Università degli Studi di Torino, A.A. 2015-2016

Tecnologie Web (6 cfu) - MFN0634

Specifiche progetto finale

Obiettivi generali

Il **progetto finale** deve essere consegnato almeno tre giorni prima della data d'appello scelta dallo studente e deve essere preceduto (consegna 15 giorni prima del caricamento del progetto) da un **documento** che descriva la struttura e le funzionalità principali del sito che si intende realizzare.

Scopo del presente documento è quello di fornire le specifiche sia del documento di progetto che del sito di cui si chiede la realizzazione.

Il **tema del sito è libero ed a scelta dello studente.** Ciononostante, il sito dovrà essere strutturato in modo tale da seguire le specifiche indicate. Parte importante della realizzazione del sito è l'**analisi dei requisiti** che dovranno essere soddisfatti per poter superare l'esame. Scopo principale della relazione che deve essere redatta e consegnata (in linea teorica)[1] *prima* di iniziare ad implementare il sito stesso è quella di spiegare come i requisiti richiesti saranno realizzati durante l'elaborazione.

Lo studente potrà utilizzare tutte le **tecnologie viste durante il corso:** HTML(5), CSS(3), PHP5.4 (OO, sessioni, connessione a db), JavaScript (Eventi JS, DOM), librerie JS (Prototype/Scriptaculous oppure JQuery), Bootstrap, Ajax, scambio dati (XML o JSON), servizi web e cosí via. I problemi che sono stati analizzati e per i quali sono state discusse le soluzioni (usabilità, separazione presentazione/contenuto/comportamento, sicurezza del web, etc.) dovranno essere affrontati: scopo dell'elaborato finale è quello di consentire la connessione di tutti gli argomenti discussi con lo scopo di realizzare un sito web professionale.

---

[1] La consegna della relazione di progetto deve avvenire entro 15 giorni prima del caricamento del sito stesso su OpenShift. Questo non significa che dovete necessariamente realizzare il sito in sole due settimane: l'idea è di "simulare" un contesto professionale dove l'analisi dei requisiti viene realizzata prima di iniziare lo sviluppo vero e proprio. Sta a voi gestirvi il tempo nel modo migliore possibile.

Requisiti

Segue la descrizione di massima delle funzionalità generali che devono essere integrate nel sito finale; quindi, sarà spiegato cosa ci si aspetta dalla relazione che deve precedere la consegna del sito.

Il sito

Come precedentemente chiarito, il tema del sito lo dovete decidere voi. Potete decidere di simulare un sito di commercio elettronico, un semplice sito di intrattenimento, un sito informativo ed interattivo, un'estensione/rivisitazione di uno dei tanti esercizi visti durante il laboratorio, etc.

La cosa importante è che il sito abbia una determinata **struttura architetturale** e che presenti alcune determinate **funzionalità.** In questo documento presenteremo i requisiti riguardo a questi aspetti.

Struttura - L'architettura del sito deve essere così strutturata:

- **front-ent:** L'utente accede al sito tramite una **home-page**, che corrisponde all'index della cartella "**progetto**". Tutto il sito deve essere coerente stilisticamente a quanto mostrato nella pagina principale. A questo livello bisogna stare molto attenti a dividere in maniera quanto più possibile presentazione (CSS), contenuto (HTML) e comportamento (JS). È consentito l'uso di Boostrap, librerie come Prototype/Scriptaculous o jQuery, etc. È importante organizzare bene le dipendenze (considerare l'inclusione dei fogli di stile e delle funzioni JS tramite file esterni e non incorporati direttamente nel file html). I file, a loro volta, devono essere organizzati per cartelle (ad esempio tutte le immagini dentro una cartella **img**, i fogli di stile dentro una cartella **css**, così come il codice javascript andrebbe sotto una cartella **js**, etc.). Evitate codice ridondante: prevedete di usare file separati per parti di codice che si ripetono in pagine diverse (header, footer, menù, etc.).

