ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il progetto del corso di Tecnologie Web - A.A. 2021/22

Fabio Vitali, Angelo di Iorio
Francesco Sovrano
Vincenzo Rubano
Corso di laurea in Informatica
Alma Mater – Università di Bologna

Il progetto di fine corso

- Un sistema VERO, che funziona e fa cose utili
- Realizzabile sia in laboratorio che a casa.
- Enfasi in parte sulla programmazione (approccio procedurale) ma soprattutto sui documenti attivi (approccio dichiarativo)
- Enfasi sul mashup di tecnologie esistenti e sofisticate

Avvertenze

- Gli studenti di Informatica per il Management ricevono una presentazione di progetto simile a questa
- I progetti sono compatibili tra loro ma si differenziano in rapporto ai CFU del corso nel piano di studi
- La regola si applica anche ai corsi mutuati: per un corso da 9CFU fate riferimento a questo progetto, per un corso da 6CFU al progetto per Informatica per il Management
- I gruppi di studenti di corsi mutuati possono includere anche studenti di Informatica o Informatica per il Management

Organizzazione dei team

- Ogni persona decide in anticipo se è interessata a sostenere l'esame in estate, autunno, sessione straordinaria o essere ancora indeciso.
- Tutti gli studenti si dividono in team di 2-3 persone. Attenzione: meno di 2 significa troppo lavoro individuale. Più di 3 significa troppo poco. AL MASSIMO 3. NIENTE ECCEZIONI.
- Ogni team porta il progetto insieme (anche qui, non ci sono eccezioni!). Il team dichiara in anticipo la natura del contributo di ciascun membro oppure accetta che chiunque sia interrogato (e nel dettaglio) su tutto il progetto.
- Il sottoscritto NON è coinvolto nell'organizzazione dei team.

Il lavoro di team

- Tutti i membri dei team sono tenuti a lavorare e lavorare insieme.
- E' meglio essere parte attiva di un progetto mediocre che passiva di un progetto meraviglioso.
- Non saranno tollerati i portatori di pizze
- Mi riservo all'esame di scoprire il contributo individuale di ciascuno, indipendentemente dalla bontà del progetto consegnato.

Ruolo di queste specifiche

- Questo documento contiene le specifiche fondamentali del progetto di fine corso.
- Quanto scritto in nero, salvo esplicite eccezioni, deve essere considerato requisito OBBLIGATORIO per la consegna.
  - [Le frasi scritte in arancione e parentesi quadre si riferiscono a servizi opzionali per migliorare la valutazione, e sono ***non obbligatorie***.]
  - {Le frasi in verde e parentesi graffe corrispondono a vincoli ***obbligatori*** introdotti solo per le esigenze del progetto universitario, non necessari o opportuni in un prodotto vero per il mercato esterno.}
- Se una o più delle specifiche qui introdotte non funzionano, il progetto NON è considerato accettabile.
- ***Un'apposita pagina su Virtuale fornirà in maniera sempre aggiornata le eventuali modifiche ai requisiti.***

Un po' di teoria

Background: strumenti per lo sviluppatore web negli anni '20

Lo sviluppo sw più sorprendente ed evidente degli ultimi anni è la nascita di innumerevoli strumenti per il programmatore, con cui realizzare servizi sofisticati

- Framework: invece di scegliere nuovi linguaggi di programmazione, si usano ora librerie per gli scopi più disparati, facilmente integrabili e mescolabili tra loro
- API (Application Programming Interfaces): invece di sviluppare applicazioni monolitiche che svolgono servizi complessi in un'unica maniera, si forniscono meccanismi di manipolazione delle strutture dati fondamentali e accesso agli algoritmi più sofisticati per applicazioni sviluppate dai clienti.

Questo permette incredibile sofisticazione, grande componibilità, e rapidità di sviluppo precedentemente irraggiungibili.

I framework

- I framework sono librerie che rendono più ricco, sofisticato e semplice l'uso di una tecnologia, come un linguaggio server-side, un linguaggio client-side o le specifiche grafiche di una pagina web.
- Server-side esistono dalla fine degli anni novanta, e hanno reso la programmazione a tre livelli drasticamente più facile.
- Client-side si sono sviluppate a partire dal 2002, su CSS e Javascript, con scopi molto difformi.

