Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

Progettazione e sviluppo di una piattaforma web personalizzata per la gestione dei contenuti

Relatore:
Prof. Giovanni Delnevo

Presentata da:
Gaia Mazzoni

Correlatori:
Prof. Silvia Mirri
Dott. Manuel Andruccioli

Sessione III
Anno Accademico 2023-2024

Dedico la mia tesi di laurea alla mia famiglia e ai miei amici,
che mi sono stati accanto durante tutto il percorso,
sostenendomi sempre.

Introduzione

Negli ultimi anni, la gestione e la divulgazione di contenuti digitali sul web hanno assunto un ruolo sempre più centrale in numerosi ambiti, dalla divulgazione culturale e scientifica alla gestione di archivi e risorse online. Con la crescita del numero di informazioni disponibili e l'evoluzione delle tecnologie web, si è resa necessaria la creazione di piattaforme che consentano una gestione dinamica e strutturata dei contenuti, garantendo al contempo una navigazione intuitiva e un'esperienza utente efficace.

In questo panorama tecnologico in continuo cambiamento l'adozione di soluzioni modulari e scalabili per la gestione dei contenuti digitali rappresenta una necessità crescente. Mentre i Content Management System (CMS) tradizionali offrono strumenti versatili per la creazione e la pubblicazione di contenuti, essi presentano anche limiti in termini di personalizzazione e ottimizzazione per specifiche esigenze.

É in questo contesto che si sviluppa il progetto illustrato in questa tesi. Il progetto riguarda la progettazione e lo sviluppo di una piattaforma personalizzata per la gestione dei contenuti digitali dell'Istituto Storico della Resistenza e dell'Età Contemporanea di Forlì-Cesena, con obiettivo il miglioramento dell'organizzazione e della presentazione dei contenuti già da loro visualizzati in una serie di siti web. La piattaforma web dovrà accorpare in sè i quattro siti attualmente in uso dall'Istituto Storico, permettendo anche la gestione e la visualizzazione di diverse tipologie di pagine, in modo da supportare la varietà di risorse offerte dall'Istituto. Per questo motivo, si è deciso di realizzare una soluzione personalizzata ispirata ai principi di un CMS, ma

progettata su misura per le specifiche esigenze dell'Istituto. In particolare, il progetto riportato in questa tesi si è concentrato sull'implementazione di una base solida per il nuovo sito, con focus sul front-end.

Al termine del progetto, il risultato ottenuto è una piattaforma web con funzionalità simili a quelle di un CMS *headless*, che permette agli utenti di visualizzare informazioni di varia natura in un'interfaccia grafica semplice ed intuitiva. La struttura del sito è modulare e organizzata per mantenere la separazione dei ruoli, in modo da supportare successive modifiche ed espansioni. Inoltre, alle spalle del sito è stato realizzato un database centralizzato, in grado di mappare qualsiasi suo contenuto, e di fornire le informazioni necessarie alla visualizzazione dinamica dei contenuti delle pagine.

La tesi è strutturata in tre capitoli:

1. Contesto:
   Nel primo capitolo, viene approfondito il contesto in cui si sviluppa la tesi, partendo dalle definizioni di "contenuto" e di Content Management System. Il capitolo prosegue illustrando il processo di sviluppo tecnologico che ha portato alla nascita dei CMS, ed esplorando vantaggi e limitazioni derivanti dal loro impiego. A seguire, viene data una panoramica sulle tipologie di CMS, facendo la distizione tra CMS statici e dinamici e definendo le quattro architetture principali utilizzate nei CMS. Per finire, vengono analizzati i principali CMS presenti sul mercato e viene contestualizzato il caso di studio dell'Istituto Storico, illustrando i quattro siti gestiti dall'Istituto ed i loro contenuti.

2. Tecnologie utilizzate:
   Nel secondo capitolo, vengono illustrate tutte le tecnologie utilizzate durante la realizzazione del progetto, esaminandone le caratteristiche principali. Vengono presi in considerazione, all'interno del capitolo, linguaggi di programmazione, ma anche framework ed API adottati, dando una solida base tecnica per il capitolo successivo.

3. Progetto:

   Nel terzo ed ultimo capitolo, viene esplorato il processo di realizzazione del progetto. In un primo momento, viene illustrato lo stato attuale delle piattaforme web dell'Istituto Storico, evidenziando le criticità presenti nelle soluzioni attuali, e le tecnologie attualmente adottate. Si passa poi alla definizione dei requisti funzionali e non funzionali, sia a livello grafico che implementativo, che forniscono una base di partenza per le successive fasi di progettazione. Il capitolo prosegue con la progettazione delle interfacce, illustrando il percorso che ha portato dall'analisi dei requistiti alle scelte grafiche e alla realizzazione dei mockup. Una volta definiti i mockup, viene progettato e sviluppato un database unico e centralizzato capace di mappare tutti i contenuti gestiti dall'Istituto Storico. Infine, viene illustrato lo sviluppo della piattaforma web vera e propria, con particolare attenzione alla struttua del progetto e all'architettura utilizzata. Il capitolo si chiude con la descrizione delle interfacce grafiche realizzate.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Contesto

In questo capitolo, viene esplorato il panorama dei CMS, strumenti che hanno rivoluzionato il modo di organizzare e gestire i contenuti digitali. Verrà data una definizione di CMS e successivamente si analizzeranno i loro vantaggi e le loro limitazioni, le diverse tipologie esistenti e le architetture possibili.

Dopo una panoramica generale, verranno analizzati i CMS statici e dinamici, e successivamente, verranno illustrati i principali CMS disponibili sul mercato, esaminandone funzionalità, caratteristiche principali e punti di forza.

Infine, verrà introdotto il caso di studio dell'Istituto Storico di Forlì-Cesena, un'associazione che utilizza diversi siti web con specifiche esigenze. Verranno illustrati i suoi siti attualmente in uso, le loro funzioni e le sfide rappresentate dalla gestione e dall'aggiornamento dei loro contenuti.

1.1 Content Management System

I CMS (Content Management System) sono software o applicativi che permettono di automatizzare la gestione di contenuti digitali, semplificando operazioni come la creazione, la modifica e la pubblicazione di informazioni.

1.1.1 Definizione di "contenuto"

Per comprendere a fondo il ruolo dei CMS, è utile definire innanzitutto cosa si intende per *contenuto*. Un contenuto può essere descritto come un insieme di informazioni strutturate attraverso un processo editoriale, il cui scopo è la preparazione e la rifinitura di dati affinchè possano essere pubblicati e fruiti da un pubblico. Questo processo include attività come la modellazione, la modifica, il controllo e l'approvazione dei dati.

Un esempio tipico di creazione di contenuti è la scrittura di un articolo di giornale: la sua stesura è influenzata dalla soggettività dell'autore, e può richiedere diverse revisioni prima che l'articolo sia pronto ad essere pubblicato. Il processo di creazione di contenuti è spesso iterativo, caratterizzato da continue modifiche e miglioramenti, e può essere interrotto e ripreso in qualsiasi momento. In generale, il fine principale della produzione di contenuti è la fruizione di questi da parte di esseri umani.

Al termine di un'attenta analisi, Deane Barker, autore di "Web Content Management: Systems, Features, and Best Practices" fornisce la seguente definizione teorica di "contenuto": "Un contenuto è l'informazione prodotta attraverso il processo editoriale e finalizzata ad essere consumata da un pubblico tramite la pubblicazione" (Barker, 2017).

1.1.2 La spinta allo sviluppo dei CMS

L'esigenza di avere sistemi per la gestione di contenuti è sempre esistita, ma con l'avvento del World Wide Web (WWW) nei primi anni '90 questa necessità è diventata ancora più urgente. La nascita del web ha permesso a chiunque di creare e pubblicare contenuti digitali, rendendoli accessibili a un pubblico globale. Con il proliferare dei siti web però, emerge una nuova sfida: la gestione efficiente delle informazioni.

Inizialmente, la creazione di contenuti digitali avveniva principalmente tramite l'uso di editor testuali per realizzare pagine HTML statiche. Questa modalità presentava diverse problematiche: la facilità di commettere errori

nel codice, la scarsa manutenibilità dei contenuti e la difficoltà nel separare la struttura grafica della parte testuale e contenutistica. Inoltre, la gestione di un sito web si basava interamente su file, che risultavano fragili e difficili da gestire, soprattutto quando erano necessarie modifiche frequenti.

Come accennato in precedenza, la creazione di contenuti digitali, soprattutto nei primi anni del web, avveniva principalmente attraverso l'uso di editor di testo e la scrittura di pagine HTML statiche. Sebbene questo approccio fosse inizialmente funzionale, le difficoltà legate alla gestione dei contenuti, alla loro manutenibilità e alla separazione tra struttura e contenuto, spingevano alla ricerca di soluzioni di maggiore efficienza. Questo ha portato alla nascita ed all'evoluzione dei Content Management System.

1.1.3 Definizione di CMS

Espandendo i concetti accennati in precedenza, un CMS è un software server-based e multiutente, progettato per interagire con i contenuti memorizzati in un repository, che può risiedere sullo stesso server del CMS oppure su un server separato accessibile tramite rete. La sua funzione è principalmente di semplificare e automatizzare il processo di creazione, modifica, gestione e pubblicazione dei contenuti digitali. Grazie ai CMS, i creatori di contenuti digitali possono concentrarsi sulla sostanza, senza preoccuparsi di scrivere codice complesso o laborioso, e senza dover gestire manualmente file HTML, eliminando molte delle difficoltà legate alla gestione di un sito web. Inoltre, i CMS si occupano non solo dell'organizzazione e gestione di contenuti, ma anche della loro protezione, attraverso diversi meccanismi:

- Gestione dei permessi, per regolare chi può visualizzare e modificare un contenuto.
- Gestione dello stato e del frusso di lavoro, per definire se un contenuti è stato pubblicato o è solo una bozza.
- Versionamento, per verificare come e quante volte un contenuto è stato modificato.

Un CMS è composto da diverse parti fondamentali, che collaborano per garantire un flusso di lavoro editoriale fluido ed efficiente. Tra queste vi sono:

- Interfaccia per l'editing, che permette agli utenti di creare e modificare contenuti in maniera intuitiva, con l'ausilio di strumenti visivi che non richiedono conoscenze nell'ambito della programmazione.
- Repository: è lo spazio dove vengono memorizzati i contenuti. Può essere all'interno di un server locale o di un sistema distribuito.
- Meccanismi di pubblicazione: sono funzionalità che permettono di preparare i contenuti alla pubblicazione su un sito, gestendo anche le versioni e gli accessi.

Quindi in sintesi, un CMS non è solo un software per gestire contenuti, ma è anche uno strumento che aiuta a mettere in pratica una versione teorica di gestione dei contenuti, ottimizzando l'intero processo e riducendo al minimo gli errori umani.

1.2 Vantaggi e limitazioni dell'utilizzo di CMS

L'adozione di un CMS porta numerosi vantaggi nella gestione dei contenuti digitali, migliorando l'efficienza del lavoro editoriale e la qualità delle pubblicazioni. Grazie alle loro funzionalità avanzate, i CMS offrono un controllo più strutturato sui contenuti e semplificano la gestione delle informazioni, rendendole più accessibili e organizzate.

1.2.1 Vantaggi

Tra i vantaggi vi sono:

- Controllo avanzato sui contenuti: i CMS permettono di tenere traccia dei contenuti, identificandone la posizione, il loro stato di pubblicazione (ad esempio "bozza", "pubblicato") e chi vi può accedere. Inoltre,

garantiscono l'integrità dei contenuti tramite meccanismi di gestione dei permessi, che si occupano di limitare l'accesso in visualizzazione e modifica ai contenuti, gestiscono lo stato e il *versioning*, ovvero il tracciamento delle modifiche e l'eventuale rispristino di versioni precedenti, oltre a gestire le dipendenze tra contenuti.

- Riutilizzo dei contenuti: grazie alle funzionalità dei CMS, è possibile riutilizzare contenuti con facilità, riprendendoli in più pagine o sezioni del sito, evitandoduplicazioni. Questa funzionalità riduce notevolmente il rischio di errori che potrebbero invece verificarsi con una gestione manuale del riuso di contenuti.

- Automazione e aggregazione di contenuti: un CMS permette la creazione automatica di elenchi, menu di navigazione e raccolte di contenuti, facilitando la fruizione di questi da parte dell'utente.

1.2.2 Limitazioni

Nonostante i CMS offrano molte funzionalità avanzate, presentano anche alcune limitazioni:

- Rigidità nella personalizzazione: Molti CMS offrono modelli e strutture predefiniti che facilitano il lavoro editoriale, ma allo stesso tempo possono risultare limitanti per esigenze particolarmente complesse. Personalizzare un CMS in modo avanzato può richiedere modifiche al codice o l'uso di plugin di terze parti, aumentando la complessità del progetto.

- Formattazione efficace non garantita: Se l'editor ha il controllo completo sulla formattazione del testo, possono verificarsi usi incoerenti di stili, grassetti, corsivi, allineamenti, hyperlink e immagini. Per questo motivo, molti CMS implementano sistemi di formattazione predefiniti per mantenere l'uniformità visiva delle pubblicazioni.

1.3 Tipolgie di CMS

Esistono diverse tipologie di CMS, ciascuna con caratteristiche specifiche che li rendono adatti a determinati scopi. I quattro principali tipi di CMS sono: Web Content Management (WCM), Enterprise Content Management (ECM), Digital Asset Management (DAM) e Records Management (RM). Ognuno di questi è progettato per gestire diversi tipi di contenuti e flussi di lavoro, e sono tutti fondamentali in ambiti specifici.

1.3.1 Web Content Management

Il WCM è la tipologia di CMS che gestisce contenuti destinati alla pubblicazione su Internet, con particolare attenzione alla separazione tra contenuti e grafica. I WCM sono progettati per gestire contenuti ideati per un pubblico di massa, e sono tipicamente utilizzati da organizzazioni che hanno bisogno di pubblicare regolarmente su siti web, blog e altre piattaforme online.

Un aspetto fondamentale dei WCM è che separano la gestione del contenuto dalla parte visiva, ovvero il design del sito web. Ciò consente una gestione più efficiente, in quanto i contenuti possono essere facilmente aggiornati senza influire sulla grafica o sul layout della pagina. Inoltre, i WCM sono solitamente ottimizzati per la pubblicazione su più canali, come desktop, mobile e tablet, e a volte possono integrarsi con i social media o altre piattaforme di distribuzione, garantendo la presenza coerente dei contenuti su tutti i canali [3].