- **back-end:** tutto quello che l'utente non vede direttamente sul browser è stato sviluppato e mantenuto lato server. Separate logicamente le parti che necessitano di un'intelligenza "server-side". Usate PHP e mantenete i dati con MySQL (eventualmente aiutandovi con lo strumento phpMyAdmin). È molto probabile che il numero di pagine php superino quelle html: in linea di principio non avete bisogno di nessun file html, anche se qualche porzione di codice HTML di supporto (es., header, footer) in alcuni casi potrebbe non contenere codice php. Anche in questo caso organizzate molto bene le informazioni: il database deve seguire i principi dell'approccio relazionale, le **query SQL** devono essere il più possibile efficienti e ottimizzate (es., evitare molte JOIN annidate su tabelle grosse). Potete realizzare il vostro database da zero usando dati fittizi, così come potete partire da uno dei database che abbiamo usato durante il corso o da altri a vostra disposizione (in questo caso, controllate bene i diritti che avete sullo sfruttamento di quei dati). Infine, organizzate bene anche le cartelle dove andrete a memorizzare i file php: anche se non seguite alla lettera un approccio MVC (che siete liberi comunque di adottare), sarebbe bene memorizzare le funzioni e le classi php in file che hanno nomi significativi e che siano anche organizzati in base alle loro logiche funzionali (es., i file che riguardano la gestione dei profili utente sotto una cartella **users**, quelli che riguardano la gestione del carrello della spesa sotto una cartella **shop**, etc. - se invece seguite il pattern MVC sarebbe meglio usare i nomi delle cartelle **model**, **view**, **controller** per

poi creare sottocartelle con i nomi delle classi php che definite per ognuna delle categorie MVC).

- **comunicazione back/front-end:** prevedete di scambiare i dati tra le parti client e server del vostro sito usando la tecnologia **Ajax.** Questo rende l'aggiornamento dinamico della pagina molto più efficiente. Non usate un formato di dati qualsiasi. Scegliete **XML** oppure **JSON:** questo vi farà imparare a fare le cose per bene, ma soprattutto avrete a disposizione le tante librerie (Prototype o JQuery) che vi faciliteranno la gestione del file ricevuto dal server e il successivo aggiornamento del **DOM HTML.** Cercate di sfruttare il più possibile i principi dei **web service:** i programmi PHP accedono ai dati lato server, li convertono nel formato scelto (XML e JSON) e li inviano al client sul canale aperto da AJAX. I metodi HTTP che potete usare per realizzare queste chiamate possono essere sia GET che POST; l'output dei programmi PHP **non** sarà in questo caso codice HTML.

Funzionalità - Il sito deve offrire le seguenti funzionalità:

- **login/logout:** l'utente che arriva all'home page trova soltanto una pagina che chiede l'inserimento delle opportune credenziali. Non si può usare il resto del sito se non vengono inserire i dati di un utente registrato.

- **registrazione:** deve essere possibile per un utente nuovo potersi registrare al sito. Non realizzate una registrazione complicata: non chiedete l'attivazione da parte di un amministratore. O la registrazione và a buon fine, oppure fallisce: gestite entrambe le situazioni, fornendo un opportuno riscontro all'utente.

- **usabilità:** il sito deve seguire i principi di usabilità che abbiamo visto a lezione. Ad esempio, ad ogni azione dell'utente deve essere fornito un feedback (sia in caso di successo che di fallimento), i messaggi devono essere chiari, ma non dare dettagli sulla struttura del sito o del codice (incomprensibili per gli utenti normali e pericolosi da mostrare agli utenti malintenzionati); non usate finestre di pop-up (semplicemente: non usate le finestre di pop-up); usate combinazioni di colori adeguate ed evitate sfondi che possono infastidire la lettura; etc.