Application Programming Interface (API)

- Librerie, protocolli e strumenti per permettere di utilizzare gli algoritmi ed i servizi messi a disposizione da un software da parte di un altro software, invece che da esseri umani.
- Applicazioni e servizi che forniscono una API delegano ad un'applicazione terza aspetti come interazione, interfaccia, navigazione, ecc. e forniscono via API il solo servizio nudo.
- Uno degli scopi tipici di ricorso ad API è per integrare più servizi in un'applicazione più ricca e potente di quelle utilizzate come base.
- Questo si chiama *mashup* ed è una delle caratteristiche più evidenti di questo periodo storico: mescolare servizi di base per ottenere applicazioni impreviste dai fornitori dei servizi stessi.

Le API REST

- Le API che ci interessano sono più concretamente quelle che vanno sotto il nome di RESTful API, ovvero che sfruttano al meglio la natura di HTTP e degli URI (i protocolli più importanti del web) per fornire i loro servizi.
- Una API RESTful fornisce:
  - Un URI base a cui accedere per ottenere i servizi
  - Una sintassi degli URI delle entità interrogabili e modificabili
  - Un media type attraverso cui ottenere e fornire dati da utilizzare nei servizi forniti (ad esempio XML, JSON, etc.)
  - Una semantica associata all'uso dei vari verbi HTTP (GET, PUT, POST, DELETE) attraverso i quali attivare e eseguire i servizi offerti.
- La documentazione di un'API tipicamente descrive nel dettaglio nomi (URI), verbi (comandi HTTP) e formati (Internet Media Type) per ogni servizio fornito.

Le API di servizi locali

- Le seconde API che ci interessano sono quelle che permettono alle applicazioni web di accedere a speciali servizi offerti dal device su cui vengono eseguite (tipicamente accesso a periferiche di I/O particolari.
- La diffusione di TCP/IP su device mobili ha ampliato radicalmente la quantità e qualità di periferiche "non tradizionali" a cui l'applicazione può avere accesso:
  - Telecamera (Camera API)
  - Microfono e sintesi vocale (Speech API)
  - Geolocazione (via GPS o altro) (Geolocation API)
  - Suoni (Web Audio API)
  - Vibrazioni (Vibration API)
  - Telefono (Web telephony API),
  - ecc. ecc.
- Ci sono circa 80 API diverse al momento utilizzabili sulla maggior parte dei browser (purché il device offra la funzionalità)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Animal House

Animal House

Un sistema gestionale per una catena di negozi e pensioni VIP per animali domestici.

I nostri ospiti
a quattro zampe

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Animal House: background (1)

Fate parte di un’azienda che sviluppa software gestionale per prodotti e servizi per animali domestici. Il vostro cliente più importante è l'azienda Animal House, leader di settore.
Animal House è una realtà importante e diffusa sul territorio: ha più di 80 sedi sparse per il territorio italiano, e offre una varietà di servizi in presenza e in remoto.
I servizi offerti sono molteplici: prodotti per animali domestici, veterinari e psicologi per animali, pensioni per cani, vendita di animali di razza e mercato libero per i cuccioli dei propri clienti.
Animal House offre anche servizi di fascia alta destinati a clienti danarosi e viziati: pensioni a 4 stelle, psicologi per animali, addestratori, wedding planner, ecc.
Queste complessità possono rendere complesso e non banale rappresentare agli utenti la quantità e la qualità dei servizi per la clientela normale e quella "VIP".

Animal House: background (2)

Dovete costruire applicazioni per la gestione dei servizi di Animal House.
Le applicazioni sono tre:

- Il *game*, app brandizzata di intrattenimento che permette ai proprietari di animali domestici di avere informazioni e curiosità sui propri cari, fare giochi a tema animalesco, e confrontare prodotti dello store. Mobile first, solo online, nessun database.
- Il *front-office*, ovvero l'applicazione per i clienti, mobile first, per permettere le attività di accesso e uso di tutti i servizi online e per la prenotazione di quelli in presenza per i clienti *regular*.
- Il *back-office*, ovvero l'applicazione per gli impiegati dell'azienda che, sede per sede, inseriscono e rimuovono servizi e prodotti, gestiscono prezzi e descrizioni, controllano e modificano le prenotazioni, ecc.).

Il game (1)

Il ***game*** è un'applicazione a sé stante, orientata a presentare l'azienda, offrire momenti di svago e istruzione ai proprietari di animali, e invogliarli ad usare i servizi a pagamento.

- NON è richiesta autenticazione e i servizi sono disponibili a chiunque si colleghi all'applicazione
- Permette agli utenti di descrivere i propri animali con specie, nomi, sesso, età, eventuali condizioni mediche. Non solo cani e gatti!
- Ha un'ampia disponibilità di semplici giochi a tema animale, e pagine informative e di curiosità specifiche sugli animali dell'utente e sul mondo animale in genere. Usate API REST di servizi online, ce ne sono tantissime.
- Ha pagine di visualizzazione dei servizi commerciali offerti da Animal House;
- Ha pagine di visualizzazione dei prodotti di e-commerce acquistabili;
- Permette di passare velocemente all'applicazione di front-office per l'accesso ai servizi e all'e-commerce.