1.3.2 Enterprise Content Management

L'ECM, precedentemente noto come Document Management (DM), è un tipo di CMS specializzato nella gestione di documenti di un'azienda. I contenuti gestiti in un ECM sono generalmente documenti aziendali, come relazioni, report, curriculum, contratti e altre informazioni che necessitano di una gestione accurata e sicura.

Gli ECM si distinguono per la loro capacità di gestire flussi di lavoro complessi, in particolare per quanto riguarda la collaborazione tra i membri di un team, il controllo degli accessi e la gestione delle versioni dei documenti. Gli ECM supportano anche la conservazione e la ricerca dei documenti, garantendo che le informazioni siano facilmente accessibili, protette e conformi alle normative aziendali.

1.3.3 Digital Asset Management

Il DAM si concentra sulla gestione di risorse digitali ricche, come immagini, video, audio e altri file multimediali. Questi sistemi sono particolarmente utilizzati da aziende che producono o utilizzano un numero elevato di contenuti multimediali, come agenzie di marketing, studi fotografici, case editrici e aziende di produzione video.

I DAM eccellono nella gestione dei metadati e nel *renditioning* (ovvero la creazione di diverse versioni di un file digitale per usi diversi). Questi sistemi permettono agli utenti di archiviare, organizzare, cercare e recuperare facilmente file multimediali. Inoltre, i DAM consentono di manipolare e modificare gli asset digitali direttamente all'interno del sistema, senza la necessità di software esterni.

1.3.4 Records Management

Il RM è un tipo di CMS che si occupa della gestione dei record aziendali e di altri documenti transazionali generati durante le operazioni di business. I sistemi RM sono particolarmente utili per la gestione e la conservazione di dati storici, come contratti, fatture, corrispondenza e altri documenti legali o finanziari.

L'obiettivo principale dei sistemi di Records Management è garantire che i documenti siano conservati in modo sicuro e conforme alle normative legali, gestendo il ciclo di vita dei documenti stessi, dalla creazione alla distruzione.

Inoltre, questi sistemi consentono di avere un'accurata tracciabilità e gestione degli accessi alle informazioni sensibili [4].

Da questo punto in avanti, si farà riferimento con il termine CMS unicamente ai Web Content Management System, ovvero quei sistemi specificamente progettati per la gestione e pubblicazione di contenuti sul web.

1.4 CMS statici e dinamici

Un CMS è un sistema che gestisce i contenuti di un sito. Il modo in cui questi contenuti vengono pubblicati dipende dalla natura del sito che si vuole ottenere, che può essere statica o dinamica.

1.4.1 CMS Statici

I CMS statici generano pagine HTML statiche in anticipo, e le distribuiscono senza bisogno di un database o un motore server-side. Infatti, nei siti statici, ogni pagina è rappresentata da un file HTML a sè stante, creato manualmente o generato in anticipo dal CMS. Nonostante i siti statici siano apprezzati per la velocità di accesso alle pagine, senza dover attendere alcun tempo di caricamento, e per la maggiore sicurezza data dalla mancanza di un database alle spalle, non offrono la possibilità di personalizzare i contenuti per l'utente e sono difficili da aggiornare e gestire.

1.4.2 CMS Dinamici

I CMS dinamici invece utilizzano un database e generano le pagine al momento della richiesta da parte dell'utente, offrendo maggiore interattività. I siti web dinamici non hanno infatti pagine HTML fisse, ma generano i contenuti delle pagine in tempo reale, in base alle richieste dell'utente, recuperando e assemblando le informazioni prese dal database. I siti dinamici offrono maggiore flessibilità e personalizzazione dei contenuti, che vengono

adattati in base alle preferenze dell'utente. Inoltre, semplificano l'automazione dei contenuti, grazie al database centralizzato che facilita la modifica e l'aggiornamento delle pagine, e permette l'utilizzo di funzionalità come filtri e transazioni. Tuttavia, i siti dinamici richiedono una complessità tecnica maggiore, a causa delle interrogazioni al database e l'elaborazione lato server, che causa un aumento dei tempi di caricamento, influenzando così le prestazioni [5][6].

1.5 Architettura di un CMS

L'architettura di un CMS definisce il modo in cui gli strumenti e le tecnologie interagiscono per gestire i contenuti sia sul back-end, dove vengono creati ed editati, sia sul front-end, dove vengono visualizzati dagli utenti. Esistono diverse architetture di CMS, ognuna con vantaggi e limitazioni a seconda delle esigenze del progetto.

1.5.1 CMS Tradizionale (Coupled)

Un CMS tradizionale, noto anche come coupled CMS, presenta un'architettura in cui il front-end e il back-end sono strettamente collegati. Questo modello è caratterizzato da quattro elementi fondamentali:

- Database per l'archiviazione di contenuti e risorse digitali.
- Piattaforma di gestione dei contenuti (back-end) dove gli editor creano e modificano i contenuti.
- Sistema di gestione del design per applicare template e stili grafici ai contenuti (back-end).
- Interfaccia di pubblicazione (front-end) che visualizza i contenuti attraverso pagine HTML.

Nei coupled CMS, il backend si occupa non solo della gestione dei contenuti, ma anche della loro presentazione, generando direttamente le pagine HTML

che vengono servite agli utenti Questo tipo di CMS è ideale per siti web personali o di piccole aziende, poichè offre un'implementazione semplice e immediata. Tuttavia, è progettato principalmente per la pubblicazione di contenuti sul web e ha una limitata capacità di riutilizzare i contenuti su più piattaforme o dispositivi.

1.5.2 CMS Decoupled

Un CMS decoupled separa il front-end dal back-end, mantenendo comunque un livello di connessione tra i due. La sua architettura comprende:

- Un database per l'archiviazione dei contenuti.
- Una piattaforma di gestione per la creazione dei contenuti.
- Una piattaforma di pubblicazione predefinita per la distribuzione dei contenuti.
- API che collegano il back-end con il front-end.

Questo tipo di CMS offre maggiore flessibilità, dal momento che modifiche al back-end non impattano il front end, e diminuisce le dipendenze tra sviluppatori e pubblicatori di contenuti. Offre inoltre maggiore sicurezza, essendo meno esposto ad attacchi informatici, e migliore performance nella condivisione di contenuti grazie all'uso di API.

1.5.3 CMS Headless

Un CMS headless elimina qualsiasi livello di presentazione predefinito, contribuendo ad una maggiore separazione tra front-end e back-end. In questa architettura, il back-end gestisce solo la creazione e l'archiviazione dei contenuti, mentre le API distribuiscono i dati ai vari canali, come web, mobile, social media. Gli sviluppatori possono inoltre scegliere in autonomia il framework e le tecnologie da utilizzare per il front-end. La sua architettura comprende:

- Un database per l'archiviazione dei contenuti.
- Una piattaforma per la gestione e la creazione dei contenuti.
- Un front-end indipendente che recupera i contenuti tramite API e li visualizza secondo le logiche definite dagli sviluppatori.
- API che connettono il back-end con il front-end.

I CMS headless offrono la possibilità di pubblicare contenuti su più canali con facilità, e permettono ai creatori di contenuti di scegliere le tecnologie front-end in autonomia. Tuttavia, i loro meccanismi e la loro gestione tendono ad essere complessi per gli autori che non possiedono competenze tecniche avanzate [7].

1.5.4 CMS Ibrido

Un CMS ibrido combina i vantaggi di un CMS headless con le funzionalità e la personalizzazione dei CMS tradizionali, permettendo di distribuire i contenuti su più canali, personalizzare l'esperienza utente e offrire strumenti user-friendly come editor WYSIWYG. In questo modo, bilancia flessibilità e controllo della pubblicazione, supportando la personalizzazione e l'analisi dei contenuti e offrendo un'esperienza più accessibile per gli autori. Tuttavia, pur essendo più intuitivo rispetto a un CMS headless, rimane più complesso da implementare e da utilizzare rispetto alla sua controparte tradizionale o decoupled [8].

La tabella 1.1 riepiloga le differenze fondamentali tra questi quattro tipi di CMS.

Tipo di CMS	**Chi genera l'HTML**	**Può servire API**	**Frontend indipendente**
Coupled CMS	back-end	No	No
Decoupled CMS	back-end	Sì	Parzialmente
Headless CMS	front-end	Sì	Sì
CMS Ibrido	back-end e/o front-end	Sì	Sì

Tabella 1.1: Confronto tra le diverse architetture CMS

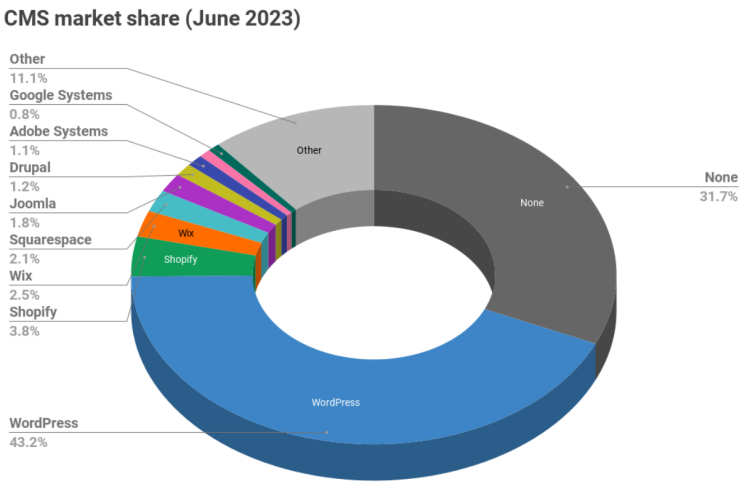

Figura 1.1: Suddivisione del mercato dei CMS. [1]

1.6 Panoramica sui CMS più diffusi

Nel panorama dei CMS esistono numerose soluzioni, ognuna con caratteristiche specifiche pensate per diversi tipi di utenti ed esigenze. I sistemi disponibili sul mercato si differenziano in base alla loro architettura, facilità d'uso, flessibilità e capacità di gestire diversi tipi di contenuti.

1.6.1 WordPress

WordPress è una piattaforma open source ampiamente utilizzata per la pubblicazione di contenuti digitali, che consente a chiunque di creare e condividere i propri contenuti. Oggi, è il CMS più diffuso al mondo, ed è utilizzato da oltre il 43% dei siti web, come illustrato dalla figura 1.1 [9]. Di default, WordPress è un CMS tradizionale, tuttavia può anche essere configurato per avere una struttura decoupled, headless o ibrida. La sua popolarità è dovuta alla possibilità di utilizzarlo "*out of the box*" o personalizzarlo in base alle necessità dell'utente [10][11].

Il design intuitivo e la facilità di utilizzo fanno di WordPress una soluzione ideale per una vasta gamma di utenti, dai principianti ai professionisti.

È progettato per essere facilmente installato, con minime esigenze di configurazione iniziale, e garantisce buona accessibilità, performance e sicurezza [9].

Per utilizzare WordPress, è necessario che l'host supporti PHP (versione 7.4 o superiore), MySQL (versione 8.0 o superiore) o MariaDB (versione 10.5 o superiore) e HTTPS. Sebbene sia compatibile con qualsiasi server che supporti PHP e MySQL, per performance avanzate è consigliato l'uso di server come Apache o Nginx [12].

Le principali caratteristiche di WordPress includono [9][10][13][14][15]:

- Semplicità d'uso, che rende veloce il processo di pubblicazione dei contenuti online.
- Flessibilità, che consente di creare qualsiasi tipo di sito web, da blog personali a siti aziendali o governativi, con possibilità di personalizzare la grafica tramite temi e di estendere le funzionalità tramite plugin.
- Possibilità di usufruire di strumenti di pubblicazione come bozze, pubblicazioni programmate, e la possibilità di mantenere i contenuti privati o pubblici [13].
- Gestione dei ruoli, che consente di definire differenti livelli di accesso per editori, autori e amministratori .
- Facilità di caricamento dei media, per gestire immagini, video e altri file multimediali.
- Ottimizzazione SEO, per garantire una buona visibilità sui motori di ricerca.

Dal lato dello sviluppatore, WordPress offre funzionalità avanzate come il sistema di plugin, la possibilità di utilizzare temi personalizzati e la gestione di contenuti avanzata grazie alle molteplici tipologie di post e ai metadati [15].

1.6.2 Joomla!

Joomla è un CMS gratuito e open-source per la pubblicazione di contenuti digitali sul web. Come WordPress, è un CMS tradizionale, ma può essere configurato per assumere un modello decoupled o headless. Basato su un'architettura MVC (Model-View-Controller), offre una struttura flessibile che consente di creare siti web dinamici ed efficienti. Grazie alla sua versatilità, Joomla può essere utilizzato per la gestione di diversi tipi di siti, da piccoli blog personali a portali aziendali e complessi e-commerce.

Progettato per essere facile da installare e configurare, Joomla offre un'interfaccia intuitiva che riduce la curva di apprendimento. Inoltre, la sua architettura modulare permette agli utenti di estendere le funzionalità del CMS attraverso un vasto ecosistema di estensioni e template [16][17].

Dal punto di vista tecnologico, Joomla richiede un ambiente PHP (versione 7.2.5 o superiore) e un database MySQL (5.6 o superiore) o MariaDB (10.1 o superiore). Per ottenere le migliori prestazioni, è consigliato l'uso di un server web come Apache, Nginx o Microsoft IIS [16].

Attualmente, Joomla è utilizzato per la gestione di oltre il 3% dei siti web mondiali, grazie ai numerosi vantaggi che offre [18][17]:

- Flessibilità e personalizzazione: supporta una vasta gamma di estensioni e template per adattarsi a esigenze specifiche.

- SEO integrato: i contenuti creati con Joomla sono automaticamente ottimizzati per i motori di ricerca.

- Elevata sicurezza: include funzionalità avanzate come l'autenticazione a due fattori e controlli di accesso granulari per proteggere il sito.

Pertanto, Joomla è un CMS particolarmente adatto a utenti che desiderano un sistema altamente configurabile, ma allo stesso tempo intuitivo.