- **interazione/animazione:** prevedete una sezione del sito che mostri un'animazione interattiva all'utente. Considerate qualcosa di difficoltà paragonabile al gioco del 15 (puzzle game) che abbiamo proposto durante il corso. Questa componente deve essere comunque coerente con il resto del sito (drag&drop di oggetti da acquistare nel carrello della spesa, film da guardare in una sorta di playlist virtuale, un giochino - semplice! - al termine del completamento di un determinato questionario scientifico, etc.). Siate creativi! Ma tenete anche ben saldi i piedi per terra...

- **sessioni:** gestite lo stato lato server usando le sessioni php. Ricordate di aprirle e chiuderle sempre al momento giusto (e nel posto giusto a livello di codice...).

- **interrogazione del database:** considerate la creazione di opportuni form che permettono un'interrogazione d'alto livello (e coerente con gli scopi del sito) di dati che sono presenti lato server. Ad esempio, potreste voler vedere gli acquisti che avete effettuato nel passato, la vostra lista della spesa attuale, la lista dei desideri, gli altri utenti con il profilo più simile al vostro, i film diretti da un determinato regista, tutte le ricette che contengono il vostro ingrediente preferito, etc.

- **validazione dati input:** i dati immessi dagli utenti devono essere sempre validati, sia lato client che lato server

- **sicurezza:** attenzione agli attacchi! Non deve essere possibile compiere attacchi XSS ed HTML/SQL Injection.

- **presentazione:** la presentazione del sito deve essere "bella". Senza che il design vada a scapito dell'usabilità, il sito deve essere chiaro, facile da usare e da leggere, con elementi ben distinti tra di loro, etc. Considerate l'adozione di un layout flessibile, responsive, etc., preferendo uno schema a due o tre colonne invece che a pagina piena.

- **single-page application:** l'intero sito potrebbe essere visto dall'utente come se fosse un'unica pagina. Si potrebbe prevedere un'unica pagina di ingresso index.php che può essere richiamata senza o con parametri (usando in modo appropriato i metodi HTTP get o post). La dinamicità della pagina viene determinata dai parametri passati o dallo stato della sessione in corso. (**NOTA:** ricordate che per avere la sufficienza dovete realizzare almeno il 60% dei requisiti richiesti complessivamente in questo documento. Date priorità alle cose che volete realizzare. Questa funzionalità potrebbe essere considerata solo se siete veramente sicuri di potere realizzare facilmente tutto il resto).

Termini e scadenze: il sito deve essere caricato su OpenShift **entro 3 (tre) giorni** prima della data d'appello ed essere raggiungibile dalla URL: **http://tweb-*nomecognome.rhcloud.com*/progetto**
Inoltre dovrete comunicare al docente la consegna del progetto. Questo potete farlo semplicemente caricando la URL del progetto nella **cartella moodle** per la consegna chiamata "**Progetto finale**" (che sarà disponibile subito dopo che sarà trascorso il termine per la consegna delle relazioni).

L'analisi dei requisiti

Scopo dell'analisi dei requisiti è di presentare il progetto che si vuole realizzare, disegnando la struttura funzionale del sito, presentando l'idea di interaction design, etc. In particolare, si chiede che - viste e considerati i requisiti descritti precedentemente - lo studente spieghi in linea di massima dove e come intende affrontarli - dando nel contempo un'idea di come si dovrebbe presentare il prodotto finito.

Andando nello specifico, la relazione che dovete presentare deve descrivere, in cinque pagine al massimo (facendo riferimento ad un formato simile a quello del presente documento):

- **il tema del sito:** descrizione generale del contenuto e degli obiettivi del sito.

- **le sezioni principali:** considerate una suddivisione in macro-sezioni, ognuna delle quali può corrisponde ad una voce del menù principale. Ogni sezione deve avere uno scopo e può essere associata ad una o più funzionalità descritte nella sezione precedente

- **lo schema di navigazione base:** la descrizione dell'interaction design. Come si passa da una sezione all'altra, cosa succede se l'utente compie una determinata interazione, etc.

- **i mock-up delle sezioni più rilevanti del sito:** a questo scopo potete usare uno dei tanti strumenti a disposizione, come ad esempio Mockingbird (https://gomockingbird.com/home) - oppure disegnare "a mano" i wireframe.