Pensatela come un'app per device mobili, sia online, sia (facoltativamente) installabile.

Il game (2)

Servizi disponibili (usando appositi API*)

- Curiosità sugli animali come i miei
- Curiosità sugli animali in generale
- Informazioni utili sanitarie e legali
- Video buffi ed interessanti da YouTube

Giochi possibili

- ***Quiz***: usando la stessa API delle curiosità il sistema genera domande a caso
- ***Memory***: usando API di immagini il sistema crea coppie di immagini e le dispone nascoste. L'utente deve scoprirle a coppie.
- ***Impiccato***: usando API di dizionari di termini sugli animali il sistema presenta una parola complessa che l'utente deve indovinare lettera per lettera
- ***Scova le differenze***: due immagini simili ma con piccoli particolari differenti che l'utente deve scovare (e.g., usare layer SVG per aggiungere i particolari cliccabili)

(*) **e.g.,** https://zoo-animal-api.herokuapp.com/animals/rand/

Il front-office (1)

Il ***front-office*** è un'applicazione web tradizionale, solo online, sia per device mobili sia per PC, orientata a fornire accesso ai prodotti e ai servizi offerti dall'azienda.

- ***servizi di comunità***: leaderboard con punteggio dei vari giochi del game, bacheca *eccolo qua*, bacheca *cerco partner*, bacheca *aiutatemi*, ecc.
- ***e-commerce***: catalogo ragionato e diviso per sezioni di prodotti per animali acquistabili online: cibo, prodotti sanitari, accessoristica, cuccioli.
- ***servizi in presenza***: veterinario, dog sitter, toelettatura, pensione estiva, psicologo, visita a domicilio per animali soli, ecc. Ogni servizio prevede una parte di illustrazione del servizio e una parte di accesso e prenotazione (secondo sede e disponibilità)
- ***servizi online***: videoconf con l'esperto, con il veterinario, con il proprio animale in ospedale o in pensione, ecc.

Il front-office (2)

I servizi disponibili dipendono dalla sede fisica:

- selezionando la sede ottengo l'elenco dei servizi disponibili, ***oppure***
- selezionando il servizio richiesto ottengo le sedi che lo offrono

I servizi disponibili dipendono dall'intervallo temporale:

- selezionando una data ottengo l'elenco dei servizi disponibili, ***oppure***
- selezionando il servizio richiesto ottengo le date libere.

La disponibilità dei servizi dipende dal numero di risorse disponibili nella sede selezionate e dal tipo e dimensione dell'animale.

La app ha memoria, e si ricorda quanti e quali animali posseggo.

Non solo cani e gatti: pesci, criceti, serpenti, alligatori, cuccioli di wookie, cuccioli di klingon, ecc.

Il back-office

Il *back-office* è la parte dell'applicazione che permette agli amministratori di gestire i dati dei clienti e abilitare e configurare i servizi e i prodotti.

E' un'applicazione web tradizionale, solo online, principalmente per PC.

Servizi:

- **Anagrafica Clienti**: per gestire le informazioni sui clienti: registrazione, login, cambio password, reset password, cancellazione, preferenze e animali preferiti, punteggi dei giochi
- ***Servizi di comunità***: per controllare o cancellare i messaggi nelle bacheche
- **Gestione e-commerce**: per aggiungere e togliere prodotti, prezzi, descrizioni. Ogni prodotto appartiene ad una categoria (e ad una o più sottocategorie) ha sempre un'immagine (con URI online o uploadata sul file system del server)
- ***Servizi in presenza***: per prenotare servizi in presenza, modificare o cancellare prenotazioni, visualizzare disponibilità. Ogni servizio può avere più calendari separati (ad esempio in una sede posso avere molti veterinari, molte "camere" della pensione, ecc.)
- ***servizi online***: per prenotare servizi online, modificare o cancellare prenotazioni, controllare disponibilità

Animal House: i dati

Per quanto riguarda i dati da mantenere nel tempo, l'applicazione deve gestire quanto meno:

- L'anagrafica delle sedi, con elenco dei servizi standard disponibili (negozio, veterinario, pensione, ecc.).
- La gestione dei prodotti, con descrizioni e prezzi e disponibilità.
- L'anagrafica clienti, con ovvie procedure di registrazione, modifica dei dati, cancellazione, ecc.). Importante: preferenze (es. cani, gatti, pesci, ecc.)
- L'organizzazione temporale dei servizi (veterinari, psicologi, "camere" della pensione, ecc.).
- La fatturazione dei servizi e dei prodotti, con gestione di totali e statistiche.