1.6.3 Drupal

Drupal è un CMS open-source, modulare e API-first, progettato per offrire un'esperienza flessibile e scalabile nella gestione dei contenuti digitali. É un CMS molto flessibile, nativamente predisposto per funzionare come CMS tradisionale ma anche come headless o decoupled. Grazie alla sua architettura componibile, Drupal consente di costruire siti web sofisticati ed adattabili a qualsiasi esigenza, dalle piattaforme editoriali ai portali aziendali e alle applicazioni web complesse. É scritto in PHP e utilizza un database MySQL, PostgreSQL o SQLite per l'archiviazione dei dati. È compatibile con server web come Apache, Nginx e Microsoft IIS, garantendo un'infrastruttura solida per la gestione dei contenuti digitali [19][20].

Le principali caratteristiche di Drupal includono [19][17]:

- Flessibilità e personalizzazione: Drupal permette di creare e gestire contenuti strutturati in modo versatile, garantendo un'ampia personalizzazione grazie alla sua architettura modulare.
- Esperienza utente ottimizzata: fornisce strumenti intuitivi per la creazione e gestione dei contenuti, adatti sia ai team di marketing che agli editori, senza necessità di competenze tecniche avanzate.
- Approccio API-first: facilita l'integrazione con diversi canali digitali, consentendo la distribuzione dei contenuti su dispositivi web e mobile.
- Opzioni low-code/no-code: offre interfacce intuitive per consentire ai marketer di gestire autonomamente strategie digitali e flussi di pubblicazione.
- Sicurezza: aderisce agli standard di privacy e sicurezza del settore, garantendo protezione avanzata per i dati e le infrastrutture web.
- Gestione dei ruoli: permette di assegnare permessi e controlli specifici per diversi utenti, migliorando la collaborazione e l'accesso dei contenuti.

1.6.4 Strapi

Strapi è un CMS headless open-source, progettato per offrire agli sviluppatori un ambiente altamente personalizzabile per la gestione dei contenuti. Consente di creare e distribuire contenuti su qualsiasi piattaforma o dispositivo, integrandosi facilmente con framework front-end come React, Vue.js e Next.js [21].

Questo CMS offre un'interfaccia intuitiva che permette di costruire e gestire i modelli di contenuto in modo semplice, senza bisogno di scrivere codice. Il Content-Type Builder e i Custom Fields permettono di modellare i dati in base alle esigenze del progetto, mentre le Dynamic Zones danno agli editor la possibilità di modificare la struttura delle pagine senza l'intervento degli sviluppatori [22][21].

Dal punto di vista delle performance e della scalabilità, Strapi offre la possibilità di essere ospitato su Strapi Cloud, un'infrastruttura ottimizzata che include database PostgreSQL, provider e-mail e CDN, senza necessità di gestione manuale. Tuttavia, può essere anche *self-hosted*, lasciando agli sviluppatori la libertà di scegliere il proprio ambiente server e database [21].

Le principali caratteristiche di Strapi includono:

- Distribuzione dei contenuti tramite REST o GraphQL API.
- Personalizzazione avanzata: gestione flessibile di modelli e relazioni dei contenuti.
- Scalabilità e hosting flessibile: possibilità di scegliere tra Strapi Cloud o self-hosting.
- Supporto per i più diffusi database e framework front-end [21][22].

1.7 Digitalizzazione e gestione dei contenuti per l'Istituto Storico di Forlì-Cesena: un caso di studio

L'Istituto Storico della Resistenza e dell'Età Contemporanea di Forlì-Cesena è un punto di riferimento per ricercatori, studiosi e insegnanti interessati alla storia contemporanea, in particolare quella dell'Ottocento e del Novecento. Fondato l'8 ottobre 1971, l'Istituto si dedica alla salvaguardia, al riordino e alla valorizzazione del patrimonio archivistico e documentario relativo alla Resistenza e alla storia contemporanea della provincia di Forlì-Cesena.

Nel corso degli anni, l'Istituto ha ampliato e diversificato le proprie attività, consolidando il proprio ruolo non solo come centro di ricerca, ma anche come promotore di iniziative culturali e di formazione. Oltre a sostenere studi e ricerche su tematiche storiche di rilievo, organizza corsi di formazione, seminari e attività rivolti a docenti e studenti, con l'obiettivo di difondere la conoscenza storica nelle scuole. Accanto a queste iniziative, realizza anche convegni, mostre e presentazioni, contribuendo alla sensibilizzazione del pubblico su tematiche storiche di attualità.

Un aspetto fondamentale del lavoro dell'Istituto è la gestione e la conservazione del proprio archivio, che raccoglie e cataloga documenti storici rendendoli accessibili a studiosi e cittadini interessati. La digitalizzazione e la pubblicazione online di parte di questi materiali ha contribuito a rendere il patrimonio documentario dell'Istituto maggiormente fruibile, e a diffonderne la conoscenza oltre il contesto locale.

La comunicazione digitale infatti riveste un ruolo centrale per un ente culturale come l'Istituto Storico, permettendo di raggiungere un pubblico sempre più ampio e rendendo i contenuti accessibili alla comunità. Grazie a strumenti e piattaforme digitali, è possibile così valorizzare il patrimonio documentario. Inoltre, i canali digitali favoriscono l'interazione diretta con la comunità, dando spazio a dialogo e condivisione.

Di seguito sono illustrati i canali principali dell'Istituto storico e i contenuti in essi proposti.

1.7.1 Sito principale

Il sito web principale dell'Istituto rappresenta il fulcro del suo ecosistema digitale, fungendo da punto di accesso alle sue attività e risorse. Oltre a fornire informazioni istituzionali, offre anche aggiornamenti su eventi e iniziative culturali, oltre a permettere agli utenti di accedere e consultare pubblicazioni, archivi digitali e materiali didattici. Ha una struttura pensata per essere intuitiva e user-friendly, con una navigazione semplice ed efficace.

Sono presenti però alcune problematiche che rappresentano delle sfide per la gestione attuale dell'ecosistema digitale dell'Istituto Storico. Con il tempo, l'ecosistema si è espanso, e sono nati a partire dalle sue iniziative numerosi siti satelliti, che seppur preziosi, si rivelano problematici da diversi punti di vista: mantenere aggiornati questi siti, al fine di garantire sicurezza e funzionalità, diventa difficoltoso. Inoltre, la gestione di molteplici siti rende difficile per l'utente la navigazione dei contenuti offerti dall'Istituto.

1.7.2 La Diga Civile

Questo portale didattico si occupa della divulgazione della storia della violenza politica e del terrorismo in Emilia-Romagna nel secondo dopoguerra, attraverso la creazione di schede informative, documenti, materiali multimediali e raccolte di testimonianze. Attraverso il suo operato, viene creato un legame tra ricostruzione storica, preservazione della memoria e partecipazione civica.

Il portale didattico nasce dalla collaborazione dell'Istituto Sotrico con la sezione forlivese del Centro di documentazione Archivio Flamigni e con il contributo dell'Istituto per i Beni Artistici, Culturali e Naturali della Regione Emilia-Romagna.

1.7.3 Viaggi della Memoria

Questo sito nasce da un progetto, finanziato dalla Regione Emilia-Romagna, e sviluppato dall'Istituto Storico. Il sito si pone come laboratorio didattico, indirizzato a docenti e studenti delle scuole superiori, nel quale vengono analizzati due episodi drammatici del secolo scorso, attraverso un percorso virtuale che permette di comprendere il contesto intorno a tali vicende. In questo modo, il sito permette agli utenti di vivere un viaggio della memoria interamente virtuale, composto di tappe caratterizzate da video e documenti che permettono la comprensione dell'argomento.

1.7.4 Migrati e migranti: passato e presente del verbo Migrare. Una storia europea

"Migrati e migranti: passato e presente del verbo migrare. Una storia europea" è stato realizzato dall'Istituto Storico della Resistenza e dell'Età Contemporanea di Forlì-Cesena e finanziato dal bando "Viaggi attraverso l'Europa" indetto dall'Assemblea legislativa dell'Emilia-Romagna. Il suo obiettivo è approfondire il fenomeno delle grandi migrazioni che hanno interessato l'Italia e l'Unione Europea, e stimolare una riflessione su sfide e opportunità che queste rappresentano.

Capitolo 2

Tecnologie utilizzate

In questo capitolo, vengono elencate, analizzate e approfondite tutte le tecnologie impiegate nel progetto, considerando linguaggi di programmazione, framework e API.

2.1 HTML

HTML (HyperText Markup Language) è un linguaggio di markup, ovvero una tipologia di linguaggio che definisce la struttura ed il significato di un documento attraverso l'uso di tag e attributi. Grazie alla sua sintassi HTML permette ai browser di interpretare e renderizzare correttamente i contenuti delle pagine web. I linguaggi di markup possono essere di due tipologie:

- Procedurali: danno specifiche istruzioni da seguire per ottenere la corretta visualizzazione del testo considerato.

- Descrittivi: individuano il ruolo semantico di una porzione di testo all'interno di un documento [23].

HTML appartiene alla seconda categoria di linguaggi di markup. Nasce infatti con l'obiettivo di permettere la connessione di documenti scientifici attraverso Internet, obiettivo che realizza tramite tag di collegamento come

<a> e <link>. Per questo motivo, è il linguaggio alla base del World Wide Web [24].

2.1.1 Struttura di un documento HTML

Un file HTML è costituito da una serie di tag che ne definiscono la struttura ed il contenuto. Un documento HTML è composto da alcuni tag fondamentali, come <!DOCTYPE html> che specifica la versione del linguaggio, <head> che contiene metadati, titolo e collegamenti a CSS, e <body> che contiene le informazioni da visualizzare nella pagina.

Permette anche l'aggiunta di elementi di testo come intestazioni e paragrafi, elementi multimediali come immagini e video, e formattazione, come grassetto e corsivo.

Con l'avvento di HTML5, il linguaggio si focalizza maggiormente sulla semantica, cioè si concentra sull'utilizzo di tag con significati specifici per migliorare la comprensione dei contenuti da parte dei browser e dei motori di ricerca. Alcuni tag semantici sono <header> per le intestazioni e <section> per sezioni tematiche [25].

2.2 CSS

CSS (Cascading Style Sheets) è un linguaggio per la scrittura di fogli di stile, ovvero insiemi di regole che specificano la resa presentazionale di un documento. Nasce dalla necessità di permettere la personalizzazione della grafica delle pagine web, e discostarsi dalle visualizzazioni predefinite generate dai browser a partire dagli elementi HTML. L'uso di CSS permette l'aggiunta di stili, come font, colori e spaziature [26] e di sviluppare i contenuti di un documento in maniera indipendente dalla sua grafica [27]

Uno dei principali vantaggi dell'utilizzo dei fogli di stile è la possibilità di centralizzare la definizione della resa grafica di determinati elementi in un unico punto del documento, aumentando la facilità di scrittura del codice e la sua manutenibilità, e diminuendo i costi computazionali e il tempo di

caricamento delle pagine [28]. Grazie a questa organizzazione, gli sviluppatori possono riutilizzare fogli di stile in più pagine, contribuendo a dare uniformità e coerenza al layout del sito [28].

Un aspetto fondamentale di CSS è il concetto di cascata (*cascading*, da cui deriva il suo nome), un meccanismo che regola la gestione dei conflitti che si verificano quando ad uno stesso elemento si tenta di applicare più stili contrastanti, da due o più fogli di stile differenti. Quando questo si verifica, lo stile da applicare viene scelto sulla base di una gerarchia definita dalla *specificità* delle regole, pertanto la regola più specifica avrà la priorità. Questo meccanismo non è solo un metodo efficace per la risoluzione dei conflitti, ma permette anche agli utenti di applicare comodamente al di sopra del foglio di stile messo a disposizione dallo sviluppatore, un proprio foglio di stile. Questa funzionalità porta numerosi vantaggi, soprattutto nel campo dell'accessibilità, permettendo all'utente di adattare la resa grafica alle proprie esigenze [28].

2.3 Bootstrap

Bootstrap è un framework CSS open-source per velocizzare e facilitare lo sviluppo front-end delle applicazioni web. Offre una vasta gamma di strumenti per lo sviluppo di interfacce *responsive*, ovvero in grado di adattarsi automaticamente alle dimensioni del dispositivo utilizzato. Tra i più importanti strumenti che mette a disposizione vi sono il suo sistema a griglia flessibile, che permette di dividere i contenuti di una pagina su 12 colonne [29] e componenti predefiniti (come bottoni, form e modali) per limitare il tempo di implementazione di funzionalità comuni [30], permettendo allo stesso tempo allo sviluppatore di seguire, senza grandi sforzi, le best practice di sviluppo web [31].

2.4 PHP

PHP è un linguaggio di scripting lato server in continua evoluzione, estremamente popolare nel mondo della programmazione web grazie alla sua sintassi semplice ed intuitiva. È apprezzato in modo particolare per la facilità d'uso, la capacità di supportare siti web basati su database e la possibilità di essere integrato direttamente all'interno di pagine HTML, creando pagine interattive e dinamiche [32]. Tra le caratteristiche chiave che hanno permesso la larga diffusione di PHP vi sono la sua capacità di interagire con i database e la forte sinergia tra PHP, il server Web Apache e il database relazionale MySQL, tre tecnologie open-source fondamentali per lo sviluppo web. Un altro grande vantaggio è che, essendo PHP un linguaggio interpretato, gestisce automaticamente la memoria allocata dagli script attraverso il suo motore di esecuzione, dando la possibilità ai programmatori di non preoccuparsi di chiudere manualmente le connessioni ai database o liberare le risorse allocate [33].

2.5 JavaScript

JavaScritp (JS) è un linguaggio di programmazione leggero ed interpretato, con capacità di essere orientato agli oggetti [34]. É un linguaggio prototype-based, multi-paradigma, single-threaded e dinamico, che supporta stili object-oriented, imperativi e dichiarativi [35].

A livello sintattico, è simile a Java, C++, e C, con costrutti di programmazione come *if*, *while* e operatori logici come &&. Tuttavia, le somiglianze si limitano alla sintassi: JS è debolmente tipizzato, ovvero non è necessario esplicitare il tipo di una variabile. Inoltre, gli oggetti in JS si basano su un modello basato su prototipi invece che su classi, permettendo a oggetti di ereditare proprietà e metodi direttamente da altri oggetti [36].

JS nasce come linguaggio client-side interpretato dal browser, con il fine di dare dinamismo ai contenuti web, combinando le capacità di scripting dell'interprete JS con il DOM (Document Object Model) definito dal browser.