Non dimenticate di dare una descrizione di massima del modo in cui pensate di poter realizzare tecnicamente le varie funzionalità (in modo davvero molto sintetico, tanto durante lo sviluppo molto probabilmente vi troverete ad adottare una soluzione diversa da quella che avevate in mente in questo stadio della progettazione!). Considerate che la relazione **non sarà valutata in base a quanto siete stati bravi ad azzeccare le soluzioni tecniche!** Sentitevi liberi in questa fase di mettere nero su bianco le difficoltà che credete di dover affrontare ed il modo in cui credete di poterle risolvere: sarà più istruttivo dopo aver realizzato il progetto capire in cosa vi sbagliavate e perché.

Termini e scadenze: il documento con l'analisi dei requisiti deve essere caricato sulla **cartella moodle** per la consegna chiamata "**Documento analisi dei requisiti**" (che sarà disponibile qualche giorno prima della scadenza e comunque entro 18 giorni prima dalla data dell'appello).

Sviluppo e valutazione

Consigli per lo sviluppo, deploy e test

Avete imparato diverse tecniche per fare debugging. Riassumiamole brevemente:

- verificate che il vostro ambiente **git/openshift** sia ben configurato. Se qualcosa non funziona per bene, considerate la possibilità di resettare tutto ed iniziare da capo con un nuovo ambiente prima di cominciare lo sviluppo del nuovo sito.

- se sapete usare **Eclipse** o **NetBeans** o altro ambiente di sviluppo, avete una marcia in più. Considerate la possibilità di impararli cogliendo l'occasione di questo progetto.

- usate i validatori ufficiali w3c per **html/css**

- usate gli **strumenti per lo sviluppatore** dei vostri browser. La console è la vostra migliore amica, ma ricordate anche anche il debugger built-in è molto utile, per non considerare la rappresentazione del box-model, la possibilità di analizzare le risorse di rete, quella di modificare proprietà css e codice html "al volo" per valutarne l'effetto, etc. Provate su browser diversi!

- attivate la modalità **hotdeploy** su openshift per velocizzare il caricamento (add/commit/push) via git dei vostri file. Guardate le istruzioni sul sito di openshift oppure quelle che ho messo su moodle.

- per poter fare debugging del codice lato server, valutate di configurare opportunamente **XDebug** su OpenShift.

Implementazione e valutazione

Durante la correzione del progetto gli errori conteranno davvero. State attenti ai problemi grossi, ma la cosa migliore è partire da un codice pulito. Sotto sono riassunti i consigli principali che abbiamo dato durante il corso.

HTML, CSS: Ricordare di usare un HTML(5) validato con W3C HTML(5) validator. Non inserite informazioni di stile nel HTML5, come fogli di stile inline o tag HTML come b o font.

Inserite tutte le informazioni di stile in uno o più file CSS. Per fare tutto correttamente, il vostro CSS deve superare la validazione con W3C CSS (jigsaw). Per esempio, se dovete usare spesso le stesse regole di stile, non definitele più volte nel foglio di stile. Se si usa lo stesso colore o tipo di font per più elementi, occorre raggruppare questi elementi in una singola regola di stile nel CSS. Scegliete di usare i selettori per evitare di definire più id e class. Limitate l'uso di posizioni fixed o absolute. Non usate le tabelle di HTML per definire il layout.

Formattate il vostro HTML e CSS in modo da renderlo più leggibile possibile. Ponete un commento nella <head> di ogni file con il vostro nome, cognome, breve descrizione del sito e il contenuto del file. Usate in modo opportuno spazi bianchi e indentazione. Per rendere la lunghezza delle righe maneggevole, non ponete più di un blocco di elementi in una linea, o vai a capo dopo massimo 100 caratteri.

Cercate di strutturare il vostro codice in un modo simile agli esempi visti in classe, facendo particolare riferimento ai diversi casi di studio che abbiamo analizzato al termine di ogni capitolo. Usate spazi bianchi ed intestazioni in modo appropriato. Non mettete più di un elemento blocco per ogni linea della vostra pagina HTML.