Animal House: servizi extra

Oltre alle funzionalità di semplice manipolazione e presentazione dei dati, l'applicazione dovrà/potrà garantire anche servizi aggiuntivi, come:

- Suggerimenti di alternative
- Riconoscimento delle immagini dell'animale dell'utente
- Suggerimenti di estensioni e prodotti aggiuntivi
- Recensioni
- etc. ...

E' indispensabile che l'utente comprenda senso e funzionamento di questi servizi aggiuntivi e si senta invogliato a farne uso

Requisiti di progetto

- Tutte le parti in nero sono obbligatorie
- Tutte le parti in arancione sono facoltative e generano punteggio extra a discrezione del docente.
- All'atto della presentazione del progetto tutti i database sono già riempiti con un numero ragionevole di utenti, prodotti, servizi, prenotazioni, ecc.
- Il game è realizzato con il framework Javascript e CSS preferito, ed è a scelta separato o integrato con il front-office, ma è visivamente ben distinto da lui.
- Il front-office è realizzato con il framework Javascript e CSS preferito, purché diverso dal game e dal back office.
- Il back office è ***obbligatoriamente*** in Javascript puro con jQuery.
- Tutti i dati locali vengono memorizzati su un DB Mongo sul server del dipartimento.

API

- Di seguito un elenco **non vincolante** di API rilevanti per il progetto
- Siete invitati a trovarne e sperimentarne altre

- Elenco API pubbliche: https://github.com/public-apis/public-apis

- Fatti e immagini di animali: https://zoo-animal-api.herokuapp.com/
- Immagini di cani: https://dog.ceo/dog-api/
- Immagini di pesci: https://www.fishwatch.gov/developers
- Generatore di Meme: https://imgflip.com/api

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Valutazione del progetto

Il lavoro di team

Tutti i membri dei team sono tenuti a lavorare e lavorare insieme.
E' meglio essere parte attiva di un progetto mediocre che passiva di un progetto meraviglioso.
Non saranno tollerati i portatori di pizze
Mi riservo all'esame di scoprire il contributo individuale di ciascuno, indipendentemente dalla bontà del progetto consegnato.

Criteri di valutazione (1)

Criteri di valutazione saranno:

1. la generalità dei tool:
   - quanto le soluzioni per la compatibilità sono forzate e quanto sono frutto di scelte ottimali per framework, organizzazione del codice e uso corretto delle tecnologie disponibili
2. la flessibilità:
   - quanto le soluzioni tecniche adottate sono solide, strutturate, facilmente comprensibili, facilmente estendibili, facilmente adattabili a nuovi device / browser / sistemi operativi / modelli di dati / modelli di annotazione

Criteri di valutazione (2)

3. l'usabilità:
   - Quanta attenzione è data alle esigenze di utenti (sia giocatori, sia clienti, sia dipendenti che usano il back-office) che non conoscono i dettagli del modello di applicazione utilizzata.
4. la sofisticazione grafica
   - Quanta attenzione viene data alla presentazione delle informazioni, al rapporto tra dimensioni delle maschere e dimensioni dei dati da rappresentare, al rapporto tra label comprensibili e dati formalizzati, alla corretta differenziazione nei tipi di dati e di annotazioni.
5. l'accessibilità
   - Tutte le applicazioni sono accessibili e compatibili con gli standard WCAG e ARIA

Vincoli hard

1. Le tre applicazioni sono fatte con tre modelli applicativi diversi: il back-office ***deve*** essere realizzato con Javascript e JQuery. Le altre applicazioni con due framework diversi tra Angular, React, Vue e Svelte.
2. Il framework per la grafica è libero ma mi aspetto sofisticazione grafica, facilità d'uso e accessibilità. Vanno bene sia Bootstrap sia Foundation, ma anche altri a vostra scelta purché compatibili con gli altri vincoli.
3. Il deploy ***deve*** avvenire su uno/due container docker sulle macchine del dipartimento.
4. Ogni applicazione che usi forme di packing o compilazione (inclusi webpack e typescript) ***deve*** avere i sorgenti caricati nella directory *source* dello spazio web)
5. Tutte le applicazioni devono essere accessibili. L'accessibilità viene verificata dal dott. Rubano o in precedenza (su appuntamento) oppure il giorno stesso della presentazione.
6. Tutti i database vengono presentati già popolati.