Il DOM permette la connessione di script a pagine web, rappresentando la struttura dei documenti attraverso un albero logico in cui ogni ramo termina con un nodo che rappresenta degli oggetti. Accedendo agli elementi del DOM, è quindi possibile modificare e controllare i contenuti mostrati, sia a livello di grafica che di struttura [37]. In questo modo, vengono aggiunte a documenti, di natura statica, delle capacità comportamentali, tramite la definizione di *event handlers*, ossia sezioni di codice JS che vengono eseguite quando si verifica un determinato evento.

L'uso di JS come linguaggio di programmazione non è limitato ai browser web: questo linguaggio è stato progettato per essere integrato all'interno di qualsiasi applicazione.

2.6 React

ReactJS è una libreria front-end, open-source e basata sui componenti, responsabile unicamente del lievello di presentazione dell'applicazione web. Sfrutta il meccanismo del *virtual DOM*, una rappresentazione del DOM reale che viene creata ogni volta che uno stato o un dato cambia in un'applicazione React, da cui React individua gli elementi che differiscono con il DOM originale, che vengono quindi aggiornati, senza ricaricare l'intera pagina. Questo meccanismo semplifica e rende più efficiente l'aggiornamento dinamico di contenuti web [38].

React è utilizzato per costruire la visualizzazione di una pagina web a partire da singoli blocchi di interfaccia utente (UI) chiamati *componenti*, che hanno una propria logica e un proprio aspetto. Concretamente, i componenti sono funzioni JavaScript che restituiscono *markup*. Alcuni sono messi a disposizione dalla libreria, ma lo sviluppatore può costruire i propri, per massimizzare la riusabilità del codice e adattarlo alle proprie esigenze [39][40].

La sintassi utilizzata dai componenti React è detta JSX (JavaScript XML), ed è un misto di codice HTML markup e tag personalizzati [41]. É possibile passare informazioni ai componenti da un componente padre a un componen-

te figlio utilizzando i *prop* (abbreviazione di *properties*), proprietà che vengono definite nel componente padre, normalmente sotto forma di attributi JSX, e ricevute dal componente figlio come oggetto.

I componenti hanno uno stato, ovvero una memoria interna, ed un ciclo di vita, che possono essere manipolati utilizzando funzioni speciali dette *hooks* [40].

2.7 MySQL

MySQL è un *database management system* (DBMS) open-source basasto su SQL. Un database è una collezione di dati strutturati, gestita con l'ausilio di un DBMS per facilitarne l'archiviazione, la manipolazione e l'interrogazione.

MySQL è un database *relazionale*, ovvero i dati al suo interno sono organizzati in tabelle strutturate. Le informazioni vengono memorizzate in file ottimizzati per garantire elevate prestazioni. Il modello logico di MySQL, basato su elementi quali database, tabelle, viste, righe e colonne, offre uno sviluppo flessibile, in cui lo sviluppatore può definire regole che gestiscono le relazioni tra i dati. Tra queste vi sono associazioni *uno-a-uno*, *uno-a-molti*, e vincoli di integrità come *unique*, *required*, e riferimenti tra tabelle. Questa struttura garantisce l'integrità del database, evitando la presenza di dati duplicati o obsoleti.

SQL, acronimo di Structured Query Language, è il linguaggio standardizzato più utilizzato per interagire con i database. Il codice SQL può essere eseguito direttamente, può essere incorporato all'interno di codice in un altro linguaggio, oppure essere nascosto dietro un'API, a seconda delle necessità [42].

MySQL è strutturato secondo un'architettura client/server: il server è il database MySQL, mentre i client sono le applicazioni che comunicano con il server, interrogandolo o salvando informazioni al suo interno.

2.7.1 Caratteristiche principali

Tra le più importanti funzionalità di MySQL vi sono:

- View: le viste sono tabelle virtuali basate sul risultato di interrogazioni SQL, ovvero mostrano dinamicamente i dati di una o più tabelle. Migliorano la sicurezza limitando l'accesso ai dati, semplificano query complesse e rendendole più leggibili.

- Stored procedures: sono blocchi di codice SQL salvati ed eseguiti direttamente nel database, al fine di automatizzare operazioni ripetitive, aumentando la leggibilità e l'efficienza delle operazioni sul database.

- Transazioni: questo termine si riferisce all'esecuzione in un "blocco" unico di una serie di operazioni su un database, rendendole atomiche. Questo significa che le operazioni della transazione sono vincolate al loro completamento con successo. Se anche solo una dovesse non essere completata con successo, allora nessuna viene completata. Questo meccanismo è di fondamentale importanza per garantire l'integrità dei dati nel database.

- Vincoli di chiave esterna: sono regole che garantiscono l'integrità referenziale tra tabelle collegate, assicurandosi che i valori presenti in una colonna che fa riferimento a una chiave primaria di un'altra tabella siano validi [43].

2.8 NPM

NPM (Node Package Manager) è il registro di pacchetti più utilizzato per la gestione delle dipendenze nei progetti JavaScript, ed è quindi essenziale per gli sviluppatori che lavorano con JS, Node.js e framework come React, Angular e Vue.js. Permette a sviluppatori di condividere, scaricare e gestire pacchetti di codice riutilizzabili con tutto il mondo [44]. Tra le funzionalità principali di NPM vi sono:

- La possibilità di adattare pacchetti di codice o incorporarli direttamente all'interno delle applicazioni.

- Condividere codice con altri utenti di NPM.

- Creare organizzazioni per coordinare il mantenimento di pacchetti.

- Gestire molteplici versioni di codice e le loro dipendenze, aggiornare facilmente le applicazioni.

NPM consiste di quattro componenti fondamentali:

- Il sito web ufficiale di npm: qui gli sviluppatori possono trovare nuovi pacchetti, impostare il proprio profilo e creare organizzazioni per gestire l'accesso a pacchetti pubblici o privati.

- L' interfaccia a linea di comando (CLI): viene eseguita da un terminale, ed è il modo principale in cui gli sviluppatori interagiscono con NPM. Dalla linea di comando è possibile installare, aggiornare, rimuovere e gestire pacchetti. Include anche molte funzionalità come manipolazione della cache dei pacchetti, controllo dei pacchetti a livello di sicurezza, gestione dei file di configurazione di NPM, modifica di un pacchetto, creazione del file package.json, installazione di pacchetti e molto altro [45].

- Il registro pacchetti JavaScript: consiste in un sito che implementa le specifiche del CommonJS Package Registry per la lettura delle informazioni dei pacchetti. É il registro utilizzato da NPM per impostazione predefinita, anche se lo sviluppatore può configurare NPM in modo da usare qualsiasi registro compatibile.

- Il file package.json: è un file di configurazione che contiene i metadati del progetto, come nome, versione, dipendenze e script personalizzati [44].

2.9 API

Le API (Application Programming Interface) sono un elemento fondamentale nello sviluppo software, poichè permettono ad applicazioni e sistemi di interagire in modo standardizzato. Concreatamente, un' API è un insieme di regole e protocolli che definiscono l'interazione tra due software, permettendo lo scambio di dati e l'accesso a funzionalità senza che lo sviluppatore debba conoscere i dettagli interni dei sistemi coinvolti. Fungono da intermediari tra due software, definendo le richieste che un software può inviare, il formato dei dati in cui la risposta sarà restituita, e le operazioni possibili. Le API possono essere classificate in diverse categorie, tra cui le principali sono:

- API locali: sono interne all'applicazione ed utilizzate per mettere in comunicazione i suoi componenti, generalmente il back-end con il front-end.
- API pubbliche: aperte a tutti e utilizzabili senza restrizioni, solitamente con l'ausilio di un sistema di autenticazione basato su chiavi. Esempi di API pubbliche sono le API di Google Maps e Twitter.
- API private: sono utilizzate all'interno di una organizzazione e non sono di pubblico dominio [46].

Le API possono avere architetture differenti a seconda delle esigenze per quanto riguarda le modalità di scambio di dati:

- API REST (Representational State Transfer): sono API basate sul protocollo HTTP. Sono *stateless*, ovvero ogni richiesta è indipendente, e lo stato non è mantenuto tra richieste successive. I metodi di manipolazione dei dati sono GET per il recuperto dei dati, POST per l'inserimento, PUT per l'aggiornamento e DELETE per la rimozione [47].
- API SOAP (Simple Object Access Protocol): hanno un protocollo più rigido rispetto a REST, basato su XML.

L'utilizzo delle API porta numerosi vantaggi: in primo luogo, permettono di aumentare la sicurezza nello scambio di dati sensibili, implementando meccanismi di autenticazione tramite token, cifratura dei dati (specialmente grazie all'utilizzo di HTTP) e CORS (Cross-Origin Resource Sharing) che permette il controllo di quali domini possono accedere all'API. In secondo luogo, permettono di modularizzare un sistema in componenti indipendenti, riutilizzabili e scalabili, rendendo le applicazioni che ne fanno uso manutenibili e integrabili.

2.10 XAMPP

XAMPP è un software open-source che fornisce un ambiente di sviluppo locale per applicazioni web. Il suo nome è un acronimo che definisce gli elementi principali che lo compongono:

- X indica che il software è multipiattaforma, ovvero è compatibile con Windows, macOS, e Linux.
- A indica Apache, il server web utilizzato.
- M indica MySQL o MariaDB, i DBMS (DataBase Management System) supportati.
- P indica PHP o Perl, i linguaggi di scripting o programmazione supportati.

É un *solution stack*, ovvero un insieme di componenti necessari per creare piattaforme complete che permettano ad applicazioni che le utilizzano di non necessitare di software aggiuntivi per funzionare. Permette agli sviluppatori di impostare un ambiente server localmente, senza la necessità di configurare ogni componente.

Alcuni dei principali vantaggi che derivano dall'impiego di XAMPP includono la compatibilità multipiattaforma, la facilità di utilizzo e la facilità di

installazione e impostazione dell'ambiente di sviluppo. XAMPP fornisce inoltre un comodo pannello di controllo che permette di avviare o interrompere vari servizi [48].

2.11 Apache

Apache è il server web più diffuso e riveste un ruolo chiave nell' infrastruttura di Internet per via della sua flessibilità, scalabilità e compatibilità multipiattaforma. Il compito di un server web è ricevere richieste da client, interpretarle e poi restituire la risorsa richiesta, che può essere un file statico o il risultato di un programma eseguito dinamicamente, come uno script PHP o un'applicazione backend.

Quando un utente digita un URL (Uniform Resource Locator) nella barra degli indirizzi del browser, viene inviata una richiesta attraverso Internet al server che ospita il sito corrispondente. Un URL è composto da uno schema che specifica il protocollo da utilizzare, l'host, che può essere un indirizzo IP o un nome di dominio, e il percorso, che identifica la risorsa richiesta all'interno del server. Quando il browser invia la richiesta HTTP al server, in essa vengono inclusi il metodo HTTP (GET, PUT, POST, DELETE, CONNECT) che indica l'azione richiesta, l'URI (Uniform Resource Identifier) che identifica la risorsa, la versione del protocollo HTTP in uso, e una serie di header che forniscono informazioni aggiuntive sulla richiesta. A questo punto il server web in esecuzione sull'host richiesto si occuperà della gestione del messaggio.

Il Server Apache sta in ascolto all'indirizzo IP specificato all'interno del suo file di configurazione, e quando riceve una richiesta, ne analizza gli header e applica ad essa le regole che trova nel file di configurazione. Apache comunica attraverso la rete usando protocolli TCP/IP, protocolli che permettono la comunicazione tra computer sulla rete, ed implementa il protocollo HTTP per la gestione delle richieste e delle risposte web. Ha inoltre un'architettura

composta da moduli di elaborazione delle richieste, di autenticazione e di caching, che lo rendono dinamico e ne migliorano le prestazioni [49].

2.12 Git

Git è un sistema di controllo delle versioni (VCS) gratuito e open-source, progettato per gestire progetti di qualsiasi dimensione in modo efficiente e flessibile [50]. Un VCS è un sistema che memorizza i cambiamenti effettuati ad un file o un insieme di file nel tempo, in modo da permettere agli sviluppatori di consultare versioni precedenti, riportare un file o l'intero progetto ad un suo stadio precedente, mettere a confronto i cambiamenti effettuati nel tempo ed individuare chi ha effettuato una determinata modifica.

L'uso di un VCS è fondamentale per sicurezza ed affidabilità di un progetto: permette di recuperare facilmente file persi o danneggiati, e consente di collaborare in modo efficace. Git, a differenza di altri VCS, è distribuito: ogni sviluppatore che clona un repository ottiene una copia completa della cronologia del progetto. Significa che, anche in caso di guasto al server centrale, il progetto non andrà perso, perchè ogni collaboratore ne possiede una copia completa in locale [51].

2.12.1 Caratteristiche

Git si contraddistingue dagli altri VCS, come Subversion o Perforce, per la sua velocità, semplicità d'uso, e capacità di gestire lo sviluppo non lineare. Permette infatti la creazione e la gestione di ramificazioni in modo efficiente, facilitando il lavoro su più versioni di un unico progetto sviluppate in contemporanea [52].

Una delle caratteristiche fondamentali di Git, che lo differenzia da altri VCS, è il suo modello di archiviazione dati: mentre la maggior parte dei VCS salva le informazioni come una lista di modifiche basate sui file, Git considera i suoi dati come una serie di snapshot dell'intero progetto. Ogni volta che viene effettuato un commit o un salvataggio del progetto, è come se venisse

scattata una fotografia ai file del progetto. Se un file non ha subito modifiche dalla sua versione precedente, non viene nuovamente salvato, ma viene inserito un riferimento alla sua versione precedente già presente in cronologia, gestendo in modo più efficiente lo spazio e migliorando le prestazioni.

Inoltre, è un sistema particolarmente sicuro, grazie alla crittografia hash SHA-1, che protegge le informazioni in database garantendo integrità e sicurezza [52].

2.12.2 Artchitettura

Git organizza il lavoro in tre stati principali:

- **Modified**: indica che il file è stato modificato ma le sue modifiche non sono ancora state registrate nel VCS.
- **Staged**: indica che il file modificato è stato selezionato per essere incluso nel prossimo commit.
- **Committed**: indica che il file è stato salvato con successo nel database locale di Git.

A livello strutturale invece, un progetto Git presenta tre sezioni fondamentali:

- **Working Tree**: rappresenta il codice estratto dal repository, ovvero i file su cui lo sviluppatore lavora.
- **Staging Area**: è l'area che registra quali file modificati saranno inclusi nel prossimo commit.
- **Git Directory**: è il database interno di Git, che memorizza la cronologia del progetto e i metadati. É la directory che viene copiata quando un repository viene clonato.