PHP, form e DB: L'output HTML per tutte le pagine dovrebbe superare il test di validazione W3C. Ovviamente il test non coinvolge il codice PHP, ma solo l'HTML da esso generato). Non usate tabelle HTML per definire il layout, ma solo per contenere dati ed informazioni (<table> serve per organizzare il contenuto, non per organizzare la presentazione e lo stile). Quando usate i form HTML, scegliete i controlli corretti ed i loro attributi di conseguenza. Scegliete GET e POST per spedire i dati al server nel modo corretto. Il vostro codice PHP non deve generare **warning** o **errori** (possibilmente neanche **notice**). Minimizzate l'uso di variabili globali (a volte sono necessarie, ma abituatevi ad usare funzioni parametrizzate), usate correttamente l'indentazione e la spaziatura, evitate linee più lunghe di cento caratteri che rendono il codice difficile da leggere. Usate il materiale prodotto durante le settimane precedenti e quello allegato ai vari capitoli del libro di testo (il materiale è reperibile su moodle).

Non filtrate i vostri dati con PHP: usate le giuste query SQL piuttosto. Se fate così, potrete trattare i vostri dati molto più semplicemente ed efficientemente. Ad esempio, è decisamente un esempio di pessimo stile di programmazione recuperare tutti i dati di una tabella dal db e poi eseguire complesse operazioni sull'array ottenuto per cercare le informazioni che corrispondono alla richiesta dell'utente. SQL è un linguaggio potente: usatelo.

Per fare tutto alla perfezione, riducete la quantità di codice PHP nel mezzo delle vostro codice HTML. Il codice ridondante può essere inserito in una funzione, da dichiarare dentro un file che può essere incluso in altri file. Ricordatevi di limitare il più possibile le istruzioni PHP print ed echo: usate piuttosto le espressioni blocco <?= ... ?>, così come abbiamo visto in classe.

Un altro aspetto importante è legato all'uso dei *commenti* PHP. Ad esempio, all'inizio di ogni file, di ogni funzione e di ogni blocco PHP, ricordatevi di mettere dei commenti descrittivi.

JS, DOM, librerie: Il (o i) file .js deve(devono) essere il più possibile privo(i) di errori (usate gli strumenti dello sviluppatore del vostro browser anche per il debugging).

Usate poche variabili globali ed evitate codice ridondante, impiegando anche opportuni parametri e la restituzione di valori in modo appropriato. Le operazioni più frequenti devono diventare funzioni, per evitare che il codice diventi troppo complicato e lungo.

Sono consentite alcune variabili globali, ma non troppe: si devono usare soprattutto variabili locali. Se una variabile è usata spesso, dichiaratela come "costante" definendola in lettere maiuscole (es: IN_UPPER_CASE) e usandola nel codice. Non salvate come variabili globali gli oggetti DOM, come nel caso dei valori restituiti da $ o da $$ o dalla funzione document.getElementById.

Commentate in modo adeguato il codice JavaScript. All'inizio del file descrivere il programma e inoltre descrivete le sezioni più complicate del vostro codice.

Separate la parte di content (HTML), dalla presentation (CSS), e dal behavior (JS). Il codice JS dovrebbe usare stili e classi del CSS, piuttosto che settare ogni proprietà in JS.

Usate in modo corretto JavaScript in modo da non avere altro codice, né funzioni onclick o altro, incluso nel codice HTML: adottate uno stile JS "discreto" (unobtrusive).

Gestite gli eventi e l'attraversamento del DOM con le librerie JS che abbiamo visto, adottando la "famiglia" Prototype/Scriptaculous oppure JQuery (a vostra scelta).

