si chiede, appunto, di compilarli.