Specifiche dei progetti per l'esame di *Applicazioni dinamiche per il Web* per l'anno accademico 2023/2024

Aggiornato al 13 dicembre 2024

1 Progetto per gli appelli di Settembre 2024 e Febbraio 2025

1.1 Premessa

Questo progetto è di tipo leggermente diverso perché chiede di sviluppare un sito che potrebbe essere il prototipo di un servizio reale.

Il centro *Multidisciplinare Infezioni Sessualmente Trasmesse (MISTRA)* è un servizio dell'Azienda Ospedaliera Universitaria Integrata di Verona che, a titolo gratuito, offre un servizio di prevenzione, diagnosi e terapia circa le infezioni sessualmente trasmesse (IST).

Il centro ha un sito web d'informazioni, `https://www.centromistra.com/`, realizzato usando la piattaforma WiX (`https://it.wix.com/`).

Per diverse ragioni, il sito deve essere migliorato per permettere agli utenti di non solo leggere le informazioni già presenti ma anche di verificare il proprio livello di conoscenza circa alcuni aspetti delle IST tramite dei *test*/*quiz* anonimi.

L'idea quindi è aiutare il centro MISTRA (che non ha fondi per commissionare un'estensione del sito) offrendo un prototipo basato su REACT che replichi l'attuale struttura e che offra anche la funzionalità di gestire dei test.

Considerato inoltre che si vuole permettere ai medici di poter aggiungere documenti/pagine in autonomia, si propone quindi si realizzare un'applicazione web dinamica basata sulla piattaforma STRAPI (`https://docs-next.strapi.io/dev-docs/plugins`).

1.1.1 Breve introduzione a STRAPI

Strapi (`https://strapi.io/`) è un framework di CMS (Content Management System) costruito interamente in Node.js. Strapi è *headless* che significa che il front-end e il back-end sono completamente separati. Supporta vari tipi di database, sia SQL (MySQL, PostgreSQL, SQLite) che NoSQL (MongoDB). I contenuti possono essere serviti tramite API RESTful o GraphQL. Strapi genera automaticamente API RESTful per i modelli di contenuto definiti, offrendo anche il supporto per GraphQL per query più flessibili e ottimizzate.

Caratteristiche Principali

- **Customizzazione**: Strapi permette una grande flessibilità nella personalizzazione di modelli di contenuto, con un builder di contenuti drag-and-drop che consente di creare modelli senza scrivere codice.
- **Autenticazione e Autorizzazione**: Include sistemi di autenticazione e autorizzazione predefiniti, con supporto per JWT (JSON Web Tokens) e OAuth, consentendo una gestione sicura degli utenti e dei permessi.
- **Sistema di Plugin**: Strapi supporta un sistema di plugin che permette di estendere le funzionalità del CMS. Esistono numerosi plugin ufficiali e di terze parti disponibili.
- **Upload di File**: Gestisce l'upload di file con supporto per vari provider di storage come Amazon S3, Cloudinary, e altri.
- **Internationalization (i18n)**: Supporta la gestione dei contenuti multilingue, permettendo di creare versioni localizzate dei contenuti.
- **Role-Based Access Control (RBAC)**: Fornisce un sistema di controllo degli accessi basato su ruoli per gestire i permessi granulari per diversi utenti e gruppi.

Interfaccia Utente

- **Admin Panel**: Strapi offre un pannello di amministrazione intuitivo, sviluppato con React, che consente agli utenti non tecnici di gestire contenuti, utenti e configurazioni del sito.
- **Content Types Builder**: Uno strumento visuale per creare e gestire tipi di contenuto e loro attributi senza necessità di scrivere codice manualmente.

DevOps e Deployment

- **Configurazione**: Le configurazioni di Strapi possono essere gestite attraverso file di configurazione o variabili d'ambiente, facilitando l'integrazione con vari ambienti di sviluppo e produzione.
- **Scalabilità**: Grazie alla sua natura headless e alla costruzione in Node.js, Strapi è altamente scalabile e può gestire grandi volumi di traffico e dati.
- **Deployment**: Può essere distribuito su vari servizi di hosting, inclusi Heroku, AWS, DigitalOcean, e altri. Inoltre, grazie alla sua architettura basata su container, è facile da distribuire utilizzando Docker e Kubernetes.

2 Specifiche del progetto

Il progetto quindi può essere diviso in due parti: una di semplice configurazione e una di sviluppo.

Parte di configurazione Si deve realizzare un sito web (locale) basato su STRAPI copiando le pagine dell'attuale sito MISTRA. Essendo STRAPI un (headless) Content Management System (CMS), le pagine di Mistra saranno inseriti come articoli/documenti in STRAPI.

L'organizzazione delle pagine pubbliche del sito dovrà essere circa uguale a quella del sito MISTRA attuale. Se si trovano soluzione esteticamente migliori, sono ben accette.

La parte accessibile dal pubblico dovrà essere validata secondo l'accessibilità AA WCAG 2.1. Visto che il sistema è un CMS, non si può garantire l'accessibilità AA per le pagine che verranno inserite in futuro. Quello che si chiede è che i menù, eventuali immagini e lo scheletro del sito pubblico siano accessibili secondo il livello AA.