Grazie alla sua architettura, Git consente una gestione flessibile e scalabile dei progetti, supportando anche lo sviluppo collaborativo [53].

2.13 JSON

JSON (JavaScript Object Notation) è un formato di scambio di dati testuale, leggero e scritto in un formato facilmente leggibile da esseri umani. É un formato per il passaggio di dati tra client e server ampiamente utilizzato, specialmente all'interno delle API web e nelle applicazioni distribuite.

Il formato JSON è facilmente leggibile, grazie al suo formato testuale in cui i dati sono rappresentati come stringhe strutturate. Inoltre, è indipendente dal linguaggio: nonostante derivi da JavaScript, ed infatti abbia un aspetto molto simile agli oggetti JS, è supportato da quasi tutti i principali linguaggi di programmazione, tra cui Python, Java, C++ e PHP. Ha una struttura semplice e flessibile, organizzata per coppie chiave/valore, rendendolo un formato ideale per la rappresentazione di dati strutturati. Inoltre è particolarmente leggero e veloce, e richiede meno spazio rispetto ad altri formati per lo scambio di dati come ad esempio XML. La sua caratteristica più importante però è la possibilità di essere gestito direttamente dalla JS Engine dei browser, senza il supporto di librerie esterne per il parsing [54].

JSON è lo standard per la comunicazione tra client e server tramite HTTP, ma è impiegato anche in altri contesti, dalla configurazione delle applicazioni tramite i file package.json, ai database NoSQL, che lo utilizzano come formato di archiviazione dati.

Capitolo 3

Progetto

In questo capitolo, vengono illustrati il processo di progettazione e le fasi di realizzazione del progetto.

Il capitolo si apre con la definizione degli obiettivi del progetto e un'analisi delle criticità legate alle tecnologie e alle soluzioni attualmente impiegate dall'Istituto Storico per i suoi siti.

Nel corpo centrale del capitolo, viene illustrata la fase di definizione e analisi dei requisiti funzionali e non funzionali, con attenzione particolare alla progettazione grafica delle interfacce. A partire dai requisiti, vengono esposte le fasi di progettazione dei mockup, del database a supporto del sito, e della piattaforma web, analizzandone anche la struttura e le tecnologie utilizzate.

Il capitolo si conclude con una panoramica delle interfacce grafiche realizzate.

3.1 Stato attuale

In questa prima sezione, vengono definiti gli obiettivi del progetto, e vengono analizzate le problematiche legate alla gestione dei siti web dell'Istituto Storico, con un focus sulle criticità derivanti dalla frammentazione dei siti e dalla gestione dei contenuti.

3.1.1 Obiettivo

Il progetto nasce con l'obiettivo di migliorare l'esperienza utente sui siti web dell'Istituto Storico di Forlì-Cesena. Attualmente, l'Istituto gestisce più siti web in parallelo, sviluppati in momenti diversi e con approcci differenti. Questo ha generato una serie di criticità sia dal punto di vista della navigazione e dell'usabilità, sia per quanto riguarda la gestione e la manutenzione dei contenuti.

L'obiettivo principale del progetto è quindi unificare i contenuti in una sola piattaforma, garantendo un'esperienza coerente per gli utenti, e semplificando la gestione e l'aggiornamento dei contenuti per l'Istituto.

Al termine del progetto, si dovrà ottenere una piattaforma con funzionalità simili a quelle di un CMS, ovvero che permetta agli amministratori del sito di aggiornare e aggiungere nuovi contenuti, e agli utenti di visualizzarli in un' interfaccia grafica semplice ed efficace.

3.1.2 Criticità dei siti esistenti

L'attuale rete di siti dell'Istituto Storico presenta diverse problematiche, principalmente legate alla frammentazione dei siti e alla loro gestione. La presenza di più piattaforme scollegate tra loro ha reso la manutenzione e l'aggiornamento dei contenuti un processo complesso e poco efficiente.

Uno dei problemi principali riguarda la gestione dei dati. Ogni sito utilizza un proprio database indipendente, il che comporta una duplicazione delle informazioni nel caso in cui determinati contenuti debbano essere pubblicati su più siti. Questo sistema decentralizzato rende alcune azioni di aggiornamento ripetitive e soggette a errori: alcune modifiche infatti potrebbero richiedere di essere effettuate manualmente su più siti, con il rischio di creare delle incongruenze nei contenuti.

Dal punto di vista grafico e strutturale, i siti si presentano con design molto differenti, dovuti allo sviluppo indipendente delle diverse piattaforme. Questa mancanza di uniformità genera diversi problemi: gli utenti che

navigano tra i vari siti possono essere disorientati a causa delle differenze stilistiche e strutturali, faticando a comprendere dove cercare le informazioni di interesse. Inoltre, ogni sito adotta un'organizzazione interna dei contenuti differente, rendendo più complesso il passaggio da un sito all'altro e la ricerca interna dei contenuti. Infine, la mancanza di un design coerente tra i siti rende difficile all'utente percepirli come entità uniche, facenti tutti parte della rete dei siti dell'Istituto Storico di Forlì-Cesena. Questo complica il riconoscimento dell'identità dell'Istituto, diminuendo la fiducia degli utenti nei contenuti proposti. La mancanza di coerenza visiva ricopre un'importanza fondamentale, perchè contrasta con i concetti di "brand consistency", che richiedono di fornire all'utenza un'immagine coerente e unitaria di un brand o un'organizzazione per rafforzarne l'immagine e migliorare l'esperienza complessiva dell'utente. Un design uniforme e una struttura unificata permetterebbero quindi all'Istituto di assumere un'identità più chiara e definita agli occhi degli utenti, che incentiva la costruzione di relazioni durature con loro [55].

Un'ulteriore problematica riguarda l'infrastruttura tecnologica di due dei siti dell'Istituto, che utilizzano un CMS open-source disponibile su GitHub: BraDypUS CMS. Questo sistema di gestione dei contenuti ha permesso di realizzare siti funzionali ed efficaci, che offrono oltretutto una struttura organizzativa dei dati basata su database SQLite. Tuttavia, l'adozione di un CMS generico ha introdotto alcune criticità: il database creato da BraDypUS CMS segue una logica standardizzata, pensata per supportare una vasta gamma di progetti senza una struttura pensata specificamente per le necessità dell'Istituto. Questo ha portato a un modello dei dati poco ottimizzato e molto generico.

A tutto ciò si aggiunge un'ulteriore fattore critico: BraDypUS CMS non è più attivamente mantenuto. L'assenza di futuri aggiornamenti e di supporto per il software ha introdotto un'incertezza sulla sua affidabilità nel lungo termine, rendendo necessario valutare soluzioni alternative per le piattaforme dell'Istituto.

3.1.3 Tecnologie attualmente utilizzate

L'Istituto Storico di Forlì-Cesena ha adottato diverse soluzioni tecnologiche per la gestione dei propri siti web.

Due delle piattaforme attualmente in uso non sono completamente statiche, ma si basano, come menzionato precedentemente, su un CMS. Questo CMS ha permesso la creazione e la gestione di contenuti dinamici tramite una comoda interfaccia.

Dal punto di vista della gestione dei dati, BraDypUS CMS utilizza un database SQLite, un database leggero e integrato direttamente in un file. SQLite offre una soluzione semplice e adatta a siti di piccole dimensioni.

Oltre a tale CMS, i siti web dell'Istituto si avvalgono anche di tecnologie standard per lo sviluppo web, tra cui HTML, CSS, e JavaScript, che vengono utilizzate per la struttura e lo stile delle pagine.

3.2 Analisi dei requisiti

A partire da un primo colloquio con alcuni esponenti dell'Istituto Storico, e dall'analisi dei siti attualmente in uso, sono stati individuati diversi requisiti, sia funzionali che non funzionali.

3.2.1 Requisiti funzionali

Al fine di garantire il corretto funzionamento del sito finale, l'applicazione deve necessariamente soddisfare i seguenti requisiti:

- Visualizzazione di varie tipologie di pagine: il sito deve permettere la visualizzazione di contenuti di vario genere, con grafiche diversificate e adattate ai tipi di contenuto. Ad esempio, deve essere possibile visualizzare pagine semplici, pagine aventi sottopagine al proprio interno, pagine composte da raccolte di risorse (bibliografie, emeroteche) e pagine che visualizzano informazioni di archivio.

- Mappatura delle informazioni nel database: ogni tipologia di contenuto deve essere correttamente modellata all'interno del database, garantendo una struttura coerente e facilmente interrogabile.

- Generazione dinamica delle pagine: per garantire un funzionamento simile a un CMS, le pagine devono essere generate dinamicamente raccogliendo e visualizzando i contenuti dal database.

- Organizzazione intuitiva dei contenuti: per permettere una navigazione efficace, i contenuti devono essere riorganizzati all'interno del menu di navigazione.

3.2.2 Requisiti non funzionali

Per migliorare l'esperienza utente, il progetto dovrà rispettare i seguenti vincoli:

- Grafica minimalista: il design del sito dovrà essere semplice e pulito, in linea con l'identità visiva dell'attuale sito principale dell'Istituto Storico. Questo punto viene approfondito ulteriormente nella prossima sezione, dedicata all'analisi e alla progettazione delle interfacce.

- Responsività: il sito dovrà essere ottimizzato per diversi dispositivi, garantendo una corretta visualizzazione per schermi di diverse dimensioni.

- Scalabilità: il sistema dovrà essere progettato in modo modulare per consentire future estensioni, come la gestione degli utenti e l'inserimento di nuovi contenuti da parte degli amministratori.

3.3 Analisi e progettazione delle interfacce

Prima della realizzazione di mockup per le nuove interfacce del sito dell'Istituto, è stato svolto un lavoro di analisi preliminare dell'organizzazione e

della visualizzazione attuale dei contenuti. Questo studio è stato guidato da un confronto diretto con alcuni membri dell'Istituto Storico, con l'obiettivo di identificare elementi da modificare e migliorare.

3.3.1 Raccolta delle esigenze e analisi preliminare

Nel primo incontro con gli esponenti dell'Istituto, sono stati analizzati i siti in uso attualmente, e sono stati definiti alcuni requisiti per quanto riguarda l'organizzazione della home page del sito principale. In questo contesto è stato richiesto esplicitamente di:

- Progettare un nuovo, seppur simile, layout della home page, che migliorasse la sua organizzazione e rimuovesse alcune sezioni ripetitive.

- Ristrutturare il menu di navigazione in modo da poter supportare tutti i contenuti provenienti dai vari siti dell'Istituto.

- Modificare o eliminare il carosello della home page (visualizzabile nella figura 3.1), dal momento che non garantisce sufficiente visibilità agli elementi mostrati.

- Rimuovere i bottoni degli eventi (visualizzabili nella figura 3.1), considerati ridondanti in quanto fungono da collegamento a contenuti già accessibili tramite il menu di navigazione.

3.3.2 Scelte grafiche

In accordo con l'Istituto, si è deciso di mantenere una coerenza visiva con il sito principale esistente, per preservare l'identità grafica già consolidata. Per questo motivo, anche la palette utilizzata è rimasta simile a quella già in uso.

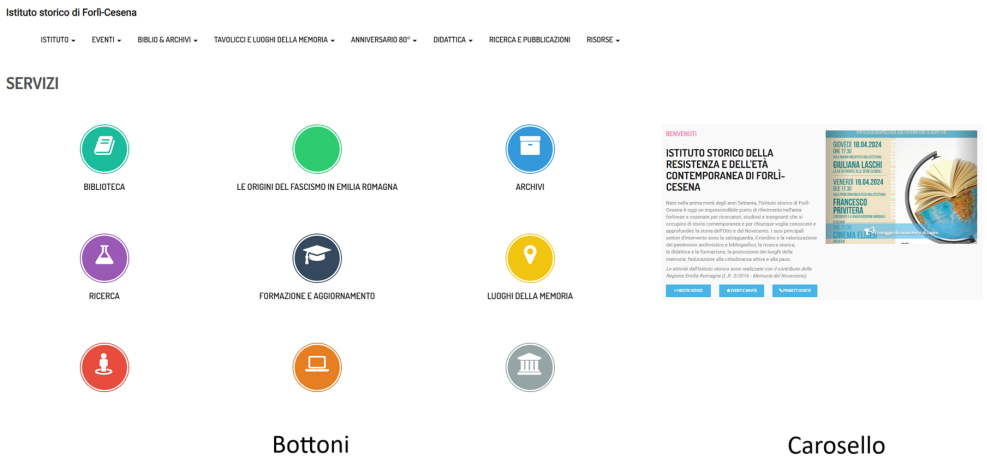

Figura 3.1: Screenshot di sezioni in home che mostrano i bottoni e il carosello

3.3.3 Riorganizzazione del menu di navigazione

Per ristrutturare il menu di navigazione, sono stati analizzati i contenuti di tutti i siti dell'Istituto, al fine di valutare la loro possibile collocazione. L'obiettivo è stato creare un menu semplice ed intuitivo, con un numero minore di elementi per un design minimal e funzionale.