Criteri di valutazione (3)

- Lo scritto pesa l'70% del voto finale
- Il progetto base pesa il 35% del voto finale
- L'aggiunta di funzionalità facoltative porta il peso fino al 40% del voto finale, a discrezione del docente.
- In casi eccezionali (tutte le funzionalità facoltative, servizi di AI integrati, ulteriori funzionalità non previste in queste specifiche) si può arrivare fino al 45% del voto finale, a discrezione del docente.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il contributo individuale

- Ogni membro di ogni team deve dimostrare di aver contribuito in maniera determinante alla realizzazione del progetto.
- Ad inizio della presentazione ogni membro dichiara che cosa ha realizzato, e il docente, in totale autonomia, decide se questo contributo è o non è sufficiente.
- Realizzare solo HTML e CSS non è sufficiente.
- Realizzare parti marginali del codice (login, logout, lettura delle preferenze, ecc.) non è sufficiente
- Un candidato ideale si è occupato sia della parte HTML/CSS, sia della parte di programmazione, sia client sia server.
- La distribuzione ideale dei compiti è funzionale e non architetturale.

Il lavoro di team

- Tutti i membri dei team sono tenuti a lavorare e lavorare insieme.
- E' meglio essere parte attiva di un progetto mediocre che passiva di un progetto meraviglioso.
- Non saranno tollerati i portatori di pizze
- Mi riservo all'esame di scoprire il contributo individuale di ciascuno, indipendentemente dalla bontà del progetto consegnato.

Suggerimenti per l'esame

- Venite alla presentazione con il progetto che funziona. Se non va io vi faccio tornare. Per questo preferisco
  - vedervi una settimana dopo l'appello con il progetto che funziona
- piuttosto che
  - perdere tempo con la presentazione di un progetto che non va,
  - mandarvi via in lacrime, e
  - vedervi una settimana dopo l'appello con il progetto che funziona
- Venite allo scritto avendolo preparato. Non c'è niente di più irritante di vedere ragazzi svegli e competenti (vi si riconosce) che prendono 8 o 10 allo scritto perché ci hanno solo provato.
- Lo scritto non è difficile per chi ha studiato, è impossibile per chi non l'ha fatto.

Rigidità del corso
(nessuna eccezione per nessun motivo)

- Il progetto deve funzionare. Completamente ed esattamente.
- Il progetto deve risiedere su una macchina del dipartimento. Questo include SIA il codice SIA tutti i dati del progetto (tranne quelli per cui è previsto l'uso di un server condiviso)
- Se volete installare librerie e SW speciali per il progetto, verificate prima che questo sia possibile sulle macchine e sui sistemi operativi offerti dal dipartimento
- Il progetto deve venire presentato da tutto il gruppo insieme. In nessun caso è accettabile che si presenti in una data una parte del gruppo e in una data diversa gli altri.

Flessibilità del corso

- Prova scritta e prova di progetto sono indipendenti.
  - Il progetto è sempre di gruppo
  - Lo scritto è sempre individuale
- Potete provare lo scritto tutte le volte che volete
  - Il voto precedente verrà cancellato solo se consegnate un nuovo scritto
  - Gli scritti sono solo alle date degli appelli ufficiali
- Potete presentare il progetto tutte le volte che volete
  - Solo se lo decide consensualmente TUTTO IL GRUPPO
  - Potete ritirarvi dalla presentazione del progetto in qualunque momento e tornare una settimana o due dopo con le correzioni che ritenete opportune.

L'appello di febbraio

- Quanto detto prima NON si applica all'appello di febbraio.
- il 28 febbraio 2023 si conclude la possibilità di presentare il progetto di quest'anno.
- Gli slot a disposizione per presentare il progetto a febbraio non sono infiniti. Tutti gli anni questi slot (più del doppio degli altri appelli) si esauriscono molto presto.
- Non riducetevi all'ultimo, cercate di portare il progetto negli appelli estivi ed autunnali
- Io cerco di ***non*** essere più esigente a febbraio, ma inevitabilmente potrò dedicare meno attenzione al vostro meraviglioso progetto.
- Non riducetevi all'ultimo

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Attenzione!

Vi ho già detto di non ridurvi all'ultimo?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

**Fabio Vitali, Angelo Di Iorio,
Francesco Sovrano, Vincenzo Rubano**

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

fabio.vitali@unibo.it
francesco.sovrano2@unibo.it

www.unibo.it