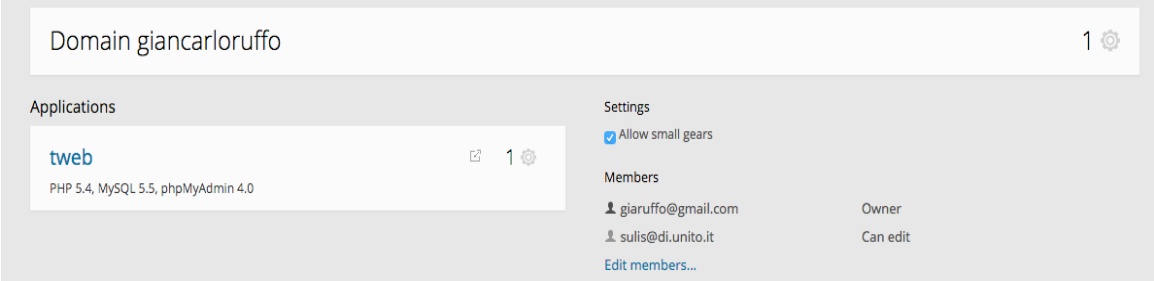
Consegna

Ricordate infine che il compito dovrà essere consegnato esclusivamente su openshift e che su moodle dovete indicare solo la URL del vostro sito:

http://tweb-*nomecognome.rhcloud.com*/progetto/

In fase di consegna, due cose molto importanti da fare sono le seguenti:

1. Dovete consegnare su moodle soltanto la url di sopra. Solo testo, no html, no commenti. NIENTE. Solo la url che consenta l'accesso al vostro sito.
2. (**Solo se non le avete già fatto**) Dovete aggiungere il nome del docente ai vostri domini per consertirmi di vedere il vostro codice php. Cliccate, sulla web console di OpenShift sul nome del dominio (ad esempio, giancarloruffo o il vostro nomecognome). Arrivate alle seguente pagina:



Quindi selezionate "Edit members" e/o "Add a user..." e abilitate alla modifica ("Can edit") TUTTE la seguente identità: **giaruffo@gmail.com**

Voto: assegnato per completezza e correttezza

In Figura 1 si può trovare lo schema riassuntivo che consente di capire la determinazione del voto finale. In Figura 2 è presentata la variante nel caso in cui lo studente sia non frequentante (ovvero non abbia mai consegnato i laboratori in classe o ne abbia consegnati meno di 4).