3.3.4 Realizzazione dei mockup

Una volta definita l'organizzazione del menu, sono stati realizzati i mockup con l'ausilio del software Balsamiq. Per i mockup, si è utilizzato l'approccio mobile-first: inizialmente sono stati pianificati e realizzati i modelli dell'interfaccia per dispositivi mobile, per passare poi a quelli con visualizzazione più grande per tablet e desktop, ottenuti adattando i modelli mobile per schermi di dimensioni maggiori. Come menzionato in precedenza, i layout sono stati modificati in maniera minima rispetto ai design già implementati, in modo da dare continuità con lo stile precedentemente stabilito dal sito principale. Al termine di questo processo, sono stati ottenuti diversi mockup possibili. Alla figura 3.2 è consultabile il mockup, sia in versione mobile che desktop, scelto come modello da seguire per la fase di implementazione della

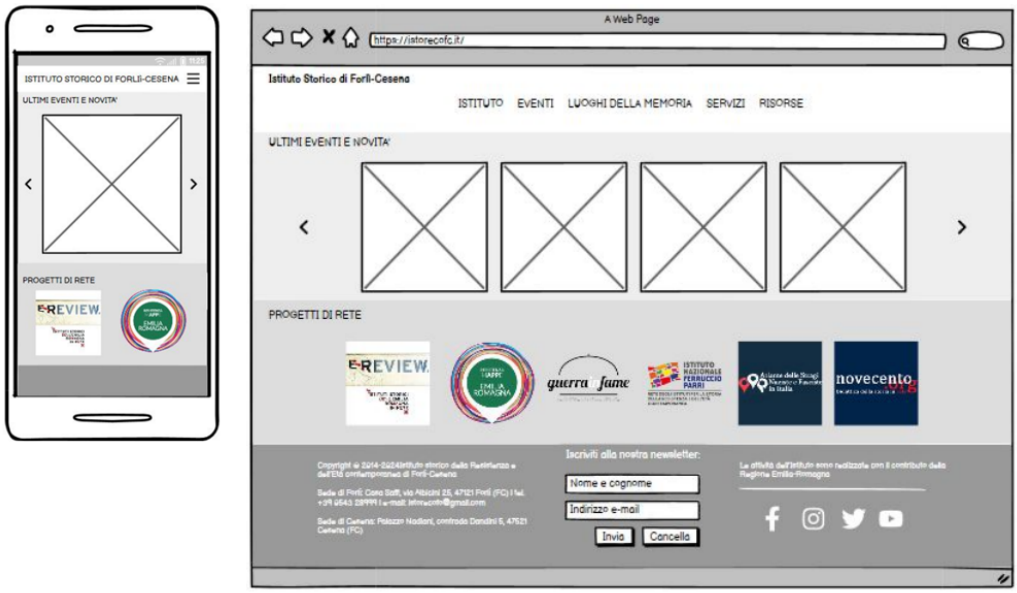

Figura 3.2: Mockup home page versione mobile e desktop

nuova grafica per la home del sito.

3.4 Progettazione e sviluppo del database

Il sistema di database attualmente in impiego dai siti dell'Istituto, come accennato in precedenza, presenta diverse criticità:

- É creato e gestito dal CMS, ed ha quindi una struttura generica pensata per supportare una vasta gamma di progetti, pertanto non è ottimizzato per le specifiche esigenze dell'Istituto.

- É presente un database per ogni sito dell'Istituto basato su CMS. Questa mancanza di un database unico e centralizzato comporta una difficoltà nell'aggiornamento dei dati.

Queste problematiche hanno determinato la decisione di studiare e creare un nuovo database, che possa supportare tutti i contenuti dei diversi siti, e

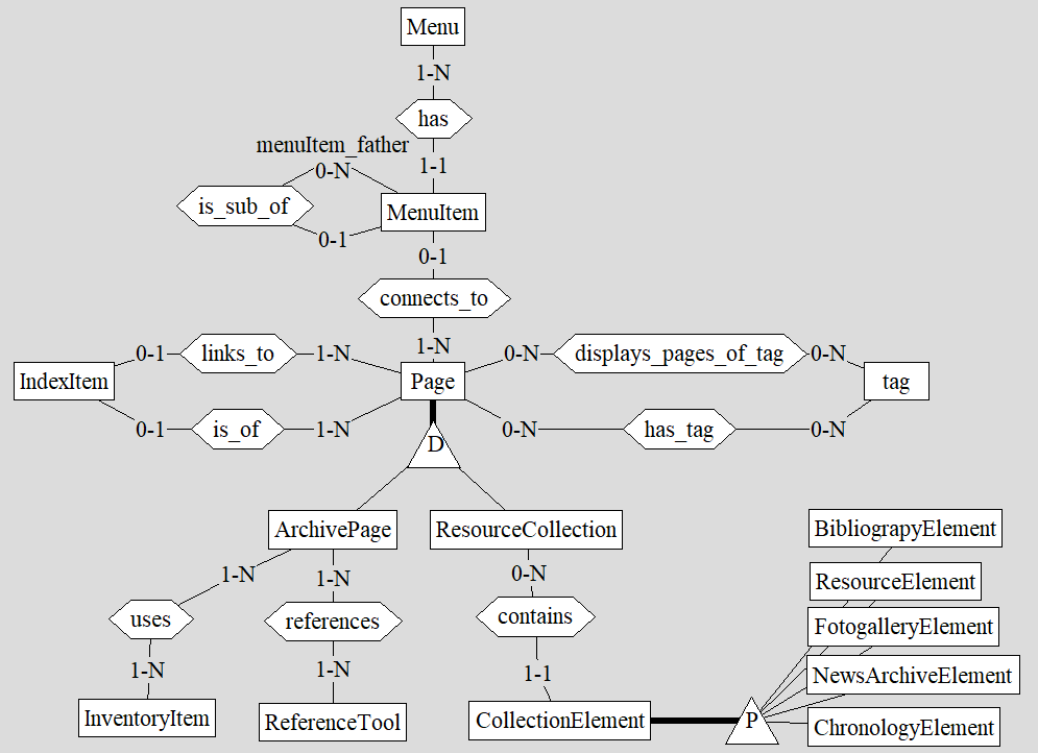

Figura 3.3: Schema E/R semplificato

possa essere pensato su misura per le esigenze di memorizzazione e gestione delle informazioni dell'Istituto.

3.4.1 Struttura e funzioni del nuovo database

A partire dall'analisi dell'architettura dei database precedentemente impiegati e delle informazioni contenute nei siti dell'Istituto, è stato progettato un database di tipo relazionale con l'ausilio di MySQL Workbench, uno strumento grafico di sviluppo per database.

Lo schema E/R del database finale è consultabile alla figura 3.3. Per ragioni di semplicità, in questo schema non sono riportati gli attributi delle entità.

Di seguito saranno spiegate le entità principali del modello:

- Menu: l'entità Menu permette di memorizzare i dati di ogni menu. É stata utilizzata per il menu di navigazione, ma la sua struttura generale permette di essere utilizzata anche per altri menu.

- MenuItem: rappresenta i singoli elementi presenti in un menu. La sua auto-associazione permette di avere voci di menu annidate.

- Page: rappresenta una pagina web e permette la memorizzazione di dati generali come titolo, slug, contenuti testuali, visibilità e metadati.

- Tag: permette di assegnare categorie semantiche alle pagine. Tramite le associazioni tra Tag e Page è possibile modellare la relazione "contenitore-contenuto", di cui i siti dell'Istituto fanno molto uso. Un esempio è la pagina "Eventi" del sito del sito principale *istoreco-fc.it*, che racchiude tante sottopagine relative a singoli eventi organizzati dall'Istituto Storico.

- IndexItem: consente la memorizzazione delle voci di un indice, utilizzato in pagine particolarmente articolate, come quelle presenti nel sito dell'Istituto Storico *ladigacivile.eu*. Ogni istanza di IndexItem corrisponde ad una riga di un indice.

- ArchivePage: rappresenta una pagina dedicata agli archivi gestiti dall'Istituto. Questa entità è una speicializzazione di Page, e include informazioni aggiuntive, come strumenti di corredo, cronologie e tipologie di materiali presenti nell'archivio. Questi dati sono gestiti attraverso le entità InventoryItem e ReferenceTool. Questo approccio migliora la struttura dei dati e riduce la ripetizione delle informazioni quando vengono create nuove pagine di archivio.

- ResourceCollection: è una specializzazione di Page, e serve a mappare le pagine che si occupano di raccogliere risorse. Il loro principale uso è in *ladigacivile.eu* come pagine di supporto, collegate a pagine di articoli, in cui vengono mostrate tutte le risorse inerenti a tali articoli,

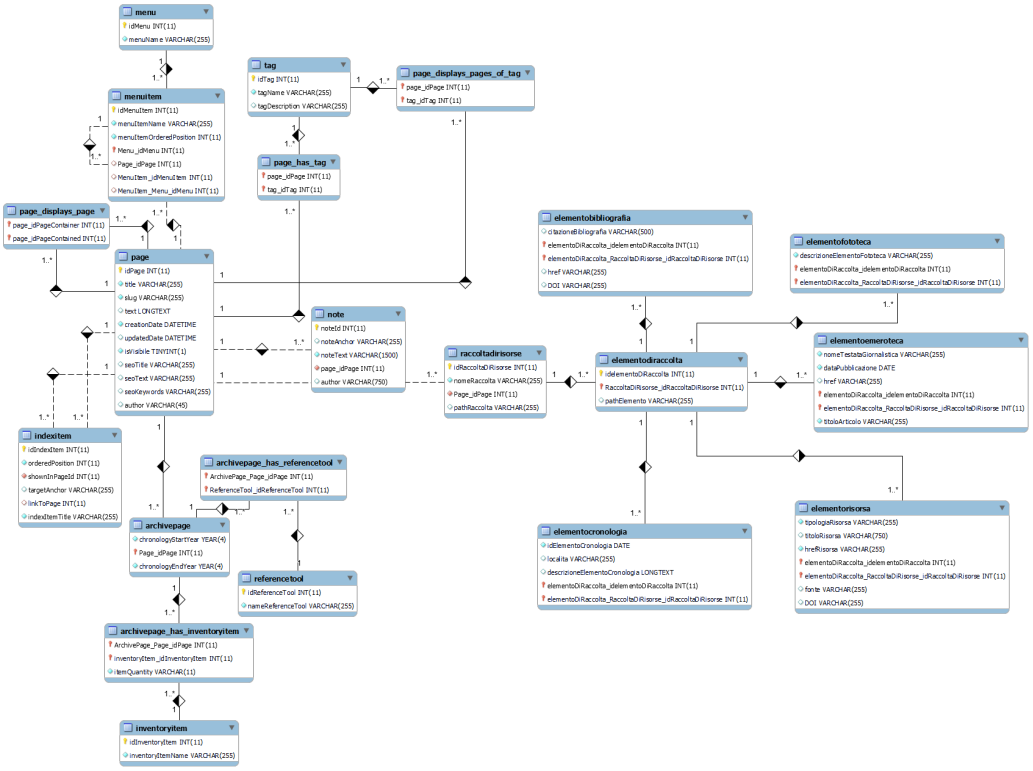

Figura 3.4: Modello EER finale

come bibliografie, videoteche ed emeroteche. Una ResourceCollection è creata a partire da un CollectionElement che, tramite gerarchia di generalizzazione, memorizza dati diversi a seconda che sia una risorsa in rete o un elemento di una bibliografia, galleria fotografica, emeroteca digitale o cronologia.

Questo modello crea una struttura solida per l'organizzazione dei dati dell'Istituto, che potrà supportare tutte le necessità di modifica e aggiornamento di pagine al sito da parte degli amministratori.

Il modello relazionale EER creato con MySQL Workbench è consultabile alla figura 3.4.

3.4.2 Popolamento del database

Per migrare i dati dai database SQLite al nuovo database MySQL, sono state prese in considerazione diverse soluzioni automatizzate. Tuttavia, query e script sono stati usati solo in casi specifici, per diverse ragioni. In primo luogo, il nuovo database ha uno schema molto diverso rispetto a quelli dei database SQLite in uso. Sebbene alcune tabelle rimanessero simili, la maggior parte aveva una struttura significativamente differente, con tabelle, colonne, tipi di dati e relazioni particolari, che rendevano complicato l'inserimento automatizzato. A causa della necessità di adattare alcuni dati al nuovo schema, ad esempio separando informazioni in colonne divise, o riformattandole, è stato necessario un certo controllo manuale sull'inserimento, per risolvere le situazioni più complesse. Inoltre, non tutte le informazioni da mappare erano effettivamente memorizzate in database. In alcuni casi, per le informazioni già sufficientemente strutturate, sono stati usati script di inserimento dei dati a partire dal loro formato HTML.

3.5 Sviluppo del sito web

Al termine della memorizzazione delle informazioni nel nuovo database, è iniziata la fase di sviluppo del progetto. Il progetto si è concentrato sulle funzionalità di visualizzazione delle informazioni da parte dell'utente, quindi sul recupero dei dati, sulla loro organizzazione e visualizzazione a schermo.

3.5.1 Tecnologie scelte

Durante lo sviluppo del progetto, si è deciso di utilizzare una combinazione di JavaScript, React, PHP e MySQL per la gestione dei contenuti, l'interazione con il database e la visualizzazione dei dati. Ogni tecnologia è stata selezionata per soddisfare determinate esigenze del progetto, ed ottenere una buona esperienza utente e una gestione robusta dei dati.

Per il front-end, la scelta del linguaggio è ricaduta su JavaScript, in combinazione con la libreria React. Questa scelta è stata dettata dalla fondamentale importanza del ruolo di JavaScript nella realizzazione di pagine web interattive e dinamiche. Per massimizzare la riusabilità del codice frontend, e per migliorare modularità e manutenibilità del progetto, è stato utilizzato React. React segue un'architettura basata su componenti, ed ha permesso di suddividere l'interfaccia in unità. Questo approccio semplifica la modifica delle singole parti della UI. Offre inoltre una gestione dello stato interno che permette di memorizzare e gestire dati dinamici e l'aggiornamento delle singole componenti, senza dover ricaricare tutta la pagina.

Per il back-end, è stato utilizzato PHP per gestire il lato server-side. PHP è una delle soluzioni più diffuse per la gestione di applicazioni web, per la sua semplicità e velocità nello sviluppo e per la sua compatibilità con MySQL.

Per l'interazione tra front e back-end sono state realizzate API RESTful, che hanno gestito le richieste al server con chiamate HTTP utilizzando JavaScript. La risposta è stata trasmessa in formato JSON. Utilizzare API ha permesso la separazione tra front-end e back-end, mantenendo chiaro il ruolo di ogni parte dell'applicazione.

3.5.2 Struttura del progetto

Il progetto è stato strutturato seguendo un'architettura di tipo Model-View-Controller (MVC), che permette di separare le responsabilità delle diverse parti del codice, al fine di garantire scalabilità e manutenibilità.