Parte di sviluppo Si deve realizzare un'applicazione web dentro il framework STRAPI (può essere trattata come un plugin) che realizza la funzionalità dei test.
Definizione di *test*:

1. Un quiz è composto da una domanda e un numero variabili di risposte possibili. Il testo della domanda deve avere formato HTML (non si ponga il limite di lunghezza). Ciascuna risposta deve avere formato HTML (non si ponga il limite di lunghezza).
2. Ogni risposta ha associato: un valore decimale tra 0 e 1 (0=significa errato, 1=significa corretto) e un campo `spiegazione` (formato HTML) che sarà usato per spiegare l'errore in caso in cui la risposta non è corretta.
3. L'associazione domanda-riposta è 1-a-n per rendere la gestione da parte dei medici più semplice.
4. Un quiz è associato n-a-1 a una *categoria*. Le categorie servono per classificare i quiz in base al tema trattato.
5. Un test è specificato da una sequenza di quiz. Una quiz può essere usato in diversi test.
6. Test, quiz, domande, categorie e risposte sono memorizzate su un database SQL. Per comodità di propone lo schema logico più avanti.
7. Ci dovrà essere infine una tabella dove si salvano i risultati di un'esecuzione di un test. Per ogni esecuzione, si salverà: id del test, per ciascuna domanda, la risposta data, giorno e orario d'inserimento del quiz, codice utente (vedi sotto), sesso ed età dell'utente.

Il back-end dovrà permettere di gestire i test e dove inserire questi test nel sito (parte del CMS). Dovrà inoltre permettere al medico di accedere alla tabella dei test fatti per poter recuperare le risposte date a un test (l'idea è che un utente si presenti con il codice del test fatto). Il medico deve quindi poter indicare che ha visionato l'esecuzione del test.

Il front-end dovrà presentare una pagina con un test (scelto in modo casuale fra quelli presenti nel database), chiedere età e sesso all'utente, generare un codice di esecuzione del test casuale (per esempio: data formato iso+3lettere) e presentare il test. Una volta ottenute tutte le risposte e che l'utente conferma l'invio, deve salvare il risultato e indicare all'utente il codice dell'esecuzione del test assegnato al test (agli utenti NON si può chiedere nulla oltre età e sesso). Deve quindi mostrare per ogni domanda, la risposta data e, se errata, la spiegazione associata. NON deve indicare la risposta esatta. Deve poi indicare il punteggio raggiunto stile '3.5/10'.
Infine, deve dare la possibilità di salvare il risultato come pagina PDF.

Schema Logico Il seguente schema logico riporta i nomi delle tabelle e degli attributi che si devono usare. Il tipo di ciascun attributo è specificato nel commento se non è immediatamente ricavabile.

Si consideri lo schema come una proposta che può essere completata aggiungendo attributi. Si chiede di mantenere i nomi assegnati. Gli attributi sottolineati sono chiave della tabella. Gli attributi `id_<nome>` sono chiave esterne e il nome indica al tabella.

```
1  Question(id, name, texto, id_category)
2  Answer(id, texto, score, correction, id_question)
3  Category(id, name)
4  Test(id, name, description)
5  QuestionInTest(id_question, id_test)
6  TestExecution(id, execution_time, age, id_sex, id_test, score, IP, revision_date,
     note)-- id dovrà essere comunicato all'utente, execution_time è un timestamp, IP è
      l'indirizzo IP, score è il totale (per evitare di calcolare ogni volta),
     checked_date è il timestamp di quando un medico visiona le risposte e note è
     eventuale commento inserito dal medico (dovrà esserci un tasto 'Visionato' che
     inserisce la data e la nota).
7  Sex(id, name)-- lasciamo ai medici definire i possibili valori
8  GivenAnswer(id_testExecution, id_answer)
```

Ulteriori richieste di progetto

1. Il gruppo di sviluppo **deve** essere di 2-3 studenti in cui le responsabilità di progetto devono essere ben definite. Usate il forum del sito moodle per cercare un collega di sviluppo.
2. L'applicazione deve essere accessibile anche via tablet o smartphone (usare CSS per rendere responsive).

Modalità di presentazione all'esame

- Preparare massimo 2-3 slide in cui si spiegano i criteri/principi decisi per lo sviluppo dell'applicazione.
- Preparare massimo 2-3 slide in cui si mostra come si è organizzato STRAPI per la parte CMS.
- Preparare massimo 2-3 slide in cui si mostra come si è sviluppato il plugin (o estensione) per il quiz.
- Preparare 2 in cui si mostra il processo di validazione AA WCAG 2.1 (che strumenti si sono scelti, che prove si sono fatte). Queste prove si devono poi mostrare dal vivo.
- È necessario saper dimostrare di essere in grado di presentare/modificare qualsiasi riga del codice sorgente a indicazione del docente.

Ricordo, infine, le raccomandazione fatte alla presentazione del corso:

- Il progetto deve funzionare esattamente come da specifiche
- Il progetto deve essere sviluppato su un proprio PC (per chi non ha PC, ci sono i PC dei laboratori)
- Il progetto deve venire presentato da tutto il gruppo insieme.
- Non sono ammesse tecnologie che richiedano compilazioni o procedure complesse di aggiornamento del codice eseguito. Deve essere possibile aprire un file sul server e modificarlo al volo durante la prova.

Ulteriori informazioni

Se sono necessarie ulteriori specifiche, **non** mandate una email, ma scrivete le forum del sito moodle del corso. In questo modo le informazioni sono condivise tra tutti risparmiando tempo mio e vostro. Sono ammessi anche scambi di idee durante lo sviluppo.