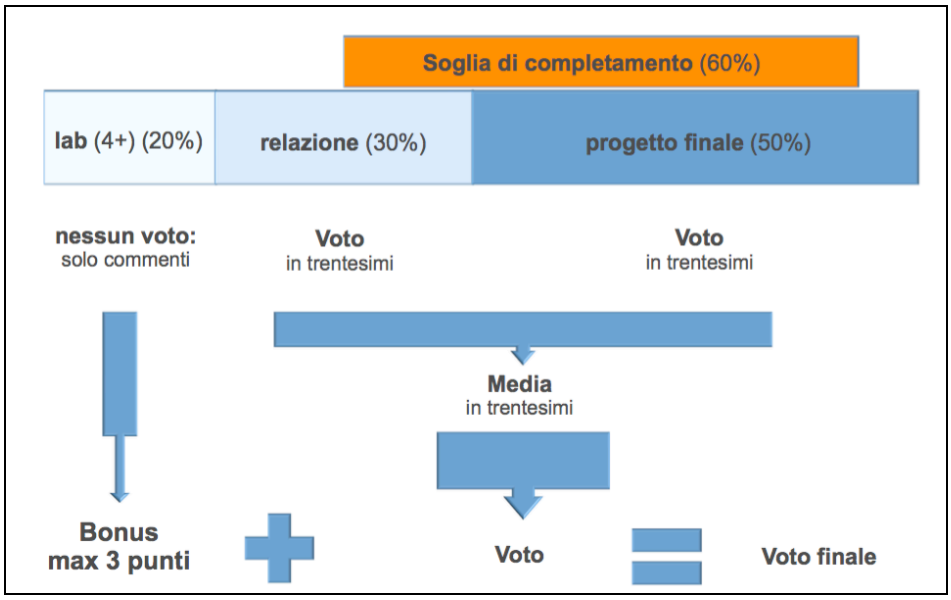


Figura 1: calcolo voto studente frequentante

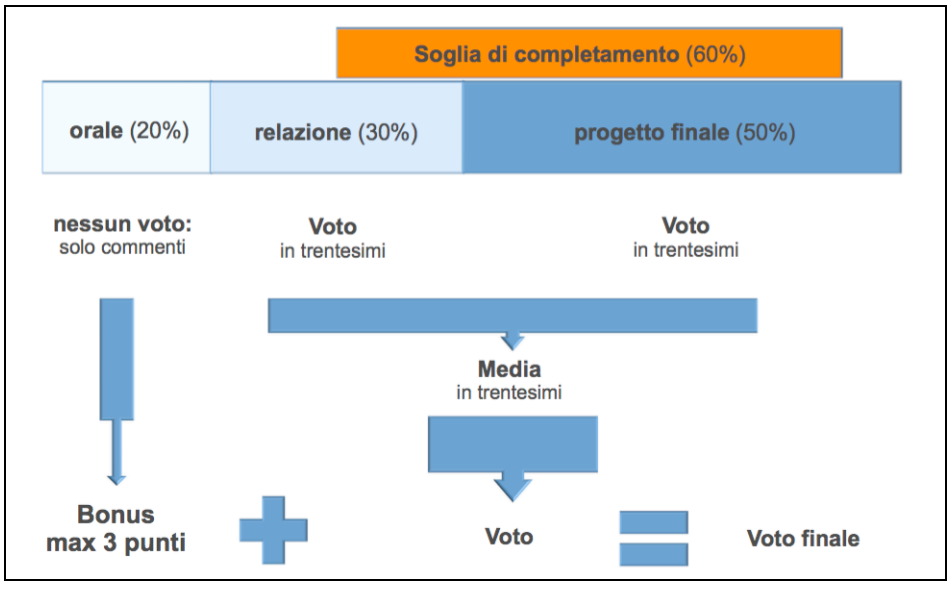


Figura 2: calcolo voto studente non frequentante

Tutti gli studenti dovranno sostenere un esame finale che consisterà nei seguenti tre passaggi:

- Consegnare la relazione "**analisi dei requisiti**" sulla **cartella moodle** per la consegna chiamata "**Documento analisi dei requisiti**" entro **18 giorni prima** dalla data dell'appello.

- Caricare il **sito** su OpenShift **entro 3 (tre) giorni** prima della data d'appello. Comunicare al docente l'avvenuta consegna semplicemente caricando la URL del progetto nella **cartella moodle** per la consegna chiamata "**Progetto finale**".

- **Discutere il progetto:** il giorno dell'appello d'esame, saranno rivolte allo studente domande specifiche sul progetto realizzato, per meglio chiarire aspetti implementativi o progettuali, o per meglio capire le scelte operate.

Inoltre, gli **studenti non frequentanti**, dovranno:

- **Sostenere un esame orale:** allo studente che non ha consegnato gli esercizi (o ne ha consegnati meno di 4) durante il corso possono essere rivolte delle domande specifiche sugli argomenti affrontati durante il corso e che non necessariamente hanno a che fare con il progetto realizzato.

Il voto sarà assegnato in base a due criteri: **correttezza** e **completezza**.

- Per **completezza** si intende che dovrà realizzare almeno il 60% dei requisiti specificati per la relazione e per il progetto in questo documento.

- Per **correttezza** si intende che quanto realizzato deve essere corretto secondo i principi appresi durante il corso.

- Al voto assegnato possono essere aggiunti fino ad un **massimo di +3 punti** in base alla partecipazione attiva (esercizi assegnati) o all’esame orale (non frequentanti)

Si ricorda inoltre che gli studenti possono presentarsi ad **un qualsiasi appello** previsto durante l’anno in corso e che le modalità discusse in questo documento valgono per tutta la durata dell'anno accademico in corso.