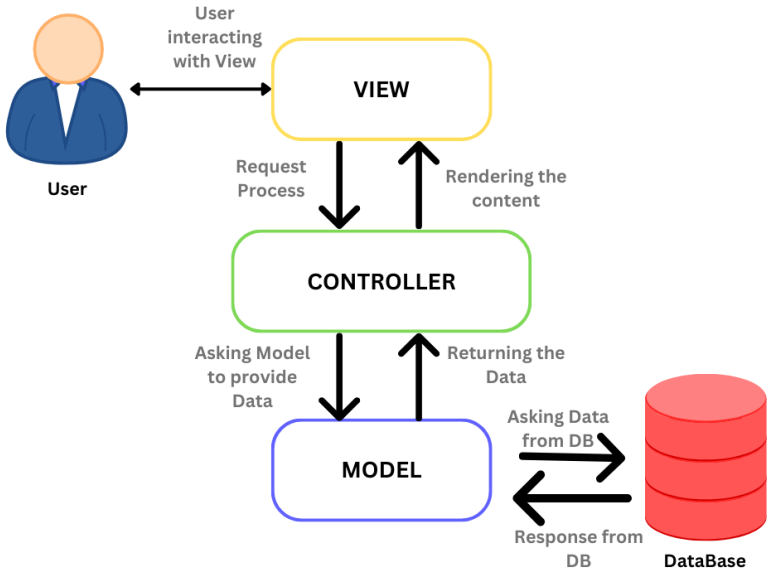

Figura 3.5: Schema esemplificativo del pattern MVC usato nel progetto. [2]

La parte di View è costituita dal front-end del sito, cioè dalla visualizzazione a schermo dei dati e dell'interazione con l'utente. La parte del Model è costituita dalla parte di back-end che si occupa dell'interazione con il database, al fine di ottenere le informazioni da visualizzare. La parte del Controller è costituita dalle API, che si occupano dell'interazione tra View e Model, richiedendo il raccoglimento dei dati necessari alla View per visualizzare i contenuti delle pagine. L'interazione tra le diverse parti è illustrata nella figura 3.5.

Il progetto è organizzato in due cartelle fondamentali, *src* per il front-end e *backend* per il back-end.

Contenuti della cartella *src*

La cartella *src* contiene il codice React e si suddivide in tre sottocartelle principali:

1. *components*: questa cartella contiene tutti i custom component React realizzati per il progetto al fine di massimizzare la riusabilità del codice e mantenere un design coerente. I custom component principali creati per il sito sono:

    - Un component per la visualizzazione delle informazioni delle pagine di archivio.
    - Un component per uno slider di contenuti da utilizzare al posto dei caroselli. Utilizza lo spazio in maniera più efficace rispetto ad un carosello, mostrando diversi contenuti contemporaneamente quando in modalità desktop, ed un solo contenuto alla volta quando in modalità mobile.
    - Un componente per il footer, che permette di modifcare in un unico punto la grafica e le informazioni da visualizzare al suo interno.
    - Un componente per la home page e uno per le pagine. La decisione di avere un componente specificamente per la home page è dovuta alle notevoli differenze grafiche e contenutistiche tra essa e le altre pagine visualizzabili nel sito.
    - Un componente per gli indici.
    - Un componente che gestisce il layout, e si occupa di integrare i componenti di navbar, logo, titolo e footer, centralizzando in un unico punto il codice che gestisce le disposizioni di tali elementi.
    - Un componente per il logo e per il titolo.
    - Un componente generico per i menu, che viene utilizzato anche per il menu di navigazione.
    - Un componente per la sezione dei progetti di rete visibile nella home.
    - Un componente per le sottopagine.

2. *routes*: questa cartella gestisce le rotte utilizzate nel sito, che sono solo due: "/" per la home page e "/:slug" per generare dinamicamente le altre pagine. La gestione dinamica delle pagine permette di evitare la creazione di file HTML statici, rendendo il sito più flessibile e scalabile.

3. *stylesheets*: contiene tutti i fogli di stile CSS necessari per la resa grafica delle pagine e dei componenti.

La View si collega al Controller tramite il file *ApiClient* che fornisce funzioni per la richiesta delle informazioni da visualizzare nel menu di navigazione e nelle pagine.

Contenuti della cartella *backend*

Questa cartella è organizzata in tre sottocartelle fondamentali:

1. *config*: Questa sottocartella contiene il file *DatabaseHelper.php* che è una classe che si occupa della configurazione del database e che contiene tre funzioni per l'interazione con esso: una per costruire le query, una per il binding dei parametri, e una per eseguire le query e prendere il risultato. Questa organizzazione permette di astrarre l'interazione con il database, evitando ripetizioni nel codice e semplificando la gestione delle query.

2. *controller*: contiene i file con funzione di Controller, gestendo le chiamate API per l'interazione del database in risposta a esigenze della View. Al suo interno si trovano gli endpoint delle API per il menu di navigazione e per le pagine. Quest'ultima API si occupa della restituzione dei dati necessari alla visualizzazione di pagine (come testo, caratteristiche legate alla natura della pagina, e indice).

3. *models*: contiene le interfacce PageModelInterface e MenuItemModelInterface e le loro implementazioni, le classi PageModel e MenuItemModel. Queste due classi si occupano, tramite l'utilizzo delle funzioni di DatabaseHelper, di interagire con il database ed eseguire le query

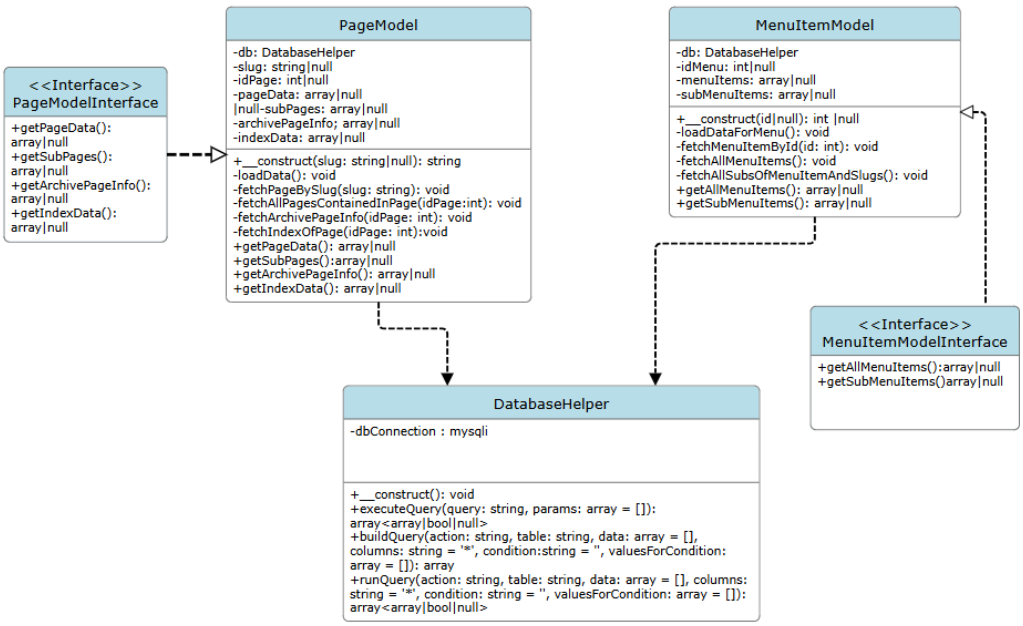

Figura 3.6: Diagramma delle classi di back-end.

per la raccolta delle informazioni necessarie alla view per visualizzare le pagine e il menu di navigazione.

Alla figura 3.6 è possibile consultare il diagramma UML delle classi relativo alle classi del back-end.

Il diagramma dei componenti in figura 3.7mostra con ulteriore dettaglio implementativo l'interazione tra le componenti del progetto. L'interfaccia grafica, realizzata con il supporto di React, utilizza API per la raccolta dei

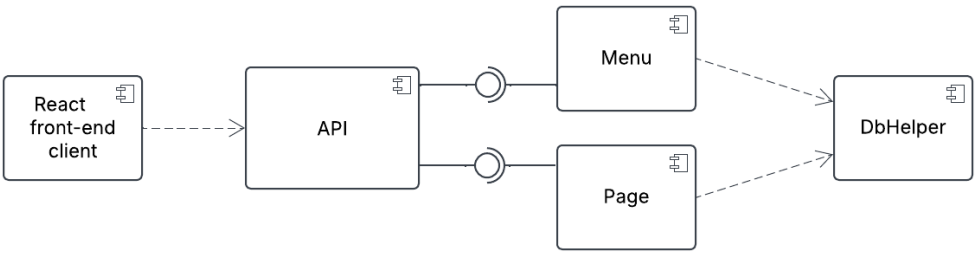

Figura 3.7: Diagramma dei componenti

dati da visualizzare. Queste API comunicano tramite interfacce con i modelli di menu e pagine, che si servono di un databaseHelper per le operazioni sul database.

3.6 Grafica

L'interfaccia grafica è stata mantenuta molto semplice, ed è stata basata sui mockup realizzati e sulle scelte stilistiche del sito principale *istorecofc.it*, in modo da dare una continuità visiva con l'identità precedentemente stabilita dall'Istituto.

3.6.1 Home

L'header è responsive, pertanto il suo contenuto dipende dalla grandezza dello schermo: nella versione desktop, visualizza il nome completo dell'Istituto Storico e il logo, insieme al menu di navigazione. Invece nella versione mobile viene visualizzato il nome abbreviato dell'Istituto e un hamburger icon, che nasconde al suo interno il menu di navigazione. In entrambe le versioni, l'header è posizionato sul margine superiore dello schermo ed è sempre visibile. In questo modo, è sempre raggiungibile in qualsiasi parte della pagina l'utente si trovi, migliorando la navigazione.

Lo slider di contenuti è responsive e mostra fino a tre elementi contemporaneamente nella versione desktop, ed un solo elemento alla volta in quella mobile. É possibile muoversi tra gli elementi attraverso l'uso delle frecce laterali.

Sono responsive anche il container dei progetti di rete e il footer, che dispongono i propri contenuti in orizzontale o in verticale a seconda delle dimensioni dello schermo.

La home page, completa di header, footer, menu di navigazione e slider, sia in versione desktop che mobile, è visibile alla figura 3.8.

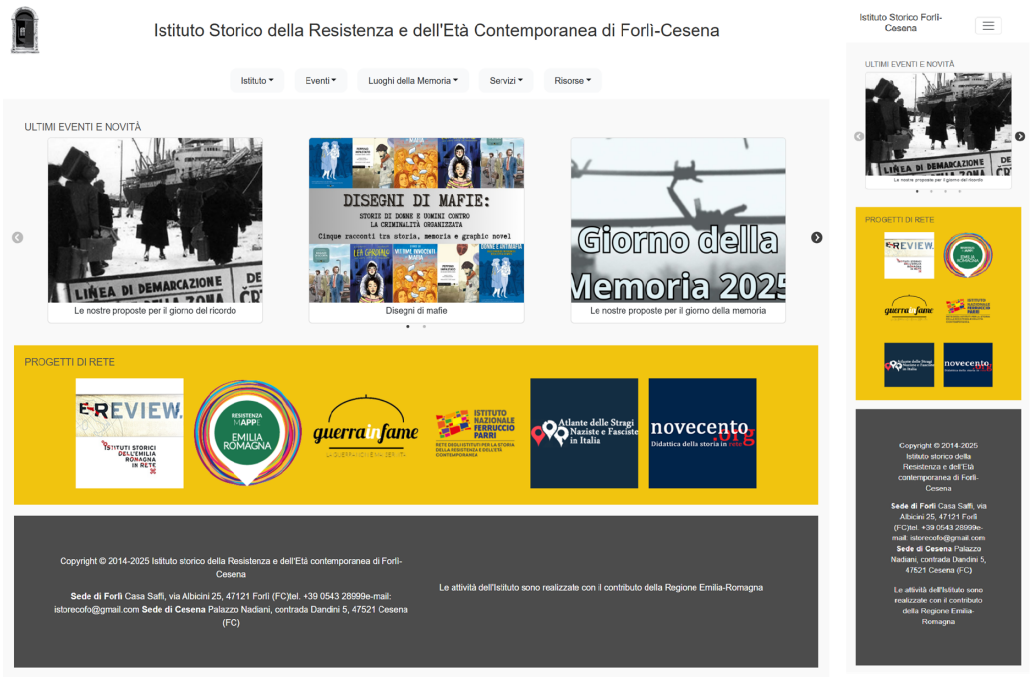

Figura 3.8: Home in versione desktop e mobile

3.6.2 Pagine

Le pagine hanno una grafica minimalista: se sono pagine semplici, allora presentano un blocco di testo. Nel caso in cui contengano altre pagine, queste sono visualizzate attraverso preview che fungono anche da collegamento alle pagine complete. Un esempio di questa tipologia di pagina è visibile alla figura 3.9.

Le pagine di archivio presentano una sezione aggiuntiva in cui vengono visualizzate tramite elenco puntato le informazioni fondamentali, come strumenti di corredo, materiali e cronologia. Gli indici sono visualizzati in una tabella.

Tutti questi elementi sono responsive e si ridimensionano in base alle dimensioni dello schermo.

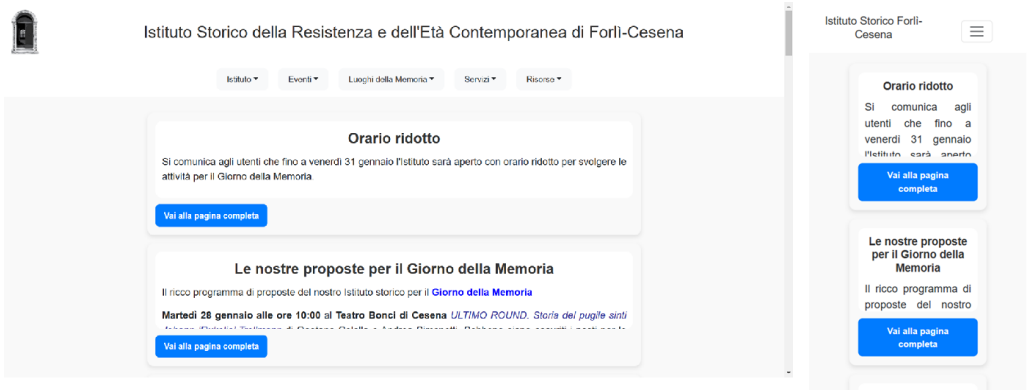

Figura 3.9: Esempio di pagina contenitore e sottopagine, in versione desktop e mobile

Conclusioni

Il progetto di unificazione dei siti dell'Istituto Storico della Resistenza e dell'Età Contemporanea di Forlì-Cesena si colloca nell'ambito dei Web Content Management Systems (WCMS), con l'obiettivo di sviluppare una piattaforma intuitiva, funzionale e in grado di rispondere alle esigenze di condivisione e gestione dei contenuti dell'Istituto. Il sistema progettato è stato pensato per strutturare e organizzare una vasta gamma di risorse, garantendo una navigazione coerente e un accesso più agevole alle informazioni storiche e documentali offerte dall'Istituto.

Per lo sviluppo del progetto, è stata condotta un'analisi approfondita delle tecnologie precedentemente adottate, seguita da uno studio delle tipologie di informazioni da gestire. Successivamente, è stata progettata l'architettura del database per supportare le necessità della piattaforma e, infine, è stata realizzata l'interfaccia utente per la visualizzazione e l'interazione con i dati. La scelta dell'architettura del progetto è ricaduta sul pattern Model-View-Controller (MVC), che ha consentito di separare logicamente le diverse responsabilità del sistema, migliorandone la manutenibilità e l'estensibilità.

Sebbene il progetto abbia gettato solide basi per la gestione unificata dei contenuti, vi sono ancora diversi aspetti che potranno essere sviluppati e migliorati in futuro.

I seguenti aspetti hanno la priorità nella realizzazione:

- Completare la grafica delle pagine più complesse, come quelle dedicate alle raccolte di risorse.

- Implementare un'interfaccia amministrativa con relatva logica di interazione con il database, per permettere agli operatori dell'Istituto di inserire, modificare e organizzare i contenuti in maniera semplice ed efficace.

Invece, di minor importanza, ma che migliorerebbero la qualità dell'esperienza utente, si potrebbero considerare i seguenti miglioramenti:

- Integrazione con API per social media, in modo da permettere la condivisione automatica di nuovi contenuti in automatico su diverse piattaforme come Facebook, Twitter e Instagram, andando ad ampliare la visibilità dei contenuti pubblicati dall'Istituto Storico.

- Inserimento di meccanismi di ricerca avanzata e filitri di informazioni.

In conclusione, il progetto rappresenta un primo passo significativo verso una gestione più moderna e centralizzata dei contenuti dell'Istituto Storico. L'adozione di tecnologie scalabili e di un'architettura modulare permetterà nel tempo di ampliare le funzionalità del sistema, garantendo un miglioramento continuo e un adattamento alle esigenze future dell'Istituto e del suo pubblico.

Bibliografia

[1] June 2023 analysis - June 2023, . URL https://joost.blog/cms-market-share/june-2023/.

[2] Medium: Read and write stories., . URL https://medium.com.

[3] Deane Barker. *Web Content Management: Systems, Features, and Best Practices*. "O'Reilly Media, Inc.", March 2016. ISBN 978-1-4919-0808-2. Google-Books-ID: 6LrNCwAAQBAJ.

[4] Elizabeth Shepherd and Geoffrey Yeo. *Managing Records: A Handbook of Principles and Practice*. Facet Publishing, 2003. ISBN 978-1-85604-370-0. Google-Books-ID: BdMqDgAAQBAJ.

[5] Bob Boiko. *Content Management Bible*. John Wiley & Sons, November 2005. ISBN 978-0-7645-8364-3. Google-Books-ID: p6nUDn3ZaBoC.

[6] Homer L. Hall and Aaron Manfull. *Behind the Screen: Running a Successful News Website*. The Rosen Publishing Group, Inc, December 2014. ISBN 978-1-4994-6022-3. Google-Books-ID: 73NhDwAAQBAJ.

[7] Pranshu Aggarwal. The Rise of Headless CMS: Empowering Modern Web Development, September 2023. URL https://medium.com/@pranshu1902/the-rise-of-headless-cms-empowering-modern-web-development-9b42fe382055.

[8] What is CMS architecture?, . URL https://www.sitecore.com/resources/insights/development/what-is-cms-architecture.

[9] About, . URL https://wordpress.org/about/.

[10] Jessica Neuman Beck and Matt Beck. *WordPress*. Peachpit Press, 2012. ISBN 978-0-321-79266-2. Google-Books-ID: 3kppd_qctwIC.

[11] Brian Messenlehner and Jason Coleman. *Building Web Apps with WordPress: WordPress as an Application Framework*. "O'Reilly Media, Inc.", April 2014. ISBN 978-1-4493-6480-9.

[12] Requirements, . URL https://wordpress.org/about/requirements/.

[13] Patrice-Anne Rutledge. *WordPress on Demand*. Que Publishing, May 2013. ISBN 978-0-13-325620-8. Google-Books-ID: l8Niu5ngfKgC.

[14] Thord Daniel Hedengren. *Smashing WordPress: Beyond the Blog*. John Wiley & Sons, February 2010. ISBN 978-0-470-71050-0. Google-Books-ID: QQ8OBAAAQBAJ.

[15] Features, . URL https://wordpress.org/about/features/.

[16] The Joomla! Project. Joomla! - Content Management System to build websites & apps. URL https://www.joomla.org/about-joomla.html.

[17] Savan K.Patel, V.R. Rathod, and Jigna B. Prajapati. "Performance Analysis of Content Management Systems Joomla, Drupal and WordPress". *International Journal of Computer Applications*, 21(4):39–43, May 2011. ISSN 09758887. doi: 10.5120/2496-3373. URL http://www.ijcaonline.org/volume21/number4/pxc3873373.pdf.

[18] O. S. M. Secretary. Joomla! Benefits & Core Features: multilingual, well supported... URL https://www.joomla.org/core-features.html.

[19] Home | Drupal.org, . URL https://new.drupal.org/home.

[20] Todd Tomlinson. *Beginning Drupal 7*. Apress, December 2010. ISBN 978-1-4302-2860-8. Google-Books-ID: P3Spl1IVitYC.

[21] Strapi - Open source Node.js Headless CMS , . URL `https://strapi.io/`.

[22] Khalid Elshafie and Mozafar Haider. *Designing Web APIs with Strapi: Get started with the Strapi headless CMS by building a complete learning management system API.* Packt Publishing Ltd, February 2022. ISBN 978-1-80056-670-5. Google-Books-ID: 5NpaEAAAQBAJ.

[23] WHATWG. (n.d.). HTML Standard. URL `https://html.spec.whatwg.org/multipage/introduction.html#introduction`.

[24] Markup - MDN Web Docs Glossary: Definitions of Web-related terms | MDN, July 2024. URL `https://developer.mozilla.org/en-US/docs/Glossary/Markup`.

[25] Deborah S. Ray and Eric J. Ray. *Mastering HTML and XHTML.* John Wiley & Sons, February 2006. ISBN 978-0-7821-5246-3. Google-Books-ID: ym8nKP7JdHwC.

[26] Cascading Style Sheets, . URL `https://www.w3.org/Style/CSS/Overview.en.html`.

[27] CSS Snapshot 2023, . URL `https://www.w3.org/TR/CSS/`.

[28] Eric A. Meyer. *CSS: The Definitive Guide: The Definitive Guide.* "O'Reilly Media, Inc.", November 2006. ISBN 978-1-4493-9725-8. Google-Books-ID: rdtCRLXAL78C.

[29] Get started with Bootstrap · Bootstrap v5.3, . URL `https://getbootstrap.com/docs/5.3/getting-started/introduction/`.

[30] and Bootstrap Mark Otto contributors, Jacob Thornton. Grid system. URL `https://getbootstrap.com/docs/4.0/layout/grid/`.

[31] Jacob Lett. *Bootstrap 4 Quick Start: Responsive Web Design and Development Basics for Beginners.* Bootstrap Creative, June 2018. ISBN 978-1-7322058-1-9. Google-Books-ID: 3fVhDwAAQBAJ.

[32] Ashok Appu. *Making Use of PHP*. John Wiley & Sons, October 2002. ISBN 978-0-471-42869-5.

[33] Paul Hudson. *PHP in a Nutshell: A Desktop Quick Reference*. "O'Reilly Media, Inc.", October 2005. ISBN 978-1-4493-7912-4. Google-Books-ID: dm2_jgULbBUC.

[34] David Flanagan. *JavaScript: The Definitive Guide: The Definitive Guide*. "O'Reilly Media, Inc.", August 2006. ISBN 978-0-596-55447-7. Google-Books-ID: 2weL0iAfrEMC.

[35] JavaScript | MDN, January 2025. URL `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.

[36] Prototype-based programming - MDN Web Docs Glossary: Definitions of Web-related terms | MDN, June 2023. URL `https://developer.mozilla.org/en-US/docs/Glossary/Prototype-based_programming`.

[37] Document Object Model (DOM) - Web APIs | MDN, December 2023. URL `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model`.

[38] Dr Ashad ullah Qureshi. *React JavaScript Notes For Professionals*. Concepts Books Publication, March 2023. ISBN 979-8-3863-8579-8. Google-Books-ID: s9eyEAAAQBAJ.

[39] React, . URL `https://react.dev/`.

[40] Quick Start – React, . URL `https://react.dev/learn`.

[41] Adam Boduch. *React and React Native*. Packt Publishing Ltd, March 2017. ISBN 978-1-78646-957-1. Google-Books-ID: jLkrDwAAQBAJ.

[42] MySQL :: MySQL 8.4 Reference Manual :: 1.2.2 The Main Features of MySQL, . URL `https://dev.mysql.com/doc/refman/8.4/en/features.html`.

[43] Michael Kofler. *The Definitive Guide to MySQL 5*. Apress, November 2006. ISBN 978-1-4302-0071-0.

[44] About npm | npm Docs, . URL `https://docs.npmjs.com/about-npm`.

[45] CLI Commands | npm Docs, . URL `https://docs.npmjs.com/cli/v11/commands`.

[46] Daniel Jacobson, Greg Brail, and Dan Woods. *APIs: A Strategy Guide*. "O'Reilly Media, Inc.", 2012. ISBN 978-1-4493-0892-6. Google-Books-ID: pLN7BxMTg7IC.

[47] Mark Masse. *REST API Design Rulebook*. "O'Reilly Media, Inc.", October 2011. ISBN 978-1-4493-1050-9. Google-Books-ID: 4lZcsRwXo6MC.

[48] Edwin Cano. *Mastering XAMPP: A Comprehensive Guide to Managing a Local Server*. Edwin Cano, December 2024. Google-Books-ID: f242EQAAQBAJ.

[49] Ben Laurie and Peter Laurie. *Apache: The Definitive Guide*. "O'Reilly Media, Inc.", 2003. ISBN 978-0-596-00203-9. Google-Books-ID: 1z6QfgsnpKsC.

[50] Git, . URL `https://git-scm.com/`.

[51] Git - About Version Control, . URL `https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control`.

[52] Git - A Short History of Git, . URL `https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git`.

[53] Git - What is Git?, . URL `https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F`.

[54] Sai Srinivas Sriparasa. *JavaScript and JSON Essentials*. Packt Publishing Ltd, October 2013. ISBN 978-1-78328-604-1. Google-Books-ID: MZOkAQAAQBAJ.

[55] "Brand Consistency": cos'è e perché è importante rimanere coerenti su tutti i canali, June 2022. URL https://www.agendadigitale.eu/mercati-digitali/brand-consistency-cose-e-perche-e-importante-rimanere-coerenti-su-tutti-i-can

Ringraziamenti

Giunta alla fine del mio percorso di laurea triennale, ci tengo a dedicare i miei più sentiti ringraziamenti a tutte le persone che, in modi diversi, hanno reso possibile il raggiungimento di questo traguardo.

Innanzitutto, desidero ringraziare il mio relatore, il professor Giovanni Delnevo, per la sua disponibilità e per i suoi preziosi consigli, che mi hanno guidata nella realizzazione di questa tesi. Un sentito ringraziamento va anche ai miei correlatori, la professoressa Silvia Mirri e il dottor Manuel Andruccioli, per il loro aiuto durante questo percorso.

Un ringraziamento speciale va alla mia famiglia, che è stata il mio punto di riferimento in ogni momento. A mio papà Marco e a mia mamma Giuliana, per avermi sempre sostenuta con amore incondizionato, dandomi la forza per affrontare ogni sfida, e per aver creduto in me anche quando io stessa volevo mollare. A mia sorella Anna, per la gioia e spensieratezza che porta in me ogni giorno, e per essere la sorellina migliore che potessi desiderare.

Con il cuore, voglio ringraziare mia nonna Guglielma, che mi ha vista intraprendere il mio percorso universitario ma che non può essere qui con me a festeggiare per il suo compimento. Spero che tu sia fiera della persona che sono diventata, e dei traguardi che ho raggiunto.

Ringrazio con l'anima i miei nonni Guido e Teresina, che mi hanno cresciuta con amore, e sono al mio fianco da tutta la vita. Vi ringrazio per aver condiviso i miei momenti di gioia e di dolore vissuti in questi anni.

Un grazie di cuore a mia cugina Chiara e ai miei zii Andrea e Barbara, per essere sempre stati una presenza costante nella mia vita, e per le tante

giornate felici trascorse insieme.

Ringrazio anche Monica, per aver festeggiato con me i miei successi di questi anni, e Claudio, che sono sicura che abbia gioito con noi.

Passando agli amici, voglio ringraziare le persone che hanno condiviso con me questo viaggio, rendendolo speciale.

Alle mie amiche di sempre, Bally, Desy e Leuz, per il loro affetto incondizionato, e per aver condiviso con me ogni momento, da quando ci siamo conosciute in prima superiore. Ringrazio anche Sara e Mattia, con cui ho approfondito i rapporti in questi ultimi anni, per avermi sempre fatta sentire inclusa.

Ringrazio Aleksandra, per il legame che nemmeno la distanza può scalfire, per le chiacchierate infinite e per essere una costante nella mia vita, anche se da lontano.

Un ringraziamento speciale ai miei amici di Cesena, che hanno reso la mia esperienza indimenticabile. Grazie a Chiara, Mati e Seba, per avermi accolta a braccia aperte dal primo laboratorio di programmazione C, e per avermi accompagnata in questo percorso.

Grazie ad Astro, Diotta e Pioli, per l'amicizia preziosa e le tante conversazioni condivise.

Grazie al "gruppo del gossip" : Ciccio, Dario, Marco e Veri, per avermi accolta nel vostro gruppo, e per averlo fatto sentire come mio.

E infine, un grazie speciale al gruppo JMP: Giak, Gino, Gio, Gigi, Manfro, Mirco, Piso, Ricky e Tom, per tutte le giornate, le serate e gli aperitivi condivisi, e per il legame che ci unisce.

Grazie di cuore a tutti coloro che, con il loro affetto e supporto, hanno reso questo percorso meraviglioso e indimenticabile.